LARGE LANGUAGE MODELS CAN LEARN RULES

Anonymous authors
Paper under double-blind review

ABSTRACT

When prompted with a few examples and intermediate steps, large language models (LLMs) have demonstrated impressive performance in various reasoning tasks. However, prompting methods that rely on implicit knowledge in an LLM often hallucinate incorrect answers when the implicit knowledge is wrong or inconsistent with the task. To tackle this challenge, we present Hypotheses-to-Theories (HtT), a framework that learns a rule library for reasoning with LLMs. HtT contains two stages, an induction stage and a deduction stage. In the induction stage, an LLM is asked to generate and verify rules over a set of training examples. Rules that appear and lead to correct answers sufficiently often are collected to form a rule library. In the deduction stage, the LLM is prompted to employ the learned rule library to perform reasoning to answer test questions. Experiments on numerical reasoning and relational reasoning tasks show that HtT improves existing prompting methods, with an absolute gain of 11-27% in accuracy. The learned rules are transferable to different models and to different forms of the same task.

1 INTRODUCTION

Coinciding with their tremendous growth in scale, large language models (LLMs) (Brown et al., 2020; Chowdhery et al., 2022; OpenAI, 2023; Anil et al., 2023, *inter alia*) have demonstrated emergent capabilities across a wide range of reasoning problems (Wei et al., 2022b; Bubeck et al., 2023), including program synthesis, arithmetic reasoning, symbolic reasoning and commonsense reasoning. Importantly, these abilities are commonly elicited by advanced prompting techniques (Wei et al., 2022c; Zhou et al., 2023; Khot et al., 2023) that teach an LLM to decompose a complex task into simple steps and perform reasoning step by step based on a small set of in-context examples.

Despite their abilities to solve complex reasoning problems, an open challenge for LLMs is their tendency to generate outputs that appear plausible but contradict knowledge of the real world, a phenomenon that is often referred to as hallucination (Ji et al., 2023; Zhang et al., 2023b; McKenna et al., 2023). As an additional challenge, whenever the problem deviates from requiring conventional knowledge (e.g. arithmetic in a non-decimal system), LLMs typically exhibit a significant drop in accuracy, sometimes performing only slightly better than random guessing (Tang et al., 2023; Wu et al., 2023). Both phenomena are caused by a mismatch between the implicit knowledge pretrained in the language model and the knowledge required for the problem (Talmor et al., 2020). While it is always possible to curate a dataset and inject required knowledge into an LLM via supervised fine-tuning (Wang et al., 2021; Talmor et al., 2020; Nye et al., 2021), a generic approach to automatically discovering and applying knowledge in reasoning problems would be more desirable.

To achieve such a solution, we reconsider the process of scientific discovery (Bacon, 1620), which has allowed humans to develop effective theories of the world. Typically, a scientific discovery induces theories by generating and verifying hypotheses with experiments. The theories can be applied to solve problems in new scenarios. An interesting aspect of this process is that it can arrive at correct knowledge even if the initial set of hypotheses is noisy (e.g. geocentrism v.s. heliocentrism). In other words, the process of scientific discovery starts by allowing humans to freely "hallucinate" hypotheses, but theories are only kept if they are verified by experiments. Motivated by this observation, we propose Hypotheses-to-Theories (HtT), a framework that enables LLMs to automatically induce a rule library in reasoning problems. HtT is composed of an induction stage and a deduction stage. In the induction stage, an LLMs is asked to generate and verify rules for question-answer pairs in a given training set. We collect the rules and filter them based on their number of occurrence and frequency of association with correct answers, forming the rule library for the deduction stage. In the deduction stage, the LLM is then asked to apply the learned rules to solve the reasoning problem, thereby reducing the chance of hallucination. To reduce the effort required for prompt engineering, we propose induction from deduction, which fuses the rule generation and verification steps into a single deduction-like step. In this way, the prompts for both stages can be easily derived from existing few-shot prompting methods, such as chain-of-thought or least-to-most prompting.

Empirically, we verify the effectiveness of HtT with GPT3.5 and GPT4 (OpenAI, 2023) on the Arithmetic (Wu et al., 2023) and CLUTRR (Sinha et al., 2019) datasets, which correspond to numerical reasoning and relational reasoning respectively. Experiments show that HtT consistently improves over baseline prompting methods across the models and datasets considered, with an absolute gain of 11-27% in most cases. Moreover, the learned rules can be directly transferred to the textual version of CLUTRR, providing a practical advantage over previous reasoning approaches. We also conduct extensive ablation studies to understand the properties of HtT, finding that the performance gain arises primarily from a reduction in rule hallucination due to the use of the learned rules. We also observe a log-linear scaling law between accuracy and the number of training examples.

2 PRELIMINARIES

Intuitively, reasoning is the process of applying sound logical steps to draw conclusions from existing information. At the core of reasoning are two concepts, facts and rules. Facts are pieces of information that describe the current state of the problem, while rules are functions that transform facts into new facts under certain conditions. Take the famous syllogism as an example.

Socrates is a man. All men are mortal. Therefore, Socrates is mortal.

Here the statements *"Socrates is mortal"* and *"Socrates is a man"* are both facts. *"All men are mortal"* could be thought of as a rule, with the understanding that facts and rules are relative concepts intuitively. That is, a rule can also be viewed as a fact when coupled with a high-order rule, e.g. *"All men are mortal"* is a fact when we consider *"If all men are animals, then all men are mortal"*.

Deductive Reasoning. Deductive reasoning aims to derive new facts based on known facts and rules. The above syllogism forms a simple deductive step. When there are multiple facts and rules, one needs to match rules against facts, which is known as *unification* in logic. In general, there are many ways to apply unification at each step, and a planning algorithm is usually required to find the optimal ordering of steps, e.g. forward chaining or backward chaining (Stuart & Peter, 2016).

For LLMs, most prompting techniques are designed to elicit deductive reasoning. For example, chain-of-thought (Wei et al., 2022c) and least-to-most (Zhou et al., 2023) prompting teach an LLM to deduce conclusions from given facts, assuming a greedy plan. We categorize these as implicit reasoning methods because they rely on implicit rules stored in the LLM. By contrast, explicit reasoning methods such as Selection-Inference (Creswell et al., 2023) and LAMBADA (Kazemi et al., 2023) operate on given facts and rules, with a focus on searching an optimal deduction order.

Inductive Reasoning. Inductive reasoning focuses on deriving general rules from observed facts. For example, if we know *"Scorates is a man"* and *"Socrates is mortal"* and the same facts hold for *Aristotle*, we might hypothesize a rule *"All men are mortal"*. While there are a lot of rules one can induce from given facts, some rules are overly specific to the given facts, e.g. *"If Scorates is a man, Socrates is mortal"* can only be applied to *Scorates*. To make rules useful for future prediction, they should have a large set of supporting examples (i.e. coverage) while also predicting new facts correctly (i.e. confidence) (Agrawal et al., 1994; Galárraga et al., 2013).

In machine learning, inductive reasoning is usually not a standalone task, since it is challenging to annotate and evaluate rules. Instead, inductiveness is studied as a desired property for deductive reasoning models to generalize to new facts. For example, in inductive logic programming (ILP) (Muggleton & De Raedt, 1994), models that learn algorithms like searching or sorting should generalize to new combinations of elements. In knowledge graph reasoning (Teru et al., 2020; Galkin et al., 2022), models that learn to answer queries on knowledge graphs should generalize to new entities.

3 HYPOTHESES-TO-THEORIES PROMPTING

Many real-world reasoning problems require deductive reasoning to reach correct answers. Often, a problem statement contains the necessary facts within the context, but the rules are not explicitly

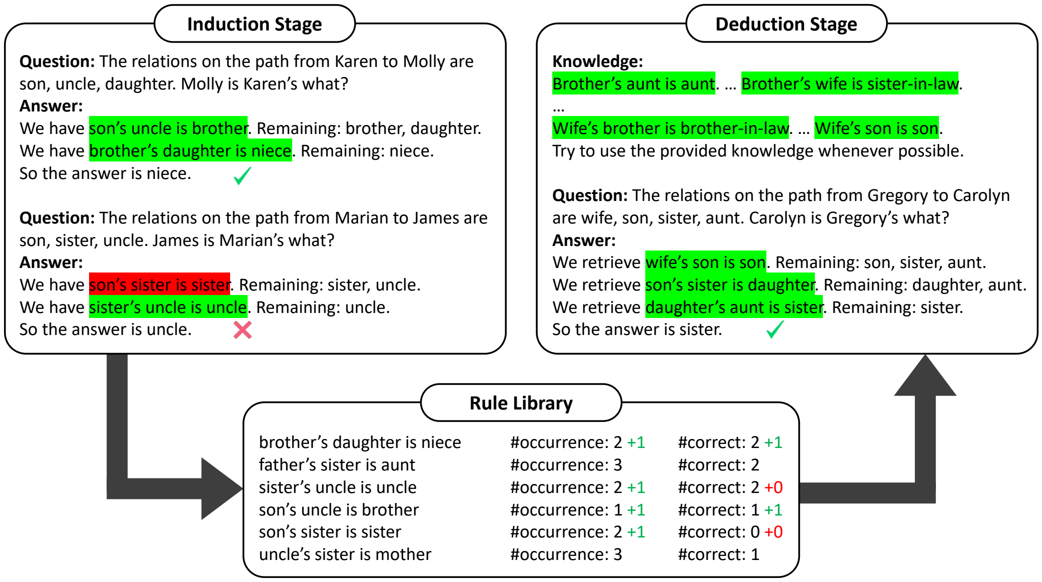

Figure 1: An example of Hypotheses-to-Theories applied to chain-of-thought for the kinship relation problem. Few-shot examples are omitted for brevity. The induction stage uses CoT to generate rules and verify them on the training samples. Rules are then collected and filtered to form the rule library. The deduction stage augments the CoT prompt with knowledge from the rule library. Correct and incorrect rules are marked with green and red respectively. Full prompts are in Appendix E.

stated. Consider arithmetic: when humans are faced with an arithmetic problem, they are not given a summation or multiplication table. In such cases, humans rely solely on the mathematical tables stored in their memory to perform deduction. An LLM pretrained on a massive corpus can retrieve certain commonsense knowledge from its parameters (Petroni et al., 2019; Roberts et al., 2020), but due to the implicit nature of this process, it can be prone to hallucinating unintended knowledge. Hallucination is especially common in problems that are not frequently covered in the pretraining data (Kandpal et al., 2023) or conflict with standard real life problem formulations (Longpre et al., 2021; Wu et al., 2023). A manual analysis indicates that rule hallucination constitutes 81% and 65% of the errors made by CoT on base-16 Arithmetic and CLUTRR respectively (Figure 3).

Interestingly, we find that hallucination in LLMs resembles hypothesis generation in scientific discovery. For example, humans historically debated the geocentric versus heliocentric hypotheses. Even though we know geocentrism is deprecated since it cannot explain many phenomena, it was a valid "hallucination" at a time when there was not enough evidence. In other words, science evolves by generating and verifying hypotheses, as well as consolidating theories for successive generations. Inspired by the process of scientific discovery, we propose Hypotheses-to-Theories (HtT) prompting to reduce hallucination in LLMs by learning a rule library. HtT consists of an induction stage and a deduction stage, both implemented by few-shot prompting. In the induction stage, rules are generated and verified on a set of question-answer examples. The rules are then collected and filtered to form a library. In the deduction stage, given a test question, we prompt the model to explicitly retrieve rules from the rule library to solve the problem. The two stages are similar to training and test in supervised learning, except that we learn rules explicitly instead of model parameters.

Scope of Tasks. Since HtT is built on top of a base few-shot prompting method, it shares the same scope of tasks with the base prompting method. However, to achieve performance gain with HtT, the following constraints apply: (1) There exist a moderately small set of rules that are reusable across most problems of the task, otherwise the rule library will be too large to fit into the context length of an LLM. (2) Rules generated by the LLM follow a template specified by few-shot exemplars. This is required to collect and filter rules accurately. (3) The base prompting method has a reasonable performance on the training samples, which is important for learning good rules. Compared to the tasks solved by common few-shot prompting methods (e.g. chain-of-thought), these constraints exclude tasks that are not logical (e.g. natural language entailment, reading comprehension) or require planning abilities (e.g. sudoku, minesweeper). Note that HtT does not impose constraints on the type of rules it learns, and is compatible with numerical rules and propositional logic rules.

3.1 Induction Stage: Rule Generation and Verification

To discover explicit knowledge for reasoning, we would like to use an LLM to induce rules for a task. For each training example (a question-answer pair), we ask the LLM to generate rules for answering the question. While the rules can be right or wrong, they can be verified from the ground truth answers. This verification can filter out rules associated with incorrect answers, keeping in the rule library only rules that are sufficiently associated with correct answers for the deduction stage. Following the principles of rule mining (Galárraga et al., 2013), we filter rules based on both coverage and confidence, requiring a rule to both be sufficiently frequent and exhibit a sufficient association with correct answers respectively. Specifically, we collect the rules generated by the LLM over the entire training set, and only keep those that appear more than k times and pass the verification test with a probability higher than p.

Induction from Deduction. The induction stage introduces two sub problems, rule generation and verification. Previous works on induction (Yang et al., 2022) use separate prompts for the generator and the verifier. While it is possible to design two separate prompts here as well, this doubles the prompt engineering effort and makes the comparison to other methods less intuitive. Moreover, due to the multi-step nature of reasoning problems, some rules can only be elicited after certain intermediate results are available. Hence we propose induction from deduction, which adapts a deductive reasoning prompt (e.g. CoT, LtM) for both rule generation and verification (Figure 1 left). The key idea is to explicitly declare rules whenever a deduction is performed. We then extract the rules from the model prediction, and use the accuracy on the training sample for verification.

3.2 Deduction Stage: Explicit Reasoning with A Rule Library

Once we discover a library of good rules, the next step is to apply these rules to solve test questions. To do so, we prepend the rule library in the prompt and ask the LLM to retrieve rules from the library to perform deductive reasoning. However, we find that even a strong LLM (e.g. GPT4) struggles to retrieve the correct rule at each step, possibly due to the large number of rules provided. As a remedy, we have developed an XML tagging trick to augment the in-context retrieval ability of LLMs. Note that it would be easy to offload the retrieval subproblem to a pretrained passage retriever (Karpukhin et al., 2020), but here we are more interested in how an LLM can perform deductive reasoning entirely on its own.

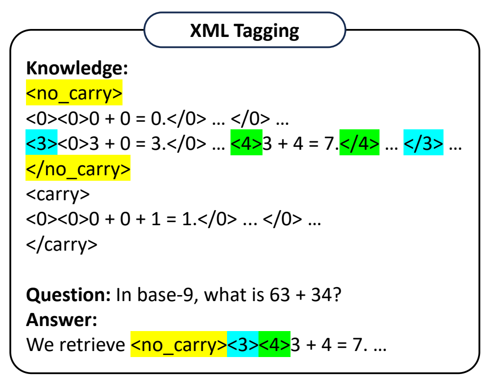

Figure 2: The XML tagging trick. With an XML hierarchy, we break down a hard retrieval problem into several easy retrieval problems.

In-Context Retrieval with XML tags. While LLMs often fail to retrieve the correct rule from a large set, we find that retrieval often succeeds when the number of rules are limited, e.g. to at most 10 rules. Therefore, a natural idea is to organize the rule set into a hierarchy, such that each step in the hierarchy only involves a small number of options. We manually define a hierarchy by grouping similar rules together. Inspired by the XML tags used in prompting tutorials[1], we label each level of the hierarchy with pairs of XML tags like <carry> and </carry> (Figure 2). In order to index through the hierarchy, we ask the LLM to generate the tags for each level before outputting the retrieved rule. We have found that the XML tagging trick significantly boosts the performance of HtT (Table 4).

4 Experimental Evaluation

To evaluate HtT, we apply it as an augmentation to existing few-shot prompting methods, including chain-of-thought and least-to-most prompting. We benchmark performance on two multi-step rea-

[1]https://docs.anthropic.com/claude/docs/constructing-a-prompt

Table 1: Results on Arithmetic. For reference, GPT4 (5-shot CoT) has 99.1% accuracy on base-10.

Model	Prompt	Base-16			Base-11			Base-9			Average
		2 digits	3 digits	4 digits	2 digits	3 digits	4 digits	2 digits	3 digits	4 digits	
GPT3.5	0-shot CoT	30.6	10.5	0.0	5.6	5.3	0.0	11.1	10.5	0.0	8.2
	5-shot CoT	83.3	34.2	11.5	5.6	2.6	0.0	25.0	13.2	11.5	20.8
	+ HtT	77.8	52.6	23.1	25.0	13.2	0.0	8.3	5.3	11.5	**24.1 (+3.3)**
	+ HtT (GPT4)	63.9	44.7	34.6	13.9	7.9	3.8	25.0	7.9	11.5	**23.7 (+2.9)**
	5-shot LtM	83.3	34.2	15.4	16.7	5.3	0.0	13.9	7.9	7.7	20.5
	+ HtT	80.6	39.5	26.9	16.7	2.6	3.8	19.4	5.3	3.8	**22.1 (+1.6)**
	+ HtT (GPT4)	72.2	31.6	30.8	47.2	15.8	11.5	44.4	21.1	15.4	**32.2 (+11.7)**
GPT4	0-shot CoT	72.2	26.3	7.7	22.2	10.5	3.8	30.6	34.2	23.1	25.6
	5-shot CoT	83.3	71.1	61.5	52.8	47.4	46.2	75.0	36.8	42.3	57.4
	+ HtT	100.0	94.7	84.6	88.9	71.1	46.2	86.1	68.4	65.4	**78.4 (+21.0)**
	5-shot LtM	88.9	81.6	61.5	52.8	47.4	30.8	52.8	31.6	11.5	51.0
	+ HtT	100.0	86.8	76.9	72.2	52.6	46.2	61.1	23.7	38.5	**62.0 (+11.0)**

soning tasks that are difficult for current few-shot prompting methods. We find that HtT improves the underlying few-shot prompting methods despite using almost the same prompt template. We also conduct extensive ablation studies to provide a more comprehensive understanding of HtT.

4.1 Experiment Setup

We evaluate HtT and the baselines using two different LLMs, `gpt-3.5-turbo` and `gpt-4`. When the prompts exceed the 4k context length of `gpt-3.5-turbo`, we use `gpt-3.5-turbo-16k` instead. The prompts never exceed the 8k context length of `gpt-4` in these experiments. Throughout the following, we will denote the two LLMs as GPT3.5 and GPT4 respectively. We use the default temperature of 1 for both models.

The baseline prompting methods include three prompting strategies: zero-shot chain-of-thought (0-shot CoT) (Kojima et al., 2022), few-shot chain-of-thought (5-shot CoT) (Wei et al., 2022c) and few-shot least-to-most (5-shot LtM) (Zhou et al., 2023). For the few-shot prompting methods, we choose the first 5 non-trivial examples from each dataset as exemplars for few-shot prompting, avoiding heavy engineering of the choice of exemplars. We keep the same set of exemplars across all few-shot prompting methods. For least-to-most prompting, we offload the generation and retrieval of rules to a separate prompt whenever we need a rule for deduction.

For the proposed HtT, we perform the induction stage on 2,000 training examples, unless stated otherwise. When the training set contains fewer than 2,000 examples, we resample examples to generate rules. In the deduction stage, we add the learned rule library to the system message, so that the LLM knows to treat the knowledge and exemplars differently. We search the hyperparameters of HtT within the following grid: minimal occurrence $k \in \{2, 3\}$, minimal accuracy $p \in \{0.3, 0.5, 0.7\}$. Due to the cost of LtM (3-5$\times$ compared to CoT), we induce rules and select the best hyperparameters based on CoT prompting, and only use LtM prompting for the deduction stage. This might slightly underestimate the performance of HtT for LtM prompting.

4.2 Numerical Reasoning

We use the Arithmetic dataset (Wu et al., 2023) to evaluate the LLMs on numerical reasoning in non-decimal systems. This dataset contains summation problems over 2 to 4 digits in several base systems. Since the rules in a non-decimal system are mostly different from those in the decimal system, arithmetic is considered to be a counterfactual setting that requires an LLM to perform reasoning rather than reciting. To prepare the dataset for HtT, we split it into training, validation and test. The training set contains 900 examples of 2 digit addition. Both the validation and test sets contain 100 examples of 2, 3 and 4 digit addition. To collect rules in the induction stage, we extract all equations that consist of summations over numbers or capitalized letters (e.g. A to F in base-16).

Table 1 shows the results on the base-16, base-11 and base-9 datasets from Arithmetic. 0-shot CoT results in the worst performance for both LLMs in all base systems. The main issue for 0-shot CoT is that the LLMs tend to convert non-decimal inputs to decimal, perform the calculation in decimal, then convert back to non-decimal, which is error prone due to the additional multiplications and divisions. For the few-shot prompting methods, while previous works (Zhang et al., 2023b) observe

Table 2: Results on the symbolic version of CLUTRR.

Model	Prompt	2 hops	3 hops	4 hops	5 hops	6 hops	7 hops	8 hops	9 hops	10 hops	Average
EdgeTransformer		100.0	94.4	96.8	88.0	68.8	61.9	50.0	50.0	36.0	71.8
GPT3.5	0-shot CoT	50.0	22.2	12.9	8.0	12.5	9.5	10.0	3.8	4.0	14.8
	5-shot CoT	0.0	27.8	45.2	36.0	18.8	19.0	16.7	11.5	16.0	21.2
	+ HtT	87.5	38.9	35.5	44.0	37.5	14.3	33.3	11.5	36.0	**37.6 (+16.4)**
	+ HtT (GPT4)	100.0	55.6	32.3	60.0	50.0	47.6	43.3	19.2	28.0	**48.4 (+27.2)**
	5-shot LtM	37.5	22.2	29.0	36.0	25.0	14.3	10.0	23.1	20.0	24.1
	+ HtT	100.0	33.3	32.3	48.0	31.3	33.3	23.3	34.6	28.0	**40.5 (+16.4)**
	+ HtT (GPT4)	75	44.4	41.9	52.0	37.5	33.3	23.3	19.2	16.0	**38.1 (+14.0)**
GPT4	0-shot CoT	50.0	22.2	22.6	32.0	37.5	38.1	33.3	46.2	16.0	33.1
	5-shot CoT	50.0	55.6	71.0	80.0	50.0	52.4	30.0	46.2	20.0	50.6
	+ HtT	100.0	61.1	74.2	84.0	75.0	38.1	56.7	53.8	36.0	**64.3 (+13.7)**
	5-shot LtM	62.5	38.9	58.1	68.0	50.0	38.1	43.3	34.6	28.0	46.8
	+ HtT	100.0	55.6	77.4	80.0	75.0	38.1	36.7	38.5	20.0	**57.9 (+11.1)**

that separating prompts can result in less hallucination, we do not observe any gain for LtM over CoT. This is because non-decimal arithmetic is so counterfactual that even models like GPT4 cannot always produce the correct rule in a separate prompt.

For both CoT and LtM prompting, HtT consistently improves the accuracy of both models. Notably, HtT improves CoT on GPT4 by a large margin of 21.0% in average accuracy. The performance gain is less significant for GPT3.5, since it is worse at inducing correct rules and performing retrieval in the deduction stage. One potential solution is to use a strong model to induce the rules, and use a weak model to perform deduction. To test this set up, we additionally conduct experiments on GPT3.5 with rules induced by GPT4. We observe a large improvement on LtM + HtT with better rules from GPT4, especially on base-11 and base-9 where GPT3.5 struggles to induce correct rules due to poor base performance. There is no improvement for CoT + HtT using the better rules, because GPT3.5 has a strong tendency to rely on its own beliefs (i.e. mostly decimal rules) rather than retrieve from the knowledge block with CoT, similar to the observation in Longpre et al. (2021).

4.3 Relational Reasoning

The next task we consider is the relational reasoning task commonly studied in the knowledge graph community. We use CLUTRR (Sinha et al., 2019) for this purpose. Each example in CLUTRR consists of a chain of kinship relations between family members, where the goal is to infer the relationship between the head and the tail entities in the chain. CLUTRR comes in two forms: a symbolic version that only contains entities and relations, and a textual version that additionally contains irrelevant context. We evaluate HtT on both the symbolic and textual versions and report the results respectively. We use a subset of the standard splits from Sinha et al. (2019). We use 2,000 samples of 2 and 3 hop examples for training, and 200 samples of 2 to 10 hop examples for both validation and test. For reference, we reproduce EdgeTransformer (Bergen et al., 2021), one of the best domain-specific models on CLUTRR, in the same setting.

Table 2 compares different methods on CLUTRR. It is observed that 0-shot CoT performs the worst with both GPT3.5 and GPT4. For the few-shot prompting methods, performance is similar between CoT and LtM. Here HtT consistently improves both the prompting methods with both models by a margin of 11.1-27.2% in average accuracy. Notably, GPT3.5 does not fall short in retrieving rules on CLUTRR and benefits more from HtT than GPT4, possibly because there are fewer rules in CLUTRR than in Arithmetic. It worth mentioning that using rules from GPT4, the performance of CoT on GPT3.5 increases by 27.2%, which is more than double performance of CoT and close to the performance of CoT on GPT4. Hence, we believe HtT may serve as a new form of knowledge distillation from strong to weak LLMs.

Compared to the supervised baseline EdgeTransformer, the best model, 5-shot CoT + HtT on GPT4, is 7.5% lower in average accuracy. We note that this is a reasonable result, given that EdgeTransformer leverages forward chaining as a strong inductive bias and is specific to this task. There are two advantages that HtT has over such domain-specific models: (1) HtT does not require a predefined relation vocabulary, while domain-specific models require such a vocabulary before training. See Appendix F for the learned rule library that contains relations not present in the exemplars. (2)

Table 3: Results on the textual CLUTRR with rules learned on the symbolic CLUTRR.

Model	Prompt	Accuracy
GPT3.5	5-shot CoT	16.0
	+ HtT	16.3 (+0.3)
GPT4	5-shot CoT	48.7
	+ HtT	**59.1 (+10.4)**

Table 4: Ablation studies on random rules and the XML tagging trick. Arithmetic has 3 levels of hierarchy, while CLUTRR has 2.

Prompt	Arithmetic	CLUTRR
5-shot CoT	57.4	50.6
5-shot CoT (tags)	63.7 (+6.3)	43.7 (-6.9)
+ random rules	23.7 (-33.7)	9.9 (-40.7)
+ HtT (unsorted)	67.2 (+9.8)	57.1 (+6.5)
+ HtT (sorted)	72.5 (+15.1)	60.0 (+9.4)
+ HtT (1 tag)	74.8 (+17.4)	59.6 (+9.0)
+ HtT (2 tags)	76.6 (+19.2)	**64.3 (+13.7)**
+ HtT (3 tags)	**78.4 (+21.0)**	N/A

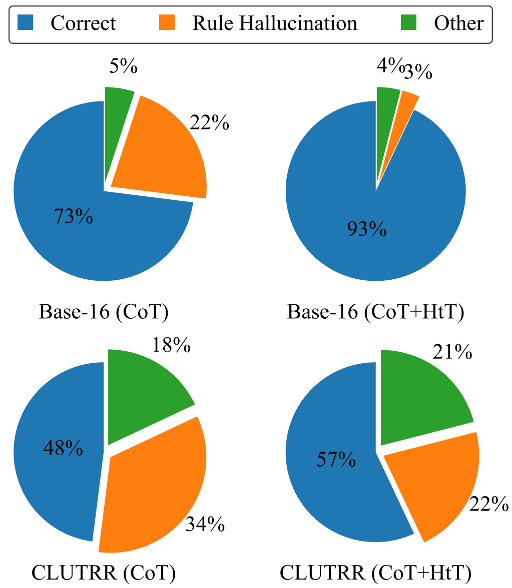

Figure 3: Statistics of different error cases.

The rules learned by HtT can directly transfer to textual inputs, while one needs to train different domain-specific models for symbolic and textual inputs. To verify this, we tackle the textual version of CLUTRR with rules learned from the symbolic version. Table 3 shows that HtT significantly improves the performance of GPT4 on the textual version. The improvement is not significant for GPT3.5, since it often produces errors other than rule hallucination when processing textual input.

4.4 Ablation Studies

We conduct ablation studies to better understand how HtT discovers good rules in the induction stage and how the learned rule library improves deductive reasoning. We perform these experiments with GPT4, since GPT3.5 sometimes struggles with inducing and retrieving rules. Due to space limitations, we report the full results of the ablation studies in Appendix B.

Does HtT reduce the hallucination of rules? Since an LLM generates free-form text to solve a problem, there can be multiple reasons for failure (Zheng et al., 2023). While HtT boosts the overall performance on reasoning tasks, it is not clear whether the gain comes from reduced hallucination or other improvements. Here we manually analyze the predictions of CoT and CoT + HtT on 100 test examples from Arithmetic (base-16) and CLUTRR, and classify the predictions into 3 categories: correct, rule hallucination and other. Note that there are two types of errors related to rules: (1) the LLM generates or retrieves a factually incorrect rule. (2) the LLM retrieves a factually correct rule that is not proper for the deductive step. We consider the former as hallucination and the latter as retrieval errors. Details of the categorization can be found in Appendix B. Figure 3 shows the distribution of error cases. We can see that HtT significantly reduces hallucination versus CoT.

Do the learned rules just hint the model about the rule space? A previous study (Min et al., 2022) found that random labels perform similarly to gold labels in in-context learning. If that was the case for our tasks, we could just generate random rules and does not resort to HtT to learn the rule library. To investigate this question, we consider replacing the conclusions in the learned rules with random answers. For example, we replace 5 + A = E with 5 + A = 7 in the base-16 dataset. We consider a variant of HtT with the same number of random rules as in the rule library. Table 4 shows that random rules significantly hurt performance, indicating the necessity of learned rules in HtT. We conjecture that the reason for the contrary observation is that Min et al. (2022) studied simple classification problems, whereas we are dealing with multi-step reasoning problems.

How do XML tags improve deductive reasoning? In Section 3.2, we introduce the XML tagging trick to augment the retrieval ability of an LLM. Here we want to verify: (1) XML tagging improves

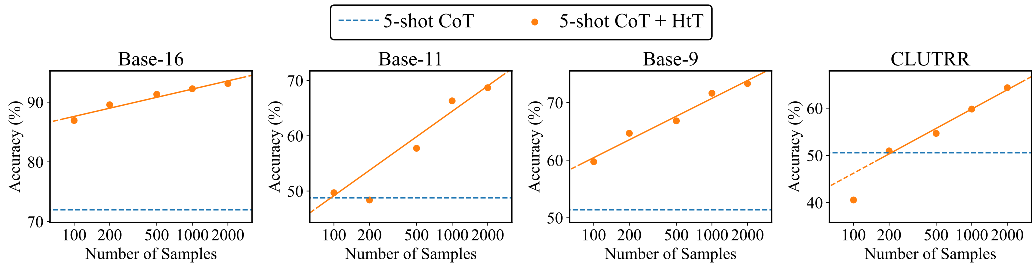

Figure 4: Performance of HtT w.r.t. the number of samples in the induction stage.

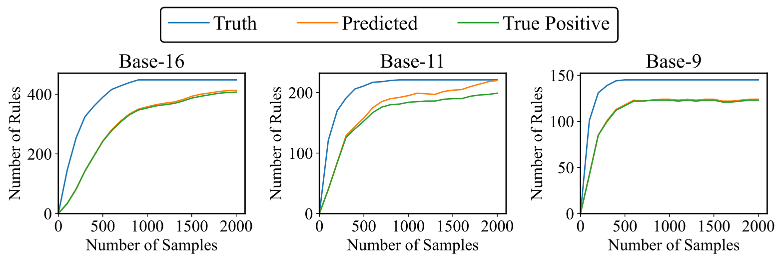

Figure 5: Number of rules discovered by HtT in the induction stage.

the retrieval ability of the LLM. (2) XML tagging performs better with a deeper hierarchy. For (1), we apply the same XML tagging trick to CoT except that we do not provide the rule library. For (2), we examine with different levels of hierarchies in XML tagging, and consider a variant with unsorted (i.e. randomly ordered) rules without any hierarchy. As shown in Table 4, the XML tagging trick only moderately improves the performance of CoT on Arithmetic and even hurts on CLUTRR, indicating that XML tagging mostly improves the retrieval ability rather than simply elicits rules from the parameteric knowledge of the LLM. With sorted rules and deeper hierarchies, the XML tagging trick doubles the performance gain on both Arithmetic and CLUTRR datasets.

How many samples does HtT need for the induction stage? For the experiments in Table 1 and 2, we use 2,000 examples for the induction stage. One may be curious about how HtT scales with the number of samples and what is the minimal number of examples required. Here we conduct experiments with different numbers of examples for the induction stage. As shown in Figure 4, there is a log-linear trend between performance and the number of examples, consistent with the scaling law for supervised learning (Kaplan et al., 2020). The minimal number of examples varies across datasets. On base-16 and base-9, 100 examples are sufficient for CoT + HtT to outperform CoT. On base-11 and CLUTRR, at least 500 examples are required to obtain a significant gain.

What proportion of rules are discovered by HtT? Since we use the CoT prompt to generate rules and verify them in the induction stage, it is likely that an LLM fails to discover some rules due to their imperfect nature. Here we are interested in what proportion of rules can be discovered by HtT. To answer this question, we compare HtT with an oracle that always induces all necessary rules from an example. Note this comparison can only be made on Arithmetic, since rules in CLUTRR are probabilistic rather than deterministic, e.g. a grandmother's daughter can be either a mother or an aunt. Figure 5 shows the number of rules induced by the oracle and HtT, as well as the number of true positive rules in HtT. We can see that HtT discovers more than 85% of the rules in all datasets. Interestingly, the number of rules grows on base-16 and base-11, but not on base-9, which indicates some rules in base-9 cannot be discovered even by GPT4.

5 RELATED WORK

Reasoning over Natural Language. Solving reasoning problems in natural language can be traced back to bAbI (Weston et al., 2016), which consists of many proxy tasks that evaluate inductive, deductive and other forms of reasoning. Early attempts designed models with memory components to scan the input and generate the answer (Weston et al., 2015; Kumar et al., 2016). With the rise of Transformers (Vaswani et al., 2017) and pretrained language models (Radford et al., 2018; **?**;

Raffel et al., 2020), several works have demonstrated that transformers can be finetuned on specific datasets to acquire various abilities, including deductive reasoning (Clark et al., 2020), reasoning with implicit knowledge (Talmor et al., 2020), and program execution (Nye et al., 2021).

The success of in-context learning (Brown et al., 2020) and instruction tuning (Wei et al., 2022a; Sanh et al., 2022) has motivated significant work on prompting LLMs to solve reasoning problems. Some notable achievements include chain-of-thought (CoT) prompting (Wei et al., 2022c) that elicits reasoning with intermediate steps in few-shot exemplars, least-to-most (LtM) prompting (Zhou et al., 2023) and decomposed prompting (Khot et al., 2023) that decompose multi-step reasoning into sub problems and invoke separate prompts for each sub problem. Recent works have extended CoT with planning (Yao et al., 2023), multi-agent debate (Wang et al., 2023b), and verifiers (Lightman et al., 2023). All these works rely on parameteric knowledge stored in the LLM's weights (Petroni et al., 2019) and do not explicitly extract rules. Another line of work (Creswell et al., 2023; Kazemi et al., 2023) considers taking explicit facts and rules as input, and searching possible proof traces that lead to the answer. Compared to existing works, HtT is the first to induce rules from examples and apply them deductively to solve reasoning problems.

Program Synthesis and Neural-Symbolic Reasoning. Prior to the era of LLMs, neural-symbolic techniques have demonstrated significant performance gains in symbolic problems such as arithmetic (Reed & De Freitas, 2016; Cai et al., 2017) and grammar learning (Chen et al., 2018; 2020; Nye et al., 2020). With LLMs, some recent work has investigated in-context learning to simulate multi-step computation for arithmetic reasoning (Zhou et al., 2022). Tool-augmented language models (Parisi et al., 2022; Schick et al., 2023) teach LLMs to generate API calls to external tools (e.g. calculator, search engine), which has been extended to the composition of tools (Gupta & Kembhavi, 2023; Lu et al., 2023) for reasoning problems. The most related work in the area of program synthesis is library learning (Ellis et al., 2021), which aims to learn a set of reusable functions that generalize to new examples. Recent works have applied library learning to LLMs to solve reasoning problems (Cai et al., 2023) and play Minecraft (Wang et al., 2023a). HtT shares a similar spirit to these works. However, HtT learns rules in natural language and does not require a symbolic program executor, increasing applicability compared to program synthesis methods.

Rule Learning. The proposed method is also related to rule learning techniques. Classical rule mining algorithms (Agrawal et al., 1994; Galárraga et al., 2013) extract rules that have a large support set and high confidence. HtT follows the same spirit and filters rules based on their frequency and accuracy. In recent years, rule learning methods have mostly been studied in knowledge graphs (Yang et al., 2017; Qu et al., 2021; Lu et al., 2022) or theorem proving (Rocktäschel & Riedel, 2017; Minervini et al., 2020). These methods start from predefined rule templates and learn rules that best fit the observations, with an emphasis on scalability and compositionality in knowledge graphs and theorem proving respectively. However, none of these methods can directly perform reasoning in natural language. While some works (Yang & Deng, 2021; Zhang et al., 2023a) can be applied to natural language, they need dedicated solvers for the learning and inference of rules. By contrast, HtT induces and applies rules by prompting an LLM, and is applicable to any generative LLM.

6 Discussion and Conclusion

Limitations. One limitation of HtT is that it requires the base model to have reasonably strong knowledge and retrieval ability. As shown in Table 1, the gain of HtT for GPT3.5 is very marginal due to weak knowledge of non-decimal systems. Even with a rule library induced by GPT4, GPT3.5 has issues in retrieving correct rules, especially in very counterfactual settings like base-11 and base-9. We hypothesize that such issues may be solved by finetuning on retrieval datasets. Another limitation is that the number of rules is limited by the LLM's context length. It remains an open problem to scale up deductive reasoning when the rule library cannot fit into the LLM's input context.

Conclusion. In this paper, we introduce Hypotheses-to-Theories (HtT) to learn explicit rules and apply them in reasoning problems. HtT consists of two stages: an induction stage that generates rules and verifies them to construct a library, and a deduction stage that applies rules from the learned rule library to solve a reasoning problem. Our empirical analysis shows that HtT significantly boosts the performance of baseline prompting methods on numerical reasoning and relational reasoning problems. The proposed method opens up a new direction of learning textual knowledge with LLMs. We expect HtT to facilitate various applications and future research of LLMs.

Ethics Statement

The goal of this paper is to learn rules from training samples and apply them to solve reasoning problems with LLMs. Despite the strong abilities presented in this paper, we should be aware that such abilities can also be potentially harmful. First, some malicious uses of LLMs may be augmented by HtT. Examples include collecting personal data, social engineering and abuse of legal process. Second, since HtT uses LLMs to generate knowledge, it is inevitable that the knowledge may be permeated by biases in existing models, such as genders, races or religions. Regarding these ethical problems, future works may better align LLMs with human values and reduce biases in their parametric knowledge.

Reproducibility Statement

All the experiments are based on publicly available datasets. We describe the experiment setup and hyperparameters in Section 4.1 and Appendix A. Full results of ablation study are presented in Appendix B. Examples of prompts for each method and each dataset can be found in Appendix E. We also attach the source code in the supplementary materials. Due to frequent updates of GPT3.5 and GPT4, we cannot guarantee the exact numbers can be reproduced with future models, but one should be able to observe the gain of HtT over basic prompting methods.

References

Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Very Large Data Bases*, volume 1215, pp. 487–499, 1994.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Francis Bacon. *Novum Organum*. 1620.

Leon Bergen, Timothy O'Donnell, and Dzmitry Bahdanau. Systematic generalization with edge transformers. In *Advances in Neural Information Processing Systems*, volume 34, pp. 1390–1402, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Jonathon Cai, Richard Shin, and Dawn Song. Making neural programming architectures generalize via recursion. In *International Conference on Learning Representations*, 2017.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. *arXiv preprint arXiv:2305.17126*, 2023.

Xinyun Chen, Chang Liu, and Dawn Song. Towards synthesizing complex programs from input-output examples. In *International Conference on Learning Representations*, 2018.

Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. Compositional generalization via neural-symbolic stack machines. In *Advances in Neural Information Processing Systems*, pp. 1690–1701, 2020.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. In *International Joint Conferences on Artificial Intelligence*, pp. 3882–3890, 2020.

Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *International Conference on Learning Representations*, 2023.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Programming Language Design and Implementation*, pp. 835–850, 2021.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *International Conference on World Wide Web*, pp. 413–422, 2013.

Mikhail Galkin, Zhaocheng Zhu, Hongyu Ren, and Jian Tang. Inductive logical query answering in knowledge graphs. In *Advances in Neural Information Processing Systems*, volume 35, pp. 15230–15243, 2022.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Computer Vision and Pattern Recognition*, pp. 14953–14962, 2023.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. In *ACM Computing Surveys*, volume 55, pp. 1–38. ACM New York, NY, 2023.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*, pp. 15696–15707. PMLR, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Empirical Methods in Natural Language Processing*, 2020.

Seyed Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. LAMBADA: Backward chaining for automated reasoning in natural language. In *Transactions of the Association for Computational Linguistics*, pp. 6547–6568, 2023.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. In *International Conference on Learning Representations*, 2023.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213, 2022.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pp. 1378–1387. PMLR, 2016.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering. In *Empirical Methods in Natural Language Processing*, pp. 7052–7063, 2021.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models. In *Advances in Neural Information Processing Systems*, 2023.

Shengyao Lu, Bang Liu, Keith G Mills, Shangling Jui, and Di Niu. R5: Rule discovery with reinforced and recurrent relational reasoning. In *International Conference on Learning Representations*, 2022.

Nick McKenna, Tianyi Li, Liang Cheng, Mohammad Javad Hosseini, Mark Johnson, and Mark Steedman. Sources of hallucination by large language models on inference tasks. *arXiv preprint arXiv:2305.14552*, 2023.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Empirical Methods in Natural Language Processing*, 2022.

Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, Edward Grefenstette, and Tim Rocktäschel. Learning reasoning strategies in end-to-end differentiable proving. In *International Conference on Machine Learning*, pp. 6938–6949. PMLR, 2020.

Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. In *The Journal of Logic Programming*, volume 19, pp. 629–679. Elsevier, 1994.

Maxwell Nye, Armando Solar-Lezama, Josh Tenenbaum, and Brenden M Lake. Learning compositional rules via neural program synthesis. In *Advances in Neural Information Processing Systems*, volume 33, pp. 10832–10842, 2020.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

OpenAI. Gpt-4 technical report. 2023.

Aaron Parisi, Yao Zhao, and Noah Fiedel. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*, 2022.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Empirical Methods in Natural Language Processing*, pp. 2463–2473, 2019.

Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. In *International Conference on Learning Representations*, 2021.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. In *The Journal of Machine Learning Research*, volume 21, pp. 5485–5551, 2020.

Scott Reed and Nando De Freitas. Neural programmer-interpreters. In *International Conference on Learning Representations*, 2016.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *Empirical Methods in Natural Language Processing*, pp. 5418–5426, 2020.

Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems*, volume 30, pp. 3791–3803, 2017.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L Hamilton. Clutrr: A diagnostic benchmark for inductive reasoning from text. In *Empirical Methods of Natural Language Processing*, pp. 4506–4515, 2019.

Russell Stuart and Norvig Peter. Artificial intelligence-a modern approach 3rd ed, 2016.

Alon Talmor, Oyvind Tafjord, Peter Clark, Yoav Goldberg, and Jonathan Berant. Leap-of-thought: Teaching pre-trained models to systematically reason over implicit knowledge. In *Advances in Neural Information Processing Systems*, volume 33, pp. 20227–20237, 2020.

Xiaojuan Tang, Zilong Zheng, Jiaqi Li, Fanxu Meng, Song-Chun Zhu, Yitao Liang, and Muhan Zhang. Large language models are in-context semantic reasoners rather than symbolic reasoners. *arXiv preprint arXiv:2305.14825*, 2023.

Komal Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, pp. 9448–9457. PMLR, 2020.

Karthik Valmeekam, Sarath Sreedharan, Matthew Marquez, Alberto Olmo, and Subbarao Kambhampati. On the planning abilities of large language models (a critical investigation with a proposed benchmark). *arXiv preprint arXiv:2302.06706*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, volume 30, pp. 6000–6010, 2017.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. Kepler: A unified model for knowledge embedding and pre-trained language representation. In *Transactions of the Association for Computational Linguistics*, volume 9, pp. 176–194. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2021.

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. Unleashing cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. *arXiv preprint arXiv:2307.05300*, 2023b.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022a.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. In *Transactions on Machine Learning Research*, 2022b.

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

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *International Conference on Learning Representations*, 2015.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart Van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. In *International Conference on Learning Representations*, 2016.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. *arXiv preprint arXiv:2307.02477*, 2023.

Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*, volume 30, pp. 2316–2325, 2017.

Kaiyu Yang and Jia Deng. Learning symbolic rules for reasoning in quasi-natural language. *arXiv preprint arXiv:2111.12038*, 2021.

Zonglin Yang, Li Dong, Xinya Du, Hao Cheng, Erik Cambria, Xiaodong Liu, Jianfeng Gao, and Furu Wei. Language models as inductive reasoners. *arXiv preprint arXiv:2212.10923*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Hanlin Zhang, Jiani Huang, Ziyang Li, Mayur Naik, and Eric Xing. Improved logical reasoning of language models via differentiable symbolic programming. *arXiv preprint arXiv:2305.03742*, 2023a.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A Smith. How language model hallucinations can snowball. *arXiv preprint arXiv:2305.13534*, 2023b.

Shen Zheng, Jie Huang, and Kevin Chen-Chuan Chang. Why does chatgpt fall short in answering questions faithfully? *arXiv preprint arXiv:2304.10513*, 2023.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*, 2023.

Hattie Zhou, Azade Nova, Hugo Larochelle, Aaron Courville, Behnam Neyshabur, and Hanie Sedghi. Teaching algorithmic reasoning via in-context learning. *arXiv preprint arXiv:2211.09066*, 2022.

A EXPERIMENT SETUP

Here we present further details of our experiment setup. Since LLMs output free-form text to solve the problems, we evaluate models by matching the predicted text and the ground truth answer. We crop the last sentence from the predicted text, and check if the ground truth answer is present in that sentence. We only consider full word match and exclude partial matches like "mother" and "grandmother". If the LLM outputs more than one answer, we always consider it as wrong.

For HtT, we extract rules in the induction stage by searching string templates with regular expressions. We note that HtT does not rely on engineering of the string template, and any templates that can extract rules from the given few-shot examples suffice here. Even if the string templates recall wrong rules, they can be easily filtered by our minimal occurrence criterion. We construct the rule library by filtering learned rules based on their minimal occurrence k and minimal accuracy p. Table 5 lists the best hyperparameter configurations of HtT on the Arithmetic and CLUTRR datasets.

Table 5: Hyperparameter configurations of HtT on different datasets.

Model	**Hyperparameter**	**Arithmetic**			**CLUTRR**
		Base-16	**Base-11**	**Base-9**	
GPT3.5	minimal occurrence k	2	2	2	2
	minimal accuracy p	0.3	0.5	0.5	0.3
GPT4	minimal occurrence k	2	2	2	2
	minimal accuracy p	0.5	0.3	0.3	0.7

B FULL EXPERIMENTAL RESULTS

Table 6 provides the full results on the textual version of CLUTRR. Table 7 and 8 show the full results for the ablation studies in Section 4.4.

Table 6: Results on the textual CLUTRR with rules learned on the symbolic CLUTRR.

Model	**Prompt**	**2 hops**	**3 hops**	**4 hops**	**5 hops**	**6 hops**	**7 hops**	**8 hops**	**9 hops**	**10 hops**	**Average**
GPT3.5	5-shot CoT	12.5	11.1	12.9	36.0	6.3	19.0	30.0	0.0	16.0	16.0
	+ HtT	12.5	22.2	12.9	32.0	25.0	23.8	6.7	3.8	8.0	16.3 (+0.3)
GPT4	5-shot CoT	50.0	50.0	71.0	68.0	50.0	47.6	26.7	34.6	40.0	48.7
	+ HtT	100.0	55.6	77.4	72.0	75.0	38.1	23.3	42.3	48.0	**59.1 (+10.4)**

Table 7: Ablation studies on Arithmetic. All the entries are based on GPT4.

Prompt	**Base-16**			**Base-11**			**Base-9**			**Average**
	2 digits	**3 digits**	**4 digits**	**2 digits**	**3 digits**	**4 digits**	**2 digits**	**3 digits**	**4 digits**	
5-shot CoT	83.3	71.1	61.5	52.8	47.4	46.2	75.0	36.8	42.3	57.4
5-shot CoT (tags)	100.0	81.6	84.6	61.1	44.7	23.076	80.6	55.3	42.3	63.7 (+6.3)
+ random rules	19.4	18.4	0.0	30.6	28.9	15.4	44.4	21.1	34.6	23.7 (-33.7)
+ HtT (unsorted)	91.7	81.6	65.4	80.6	52.6	57.7	77.8	55.3	42.3	67.2 (+9.8)
+ HtT (sorted)	97.2	89.5	76.9	80.6	60.5	65.4	86.1	50.0	46.2	72.5 (+15.1)
+ HtT (1 tag)	91.7	89.5	84.6	80.6	60.5	46.2	88.9	65.8	65.4	74.8 (+17.4)
+ HtT (2 tags)	91.7	84.2	96.2	88.9	63.2	46.2	94.4	63.2	61.5	76.6 (+19.2)
+ HtT (3 tags)	100.0	94.7	84.6	88.9	71.1	46.2	86.1	68.4	65.4	**78.4 (+21.0)**

We manually analyze the predictions of CoT and CoT + HtT on 100 test examples from Arithmetic (base-16) and CLUTRR. We classify the predictions into five categories

- Correct: correct examples.
- Rule hallucination (only): error examples solely due to factually incorrect rules.
- Rule hallucination (and other): error examples due to both incorrect rules and other reasons.
- Retrieval: error examples solely due to the retrieval of undesired, but factually correct rules.

Table 8: Ablation studies on CLUTRR. All the entries are based on GPT4.

Prompt	2 hops	3 hops	4 hops	5 hops	6 hops	7 hops	8 hops	9 hops	10 hops	Average
5-shot CoT	50.0	55.6	71.0	80.0	50.0	52.4	30.0	46.2	20.0	50.6
5-shot CoT (tags)	37.5	38.9	74.2	76.0	50.0	47.6	26.7	38.5	4.0	43.7 (-6.9)
+ random rules	0.0	16.7	3.2	20.0	6.3	23.8	3.3	11.5	4.0	9.9 (-40.7)
+ HtT (unsorted)	87.5	50.0	58.1	76.0	56.3	52.4	43.3	46.2	44.0	57.1 (+6.5)
+ HtT (sorted)	100.0	61.1	77.4	72.0	62.5	42.9	53.3	38.5	32.0	60.0 (+9.4)
+ HtT (1 tag)	100.0	61.1	74.2	76.0	62.5	42.9	53.3	30.8	36.0	59.6 (+9.0)
+ HtT (2 tags)	100.0	61.1	74.2	84.0	75.0	38.1	56.7	53.8	36.0	**64.3 (+13.7)**

- Non-retrieval: error examples that do not fit into any of the above categories.

In the main paper, we aggregate the five categories into three coarse categories: correct, rule hallucination and others. Table 9 demonstrates the full results of error cases on base-16 and CLUTRR. It is observed that HtT significantly reduced rule hallucination versus CoT. On CLUTRR, HtT has a moderate ratio of errors in the retrieval category. This is because either the required rule is missing in the library, or the LLM cannot retrieve the required rule. We believe such an issue can be mitigated by learning a more complete rule library and finetuning the LLM on retrieval problems.

Table 9: Statistics of error cases.

Dataset	Prompt	Correct	Rule Hallucination		Others	
			Only	And Others	Retrieval	Non-Retrieval
Base-16	CoT	73%	19%	3%	N/A	5%
	CoT+HtT	93%	2%	1%	0%	4%
CLUTRR	CoT	48%	27%	7%	N/A	18%
	CoT+HtT	57%	12%	10%	14%	4%

C Learning First-Order Logic Rules on CLUTRR

While we apply HtT to learn numerical rules and propositional logic rules in the paper, HtT is not limited to certain types of rules. Here we show that HtT can be used to learn first-order logic rules on CLUTRR, e.g. if Y is X's father and Z is Y's sister, Z is X's aunt. We modify CoT and CoT + HtT to leverage first-order logic rules in deduction steps. Table 10 shows the results of first-order logic rules on CLUTRR. It is observed that first-order logic rules significantly improve performance over propositional logic rules on CoT, and HtT further improves over CoT with the learned rule library. Nevertheless, we note that CLUTRR is a simple dataset and does not require unification. In general, reasoning with first-order logic rules involves unification steps (i.e. to match rules and facts at each step), which require planning abilities that LLMs typically do not possess Valmeekam et al. (2023).

Table 10: Results of different rule types on CLUTRR. All entries are based on GPT-4.

Rule Type	Prompt	2 hops	3 hops	4 hops	5 hops	6 hops	7 hops	8 hops	9 hops	10 hops	Average
Propositional	5-shot CoT	50.0	55.6	71.0	80.0	50.0	52.4	30.0	46.2	20.0	50.6
	+ HtT	100.0	61.1	74.2	84.0	75.0	38.1	56.7	53.8	36.0	**64.3 (+13.7)**
First-order	5-shot CoT	75.0	61.1	87.1	100.0	87.5	61.9	93.3	76.9	68.0	79.0
	+ HtT	100.0	61.1	93.5	100.0	93.8	76.2	86.7	88.5	72.0	**85.7 (+6.7)**

D Summarize Rule Libraries into High-Level Knowledge

In HtT, we learn rules as independent atoms, which often results in a very large rule library. We notice that in some tasks, rules may be highly correlated and there might be a way to describe the rule library in a high-level form. For example, one may describe summation rules by introducing the definition of digits, summation and carry. Inspired by this idea, we prompt GPT-4 to summarize the rule libraries learned on Arithmetic into high-level knowledge, and replace the rule library with the summary in the deduction stage. Due to the large variation in the generated summaries, we sample 5 summaries for each base system, and report the mean and variance of the performance.

Table 11 shows the performance of the original HtT and HtT with summarized knowledge. Compared to CoT, the summaries improve the performance on base-11 and base-9, yet remain the same performance on base-16. The gain of summaries is generally smaller than the gain of rule libraries, which is expected as a summary usually loses some details of a rule library. It is worth mentioning that summaries require significantly shorter prompts and are more practical for applications. On base-11 where summaries are only 3.1% worse than rule libraries, summaries reduce the number of tokens in HtT from 3,997 to 1,510, only 60% more tokens compared to the standard CoT, as shown in Table 12. However, we notice that there is a large variance in the performance of summaries. We leave the problem of generating better and more stable summaries as future work.

Table 11: Results of HtT with rule libraries and summarized knowledge on Arithmetic. All the entries are based on GPT-4.

Prompt	**Base-16**			**Base-11**			**Base-9**			**Average**
	2 digits	**3 digits**	**4 digits**	**2 digits**	**3 digits**	**4 digits**	**2 digits**	**3 digits**	**4 digits**	
5-shot CoT	83.3	71.1	61.5	52.8	47.4	46.2	75.0	36.8	42.3	57.4
+ HtT (rules)	100.0	94.7	84.6	88.9	71.1	46.2	86.1	68.4	65.4	**78.4 (+21.0)**
+ HtT (summary)	$86.7^{\pm 7.7}$	$70.5^{\pm 8.1}$	$63.8^{\pm 4.6}$	$79.4^{\pm 3.8}$	$60.5^{\pm 15.0}$	$56.9^{\pm 5.1}$	$85.6^{\pm 5.7}$	$63.2^{\pm 3.3}$	$53.1^{\pm 5.1}$	$68.9^{\pm 2.7}$ (+11.5)

Table 12: Number of tokens used by different prompts on base-11.

Prompt	**Knowledge**	**Exemplars**	**Question**	**Total**
5-shot CoT	0	945	17	962
+ HtT (rules)	2,959	1,021	17	3,997
+ HtT (summary)	472	1,021	17	1,510

E Prompts

For all HtT prompts, rules in the prompt are just for demonstration purpose. We do not use any rules from human prior knowledge in the rule library. All the rules in the rule library are generated by the LLM in the induction stage.

E.1 Numerical Reasoning

Prompt 1-5 show the prompts we use for different prompting methods on the base-16 dataset. For base-11 and base-9, we use the same prompts except we change the exemplars in the corresponding few-shot prompts.

Prompt 1: 0-shot CoT prompt for Arithmetic.

Question: In base-{{ base }}, what is {{ x }} + {{ y }}?
Answer: Let's think step by step.

Prompt 2: 5-shot CoT prompt for Arithmetic.

Question: In base-16, what is EC + DD?
Answer:
EC is E, C. DD is D, D. So the steps are C + D, E + D.
There is no carry. C + D = 19. 19 is 1, 9. So we set the carry to 1. Prepend 9 to the answer. So far the answer has 1 digit: 9.
The carry is 1. E + D + 1 = 1C. 1C is 1, C. So we set the carry to 1. Prepend C to the answer. So far the answer has 2 digits: C, 9.

The carry is 1. Prepend 1 to the answer. So far the answer has 3 digits: 1, C, 9.
Therefore, the answer is 1C9.

Question: In base-16, what is 18 + 9F?
Answer:
18 is 1, 8. 9F is 9, F. So the steps are 8 + F, 1 + 9.
There is no carry. 8 + F = 17. 17 is 1, 7. So we set the carry to 1. Prepend 7 to the answer. So far the answer has 1 digit: 7.
The carry is 1. 1 + 9 + 1 = B. B is 0, B. So we clear the carry. Prepend B to the answer. So far the answer has 2 digits: B, 7.
There is no carry. So far the answer has 2 digits: B, 7.
Therefore, the answer is B7.

Question: In base-16, what is 79 + 8B?
Answer:
79 is 7, 9. 8B is 8, B. So the steps are 9 + B, 7 + 8.
There is no carry. 9 + B = 14. 14 is 1, 4. So we set the carry to 1. Prepend 4 to the answer. So far the answer has 1 digit: 4.
The carry is 1. 7 + 8 + 1 = 10. 10 is 1, 0. So we set the carry to 1. Prepend 0 to the answer. So far the answer has 2 digits: 0, 4.
The carry is 1. Prepend 1 to the answer. So far the answer has 3 digits: 1, 0, 4.
Therefore, the answer is 104.

Question: In base-16, what is A6 + 94?
Answer:
A6 is A, 6. 94 is 9, 4. So the steps are 6 + 4, A + 9.
There is no carry. 6 + 4 = A. A is 0, A. So we clear the carry. Prepend A to the answer. So far the answer has 1 digit: A.
There is no carry. A + 9 = 13. 13 is 1, 3. So we set the carry to 1. Prepend 3 to the answer. So far the answer has 2 digits: 3, A.
The carry is 1. Prepend 1 to the answer. So far the answer has 3 digits: 1, 3, A.
Therefore, the answer is 13A.

Question: In base-16, what is 54 + D3?
Answer:
54 is 5, 4. D3 is D, 3. So the steps are 4 + 3, 5 + D.
There is no carry. 4 + 3 = 7. 7 is 0, 7. So we clear the carry. Prepend 7 to the answer. So far the answer has 1 digit: 7.
There is no carry. 5 + D = 12. 12 is 1, 2. So we set the carry to 1. Prepend 2 to the answer. So far the answer has 2 digits: 2, 7.
The carry is 1. Prepend 1 to the answer. So far the answer has 3 digits: 1, 2, 7.
Therefore, the answer is 127.

Question: In base-16, what is {{ x }} + {{ y }}?
Answer:

Prompt 3: 5-shot CoT+HtT prompt for Arithmetic.

Instruction: When you answer the questions, try to use the provided knowledge whenever possible. Try not to invent knowledge by yourself unless necessary.
Knowledge:
<no_carry>
<0><0>0 + 0 = 0.</0>... <F>0 + F = F.</F></0>
...
<F><0>F + 0 = F.</0>... <F>F + F = 1E.</F></F>
</no_carry>
<carry>
<0><0>0 + 0 + 1 = 1.</0>... <F>0 + F + 1 = 10.</F></0>
...
<F><0>F + 0 + 1 = F.</0>... <F>F + F + 1 = 1F.</F></F>
</carry>

Question: In base-16, what is EC + DD?
Answer:
EC is E, C. DD is D, D. So the steps are C + D, E + D.
There is no carry. <no_carry><C><D>C + D = 19. 19 is 1, 9. So we set the carry to 1. Prepend 9 to the answer. So far the answer has 1 digit: 9.
The carry is 1. <carry><E><D>E + D + 1 = 1C. 1C is 1, C. So we set the carry to 1. Prepend C to the answer. So far the answer has 2 digits: C, 9.
The carry is 1. Prepend 1 to the answer. So far the answer has 3 digits: 1, C, 9.
Therefore, the answer is 1C9.

Question: In base-16, what is 18 + 9F?
Answer:
18 is 1, 8. 9F is 9, F. So the steps are 8 + F, 1 + 9.
There is no carry. <no_carry><8><F>8 + F = 17. 17 is 1, 7. So we set the carry to 1. Prepend 7 to the answer. So far the answer has 1 digit: 7.
The carry is 1. <carry><1><9>1 + 9 + 1 = B. B is 0, B. So we clear the carry.
Prepend B to the answer. So far the answer has 2 digits: B, 7.
There is no carry. So far the answer has 2 digits: B, 7.
Therefore, the answer is B7.

Question: In base-16, what is 79 + 8B?
Answer:
79 is 7, 9. 8B is 8, B. So the steps are 9 + B, 7 + 8.
There is no carry. <no_carry><9><B>9 + B = 14. 14 is 1, 4. So we set the carry to 1. Prepend 4 to the answer. So far the answer has 1 digit: 4.
The carry is 1. <carry><7><8>7 + 8 + 1 = 10. 10 is 1, 0. So we set the carry to 1. Prepend 0 to the answer. So far the answer has 2 digits: 0, 4.
The carry is 1. Prepend 1 to the answer. So far the answer has 3 digits: 1, 0, 4.
Therefore, the answer is 104.

Question: In base-16, what is A6 + 94?
Answer:
A6 is A, 6. 94 is 9, 4. So the steps are 6 + 4, A + 9.
There is no carry. <no_carry><6><4>6 + 4 = A. A is 0, A. So we clear the carry. Prepend A to the answer. So far the answer has 1 digit: A.
There is no carry. <no_carry><A><9>A + 9 = 13. 13 is 1, 3. So we set the carry to 1. Prepend 3 to the answer. So far the answer has 2 digits: 3, A.
The carry is 1. Prepend 1 to the answer. So far the answer has 3 digits: 1, 3, A.
Therefore, the answer is 13A.

Question: In base-16, what is 54 + D3?
Answer:
54 is 5, 4. D3 is D, 3. So the steps are 4 + 3, 5 + D.
There is no carry. <no_carry><4><3>4 + 3 = 7. 7 is 0, 7. So we clear the carry. Prepend 7 to the answer. So far the answer has 1 digit: 7.
There is no carry. <no_carry><5><D>5 + D = 12. 12 is 1, 2. So we set the carry to 1. Prepend 2 to the answer. So far the answer has 2 digits: 2, 7.
The carry is 1. Prepend 1 to the answer. So far the answer has 3 digits: 1, 2, 7.
Therefore, the answer is 127.

Question: In base-16, what is {{ x }} + {{ y }}?
Answer:

Prompt 4: 5-shot LtM prompt for Arithmetic. LtM uses the same prompt as CoT for deductive reasoning, except that LtM calls the following prompt whenever it encounters a rule.

Question: In base-16, what is E + D + 1?
Answer: E + D + 1 = 1C.

Question: In base-16, what is 8 + F?
Answer: 8 + F = 17.

Question: In base-16, what is 7 + 8 + 1?
Answer: 7 + 8 + 1 = 10.

Question: In base-16, what is 6 + 4?
Answer: 6 + 4 = A.

Question: In base-16, what is 5 + D?
Answer: 5 + D = 12.

Question: In base-16, what is {{ x }} + {{ y }}?
Answer:

Prompt 5: 5-shot LtM+HtT prompt for Arithmetic. LtM+HtT uses the same prompt as CoT for deductive reasoning, except that LtM+HtT calls the following prompt whenever it encounters a rule.

Instruction: When you answer the questions, try to use the provided knowledge whenever possible. Try not to invent knowledge by yourself unless necessary.
Knowledge:
<no_carry>
<0><0>0 + 0 = 0.</0>... <F>0 + F = F.</F></0>
...
<F><0>F + 0 = F.</0>... <F>F + F = 1E.</F></F>
</no_carry>
<carry>
<0><0>0 + 0 + 1 = 1.</0>... <F>0 + F + 1 = 10.</F></0>
...
<F><0>F + 0 + 1 = F.</0>... <F>F + F + 1 = 1F.</F></F>
</carry>

Question: In base-16, what is E + D + 1?
Answer: We retrieve <carry><E><D>E + D + 1 = 1C.

Question: In base-16, what is 8 + F?
Answer: We retrieve <no_carry><8><F>8 + F = 17.

Question: In base-16, what is 7 + 8 + 1?
Answer: We retrieve <carry><7><8>7 + 8 + 1 = 10.

Question: In base-16, what is 6 + 4?
Answer: We retrieve <no_carry><6><4>6 + 4 = A.

Question: In base-16, what is 5 + D?
Answer: We retrieve <no_carry><5><D>5 + D = 12.

Question: In base-16, what is {{ x }} + {{ y }}?
Answer:

E.2 Relational Reasoning

Prompt 6-10 list the prompts we use for different prompting methods on the symbolic version of CLUTRR. Prompt 11 and 12 are used for the experiments on the textual version of CLUTRR.

Prompt 6: 0-shot CoT prompt for CLUTRR.

Context: The relations on the path from {{ head }} to {{ tail }} are {{ relations | join(", ") }}.
Question: {{ tail }} is {{ head }}'s what?
Answer: Let's think step by step.

Prompt 7: 5-shot CoT prompt for CLUTRR.

Context: The relations on the path from Alan to Anthony are daughter, uncle, son.
Question: Anthony is Alan's what?
Answer:
For daughter's uncle, we have daughter's uncle is brother. So the relations are reduced to brother, son.
For brother's son, we have brother's son is nephew. So the relations are reduced to nephew.
Therefore, the answer is nephew.

Context: The relations on the path from Annie to Carlos are brother, mother, son.
Question: Carlos is Annie's what?
Answer:
For brother's mother, we have brother's mother is mother. So the relations are reduced to mother, son.
For mother's son, we have mother's son is brother. So the relations are reduced to brother.
Therefore, the answer is brother.

Context: The relations on the path from Beverly to Michelle are father, daughter, aunt.
Question: Michelle is Beverly's what?
Answer:
For father's daughter, we have father's daughter is sister. So the relations are reduced to sister, aunt.
For sister's aunt, we have sister's aunt is aunt. So the relations are reduced to aunt.
Therefore, the answer is aunt.

Context: The relations on the path from Lee to Jeanna are father, daughter, sister.
Question: Jeanna is Lee's what?
Answer:
For father's daughter, we have father's daughter is sister. So the relations are reduced to sister, sister.
For sister's sister, we have sister's sister is sister. So the relations are reduced to sister.
Therefore, the answer is sister.

Context: The relations on the path from Craig to Molly are sister, father, mother.
Question: Molly is Craig's what?
Answer:
For sister's father, we have sister's father is father. So the relations are reduced to father, mother.
For father's mother, we have father's mother is grandmother. So the relations are reduced to grandmother.
Therefore, the answer is grandmother.

Context: The relations on the path from {{ head }} to {{ tail }} are {{ relations | join(", ") }}.
Question: {{ tail }} is {{ head }}'s what?
Answer:

Prompt 8: 5-shot CoT+HtT prompt for CLUTRR.

Instruction: When you answer the questions, try to use the provided knowledge whenever possible. Try not to invent knowledge by yourself unless necessary.
Knowledge:
<aunt><brother>aunt's brother is uncle.</brother>... <aunt><son>aunt's son is cousin.</son></aunt>
...
<wife><brother>wife's brother is brother-in-law.</brother>... <son>wife's son is son.</son></wife>

Context: The relations on the path from Alan to Anthony are daughter, uncle, son.
Question: Anthony is Alan's what?
Answer:
For daughter's uncle, we retrieve <daughter><uncle>daughter's uncle is brother. So the relations are reduced to brother, son.
For brother's son, we retrieve <brother><son>brother's son is nephew. So the relations are reduced to nephew.
Therefore, the answer is nephew.

Context: The relations on the path from Annie to Carlos are brother, mother, son.
Question: Carlos is Annie's what?

Answer:
For brother's mother, we retrieve <brother><mother>brother's mother is mother. So the relations are reduced to mother, son.
For mother's son, we retrieve <mother><son>mother's son is brother. So the relations are reduced to brother.
Therefore, the answer is brother.

Context: The relations on the path from Beverly to Michelle are father, daughter, aunt.
Question: Michelle is Beverly's what?
Answer:
For father's daughter, we retrieve <father><daughter>father's daughter is sister. So the relations are reduced to sister, aunt.
For sister's aunt, we retrieve <sister><aunt>sister's aunt is aunt. So the relations are reduced to aunt.
Therefore, the answer is aunt.

Context: The relations on the path from Lee to Jeanna are father, daughter, sister.
Question: Jeanna is Lee's what?
Answer:
For father's daughter, we retrieve <father><daughter>father's daughter is sister. So the relations are reduced to sister, sister.
For sister's sister, we retrieve
<sister><sister>Sister's sister is sister. So the relations are reduced to sister.
Therefore, the answer is sister.

Context: The relations on the path from Craig to Molly are sister, father, mother.
Question: Molly is Craig's what?
Answer:
For sister's father, we retrieve <sister><father>sister's father is father. So the relations are reduced to father, mother.
For father's mother, we retrieve <father><mother>father's mother is grandmother. So the relations are reduced to grandmother.
Therefore, the answer is grandmother.

Context: The relations on the path from {{ head }} to {{ tail }} are {{ relations | join(", ") }}.
Question: {{ tail }} is {{ head }}'s what?
Answer:

Prompt 9: 5-shot LtM prompt for CLUTRR. LtM uses the same prompt as CoT for deductive reasoning, except that LtM calls the following prompt whenever it encounters a rule.

Question: What is daughter's uncle?
Answer: Daughter's uncle is mother.

Question: What is brother's mother?
Answer: Brother's mother is mother.

Question: What is sister's aunt?
Answer: Sister's aunt is aunt.

Question: What is father's daughter?
Answer: Father's daughter is sister.

Question: What is sister's father?
Answer: Sister's father is father.

Question: What is {{ relations[0] }}'s {{ relations[1] }}?
Answer:

Prompt 10: 5-shot LtM+HtT prompt for CLUTRR. LtM+HtT uses the same prompt as CoT for deductive reasoning, except that LtM+HtT calls the following prompt whenever it encounters a rule.

Instruction: When you answer the questions, try to use the provided knowledge whenever possible. Try not to invent knowledge by yourself unless necessary.
Knowledge:
<aunt><brother>aunt's brother is uncle.</brother>... <aunt><son>aunt's son is cousin.</son></aunt>
...
<wife><brother>wife's brother is brother-in-law.</brother>... <son>wife's son is son.</son></wife>

Question: What is daughter's uncle?
Answer: <daughter><uncle>Daughter's uncle is mother.

Question: What is brother's mother?
Answer: <brother><mother>Brother's mother is mother.

Question: What is sister's aunt?
Answer: <sister><aunt>Sister's aunt is aunt.

Question: What is father's daughter?
Answer: <father><daughter>Father's daughter is sister.

Question: What is sister's father?
Answer: <sister><father>Sister's father is father.

Question: What is {{ relations[0] }}'s {{ relations[1] }}?
Answer:

Prompt 11: 5-shot CoT prompt for the textual version of CLUTRR.

Document: Anthony went to the park with his father, James. Annie took her uncle James to the grocery store. Alan and his daughter Annie spent Father's Day together. Annie took her dad out to a sports bar, and they had a great time watching football and drinking beer there.
Question: Anthony is Alan's what?
Answer: We first extract all triplets from the document. We then find the path from Alan to Anthony. Finally, we reduce the relations on the path to get the answer.
The triplets include (Anthony, father, James), (Annie, uncle, James), (Alan, daughter, Annie).
The path from Alan to Anthony is (Alan, daughter, Annie), (Annie, uncle, James), (James, son, Anthony).
The relations on the path are daughter, uncle, son.
Daughter's uncle is brother. So the relations are reduced to brother, son.
Brother's son is nephew. So the relations are reduced to nephew.
Therefore, Anthony is Alan's nephew.

Document: Valerie's biggest accomplishment is raising her son Carlos. Annie does n't like having to babysit her younger brother, Emmanuel. Valerie and her son Emmanuel had lunch together at a local Chinese restaurant.
Question: Carlos is Annie's what?
Answer: We first extract all triplets from the document. We then find the path from Annie to Carlos. Finally, we reduce the relations on the path to get the answer.
The triplets include (Valerie, son, Carlos), (Annie, brother, Emmanuel), (Valerie, son, Emmanuel).
The path from Annie to Carlos is (Annie, brother, Emmanuel), (Emmanuel, mother, Valerie), (Valerie, son, Carlos). The relations on the path are brother, mother, son.
Brother's mother is mother. So the relations are reduced to mother, son.
Mother's son is brother. So the relations are reduced to brother.
Therefore, Carlos is Annie's brother.

Document: James likes to take his daughter Jeanna fishing. James loves cooking with his daughter. Her name is Beverly. Jeanna loves visiting with her aunt Michelle.
Question: Michelle is Beverly's what?
Answer: We first extract all triplets from the document. We then find the path from Beverly to Michelle. Finally, we reduce the relations on the path to get the answer.

The triplets include (James, daughter, Jeanna), (James, daughter, Beverly), (Jeanna, aunt, Michelle).
The path from Beverly to Michelle is (Beverly, father, James), (James, daughter, Jeanna), (Jeanna, aunt, Michelle). The relations on the path are father, daughter, aunt.
Father's daughter is sister. So the relations are reduced to sister, aunt.
Sister's aunt is aunt. So the relations are reduced to aunt.
Therefore, Michelle is Beverly's aunt.

Document: Lee was finally coming of age and it was time for him and his father to go on a coming of age camping trip. Beverly, James's younger daughter, decided she wanted to go on the trip despite being several years younger. Jeanna took her younger sister Beverly to the carnival last weekend.
Question: Jeanna is Lee's what?
Answer: We first extract all triplets from the document. We then find the path from Lee to Jeanna. Finally, we reduce the relations on the path to get the answer.
The triplets include (Lee, father, James), (James, daughter, Beverly), (Jeanna, sister, Beverly).
The path from Lee to Jeanna is (Lee, father, James), (James, daughter, Beverly), (Beverly, sister, Jeanna). The relations on the path are father, daughter, sister.
Father's daughter is sister. So the relations are reduced to sister, sister.
Sister's sister is sister. So the relations are reduced to sister.
Therefore, Jeanna is Lee's sister.

Document: Craig's sister, Rosie, bought movie tickets at a discount rate. Rosie and her father Elliott love to go skiing. Often, Elliott will invite his mother Molly to join them.
Question: Molly is Craig's what?
Answer: We first extract all triplets from the document. We then find the path from Craig to Molly. Finally, we reduce the relations on the path to get the answer.
The triplets include (Craig, sister, Rosie), (Rosie, father, Elliott), (Elliott, mother, Molly).
The path from Craig to Molly is (Craig, sister, Rosie), (Rosie, father, Elliott), (Elliott, mother, Molly). The relations on the path are sister, father, mother.
Sister's father is father. So the relations are reduced to father, mother.
Father's mother is grandmother. So the relations are reduced to grandmother.
Therefore, Molly is Craig's grandmother.

Document: {{ document }}
Question: {{ tail }} is {{ head }}'s what?
Answer:

Prompt 12: 5-shot CoT+HtT prompt for the textual version of CLUTRR.

Instruction: When you answer the questions, try to use the provided knowledge whenever possible. Try not to invent knowledge by yourself unless necessary.
Knowledge:
<aunt><brother>aunt's brother is uncle.</brother>... <aunt><son>aunt's son is cousin.</son></aunt>
...
<wife><brother>wife's brother is brother-in-law.</brother>... <son>wife's son is son.</son></wife>

Document: Anthony went to the park with his father, James. Annie took her uncle James to the grocery store. Alan and his daughter Annie spent Father's Day together. Annie took her dad out to a sports bar, and they had a great time watching football and drinking beer there.
Question: Anthony is Alan's what?
Answer: We first extract all triplets from the document. We then find the path from Alan to Anthony. Finally, we reduce the relations on the path to get the answer.
The triplets include (Anthony, father, James), (Annie, uncle, James), (Alan, daughter, Annie).
The path from Alan to Anthony is (Alan, daughter, Annie), (Annie, uncle, James), (James, son, Anthony). The relations on the path are daughter, uncle, son.
<daughter><uncle>Daughter's uncle is brother. So the relations are reduced to brother, son.
<brother><son>Brother's son is nephew. So the relations are reduced to nephew.
Therefore, Anthony is Alan's nephew.

Document: Valerie's biggest accomplishment is raising her son Carlos. Annie does n't like having to babysit her younger brother, Emmanuel. Valerie and her son Emmanuel had lunch together at a local Chinese restaurant.
Question: Carlos is Annie's what?

Answer: We first extract all triplets from the document. We then find the path from Annie to Carlos. Finally, we reduce the relations on the path to get the answer.
The triplets include (Valerie, son, Carlos), (Annie, brother, Emmanuel), (Valerie, son, Emmanuel).
The path from Annie to Carlos is (Annie, brother, Emmanuel), (Emmanuel, mother, Valerie), (Valerie, son, Carlos). The relations on the path are brother, mother, son.
<brother><mother>Brother's mother is mother. So the relations are reduced to mother, son.
<mother><son>Mother's son is brother. So the relations are reduced to brother.
Therefore, Carlos is Annie's brother.

Document: James likes to take his daughter Jeanna fishing. James loves cooking with his daughter. Her name is Beverly. Jeanna loves visiting with her aunt Michelle.
Question: Michelle is Beverly's what?
Answer: We first extract all triplets from the document. We then find the path from Beverly to Michelle. Finally, we reduce the relations on the path to get the answer.
The triplets include (James, daughter, Jeanna), (James, daughter, Beverly), (Jeanna, aunt, Michelle).
The path from Beverly to Michelle is (Beverly, father, James), (James, daughter, Jeanna), (Jeanna, aunt, Michelle). The relations on the path are father, daughter, aunt.
<father><daughter>Father's daughter is sister. So the relations are reduced to sister, aunt.
<sister><aunt>Sister's aunt is aunt. So the relations are reduced to aunt.
Therefore, Michelle is Beverly's aunt.

Document: Lee was finally coming of age and it was time for him and his father to go on a coming of age camping trip. Beverly, James's younger daughter, decided she wanted to go on the trip despite being several years younger. Jeanna took her younger sister Beverly to the carnival last weekend.
Question: Jeanna is Lee's what?
Answer: We first extract all triplets from the document. We then find the path from Lee to Jeanna. Finally, we reduce the relations on the path to get the answer.
The triplets include (Lee, father, James), (James, daughter, Beverly), (Jeanna, sister, Beverly).
The path from Lee to Jeanna is (Lee, father, James), (James, daughter, Beverly), (Beverly, sister, Jeanna). The relations on the path are father, daughter, sister.
<father><daughter>Father's daughter is sister. So the relations are reduced to sister, sister.
<sister><sister>Sister's sister is sister. So the relations are reduced to sister.
Therefore, Jeanna is Lee's sister.

Document: Craig's sister, Rosie, bought movie tickets at a discount rate. Rosie and her father Elliott love to go skiing. Often, Elliott will invite his mother Molly to join them.
Question: Molly is Craig's what?
Answer: We first extract all triplets from the document. We then find the path from Craig to Molly. Finally, we reduce the relations on the path to get the answer.
The triplets include (Craig, sister, Rosie), (Rosie, father, Elliott), (Elliott, mother, Molly).
The path from Craig to Molly is (Craig, sister, Rosie), (Rosie, father, Elliott), (Elliott, mother, Molly). The relations on the path are sister, father, mother.
<sister><father>Sister's father is father. So the relations are reduced to father, mother.
<father><mother>Father's mother is grandmother. So the relations are reduced to grandmother.
Therefore, Molly is Craig's grandmother.

Document: {{ document }}
Question: {{ tail }} is {{ head }}'s what?
Answer:

F Rule Libraries Learned by HtT

Here we present rule libraries learned by HtT on GPT4. For numerical reasoning, we compare the learned rules with rules induced by an oracle model, and compute precision and recall for each rule library. For relational reasoning, due to lack of ground truth rules, we cannot report precision and recall for this problem.

F.1 Numerical Reasoning

We illustrate the rule libraries for base-16, base-11 and base-9 in Prompt 13-15 respectively.

Prompt 13: 413 rules learned by GPT4 on base-16. Precision: 98.5%. Recall: 92.2%.

0 + 0 = 0. 0 + 1 = 1. 0 + 2 = 2. 0 + 3 = 3. 0 + 4 = 4. 0 + 5 = 5. 0 + 6 = 6. 0 + 7 = 7. 0 + 8 = 8. 0 + 9 = 9. 0 + A = A. 0 + B = B. 0 + C = C. 0 + D = D. 0 + E = E. 0 + F = F. 1 + 0 = 1. 1 + 1 = 2. 1 + 2 = 3. 1 + 3 = 4. 1 + 4 = 5. 1 + 5 = 6. 1 + 7 = 8. 1 + 8 = 9. 1 + 9 = A. 1 + A = B. 1 + B = C. 1 + C = D. 1 + D = E. 1 + E = F. 1 + F = 10. 2 + 0 = 2. 2 + 1 = 3. 2 + 2 = 4. 2 + 3 = 5. 2 + 4 = 6. 2 + 5 = 7. 2 + 6 = 8. 2 + 7 = 9. 2 + 8 = A. 2 + 9 = B. 2 + A = C. 2 + C = E. 2 + D = F. 2 + E = 10. 2 + F = 11. 3 + 0 = 3. 3 + 1 = 4. 3 + 2 = 5. 3 + 3 = 6. 3 + 4 = 7. 3 + 5 = 8. 3 + 6 = 9. 3 + 7 = A. 3 + 8 = B. 3 + 9 = C. 3 + A = D. 3 + B = E. 3 + C = F. 3 + D = 10. 3 + D = 16. 3 + E = 11. 3 + F = 12. 4 + 0 = 4. 4 + 1 = 5. 4 + 2 = 6. 4 + 3 = 7. 4 + 4 = 8. 4 + 5 = 9. 4 + 6 = A. 4 + 7 = B. 4 + 8 = C. 4 + 9 = D. 4 + A = E. 4 + B = F. 4 + C = 10. 4 + E = 12. 4 + F = 13. 5 + 0 = 5. 5 + 1 = 6. 5 + 2 = 7. 5 + 3 = 8. 5 + 4 = 9. 5 + 5 = A. 5 + 6 = B. 5 + 7 = C. 5 + 8 = D. 5 + 9 = E. 5 + A = F. 5 + B = 10. 5 + C = 11. 5 + D = 12. 5 + E = 13. 5 + F = 14. 6 + 0 = 6. 6 + 1 = 7. 6 + 2 = 8. 6 + 3 = 9. 6 + 4 = A. 6 + 5 = B. 6 + 6 = C. 6 + 7 = D. 6 + 8 = E. 6 + 9 = F. 6 + A = 10. 6 + B = 11. 6 + C = 12. 6 + D = 13. 6 + E = 14. 6 + F = 15. 7 + 0 = 7. 7 + 1 = 8. 7 + 2 = 9. 7 + 3 = A. 7 + 4 = B. 7 + 5 = C. 7 + 6 = D. 7 + 7 = E. 7 + 8 = F. 7 + 9 = 10. 7 + 9 = 16. 7 + A = 11. 7 + B = 12. 7 + C = 13. 7 + D = 14. 7 + E = 15. 7 + F = 16. 8 + 0 = 8. 8 + 1 = 9. 8 + 2 = A. 8 + 3 = B. 8 + 4 = C. 8 + 5 = D. 8 + 6 = E. 8 + 7 = F. 8 + 8 = 10. 8 + 9 = 11. 8 + A = 12. 8 + B = 13. 8 + C = 14. 8 + D = 15. 8 + F = 17. 9 + 0 = 9. 9 + 1 = A. 9 + 2 = B. 9 + 3 = C. 9 + 4 = D. 9 + 5 = E. 9 + 6 = F. 9 + 7 = 10. 9 + 7 = 16. 9 + 8 = 11. 9 + 9 = 12. 9 + A = 13. 9 + B = 14. 9 + C = 15. 9 + E = 17. 9 + F = 18. A + 1 = B. A + 2 = C. A + 3 = D. A + 4 = E. A + 5 = F. A + 6 = 10. A + 7 = 11. A + 8 = 12. A + 9 = 13. A + A = 14. A + B = 15. A + D = 17. A + E = 18. A + F = 19. B + 0 = B. B + 1 = C. B + 2 = D. B + 3 = E. B + 4 = F. B + 5 = 10. B + 6 = 11. B + 7 = 12. B + 8 = 13. B + 9 = 14. B + A = 15. B + C = 17. B + D = 18. B + E = 19. B + F = 1A. C + 0 = C. C + 1 = D. C + 2 = E. C + 3 = F. C + 4 = 10. C + 5 = 11. C + 6 = 12. C + 7 = 13. C + 8 = 14. C + 9 = 15. C + B = 17. C + C = 18. C + D = 19. C + E = 1A. C + F = 1B. D + 0 = D. D + 1 = E. D + 2 = F. D + 3 = 10. D + 4 = 11. D + 5 = 12. D + 6 = 13. D + 7 = 14. D + 9 = 16. D + A = 17. D + B = 18. D + C = 19. D + D = 1A. D + E = 1B. D + F = 1C. E + 0 = E. E + 1 = F. E + 2 = 10. E + 3 = 11. E + 4 = 12. E + 5 = 13. E + 6 = 14. E + 7 = 15. E + 8 = 16. E + 9 = 17. E + A = 18. E + B = 19. E + C = 1A. E + D = 1B. E + E = 1C. E + F = 1D. F + 0 = F. F + 1 = 10. F + 2 = 11. F + 3 = 12. F + 4 = 13. F + 5 = 14. F + 6 = 15. F + 8 = 17. F + 9 = 18. F + A = 19. F + B = 1A. F + C = 1B. F + D = 1C. F + E = 1D. F + F = 1E. 1 + 1 + 1 = 3. 1 + 3 + 1 = 5. 1 + 4 + 1 = 6. 1 + 6 + 1 = 8. 1 + 7 + 1 = 9. 1 + 8 + 1 = A. 1 + 9 + 1 = B. 1 + A + 1 = C. 1 + B + 1 = D. 1 + C + 1 = E. 1 + F + 1 = 11. 2 + 1 + 1 = 4. 2 + 2 + 1 = 5. 2 + 3 + 1 = 6. 2 + 4 + 1 = 7. 2 + 5 + 1 = 8. 2 + 6 + 1 = 9. 2 + 7 + 1 = A. 2 + 8 + 1 = B. 2 + 9 + 1 = C. 2 + A + 1 = D. 2 + B + 1 = E. 2 + C + 1 = F. 3 + 1 + 1 = 5. 3 + 2 + 1 = 6. 3 + 3 + 1 = 7. 3 + 4 + 1 = 8. 3 + 5 + 1 = 9. 3 + 6 + 1 = A. 3 + B + 1 = F. 3 + D + 1 = 11. 3 + E + 1 = 12. 3 + F + 1 = 13. 4 + 2 + 1 = 7. 4 + 3 + 1 = 8. 4 + 4 + 1 = 9. 4 + 6 + 1 = B. 4 + 7 + 1 = C. 4 + 8 + 1 = D. 4 + 9 + 1 = E. 4 + B + 1 = 10. 4 + C + 1 = 11. 4 + E + 1 = 13. 4 + F + 1 = 14. 5 + 1 + 1 = 7. 5 + 2 + 1 = 8. 5 + 5 + 1 = B. 5 + 6 + 1 = C. 5 + 7 + 1 = D. 5 + 8 + 1 = E. 5 + 9 + 1 = 15. 5 + 9 + 1 = F. 5 + A + 1 = 10. 5 + B + 1 = 11. 5 + C + 1 = 12. 5 + E + 1 = 14. 6 + 1 + 1 = 8. 6 + 2 + 1 = 9. 6 + 3 + 1 = A. 6 + 4 + 1 = B. 6 + 5 + 1 = C. 6 + 6 + 1 = D. 6 + 8 + 1 = F. 6 + 9 + 1 = 10. 6 + A + 1 = 11. 6 + B + 1 = 12. 6 + C + 1 = 13. 6 + D + 1 = 14. 6 + E + 1 = 15. 7 + 1 + 1 = 9. 7 + 2 + 1 = A. 7 + 3 + 1 = B. 7 + 4 + 1 = C. 7 + 5 + 1 = D. 7 + 8 + 1 = 10. 7 + 8 + 1 = 16. 7 + B + 1 = 13. 7 + C + 1 = 14. 7 + D + 1 = 15. 7 + F + 1 = 17. 8 + 2 + 1 = B. 8 + 3 + 1 = C. 8 + 5 + 1 = E. 8 + 6 + 1 = F. 8 + A + 1 = 13. 8 + B + 1 = 14. 8 + D + 1 = 16. 8 + E + 1 = 17. 8 + F + 1 = 18. 9 + 1 + 1 = B. 9 + 2 + 1 = C. 9 + 3 + 1 = D. 9 + 4 + 1 = E. 9 + 5 + 1 = F. 9 + 7 + 1 = 11. 9 + 9 + 1 = 13. 9 + 9 + 1 = 19. 9 + C + 1 = 16. 9 + E + 1 = 18. A + 1 + 1 = C. A + 3 + 1 = E. A + 4 + 1 = F. A + 5 + 1 = 10. A + 6 + 1 = 11. A + 7 + 1 = 12. A + 8 + 1 = 13. A + 9 + 1 = 14. A + A + 1 = 15. A + C + 1 = 17. A + D + 1 = 18. A + E + 1 = 19. B + 1 + 1 = D. B + 2 + 1 = E. B + 3 + 1 = F. B + 4 + 1 = 10. B + 5 + 1 = 11. B + 7 + 1 = 13. B + 9 + 1 = 15. B + C + 1 = 18. B + D + 1 = 19. B + E + 1 = 1A. B + F + 1 = 1B. C + 2 + 1 = F. C + 3 + 1 = 10. C + 4 + 1 = 11. C + 5 + 1 = 12. C + 7 + 1 = 14. C + 8 + 1 = 15. C + A + 1 = 17. C + B + 1 = 18. C + C + 1 = 19. C + D + 1 = 1A. C + F + 1 = 1C. D + 1 + 1 = F. D + 2 + 1 = 10. D + 3 + 1 = 11. D + 4 + 1 = 12. D + 5 + 1 = 13. D + 7 + 1 = 15. D + A + 1 = 18. D + B + 1 = 19. D + D + 1 = 1B. D + F + 1 = 1D. E + 1 + 1 = 10. E + 2 + 1 = 11. E + 4 + 1 = 13. E + 5 + 1 = 14. E + 6 + 1 = 15. E + 8 + 1 = 17. E + B + 1 = 1A. E + C + 1 = 1B. E + D + 1 = 1C. E + E + 1 = 1D. E + F + 1 = 1E. F + 1 + 1 = 11. F + 3 + 1 = 13. F + 4 + 1 = 14. F + 5 + 1 = 15. F + 7 + 1 = 17. F + 8 + 1 = 18. F + 9 + 1 = 19. F + A + 1 = 1A. F + C + 1 = 1C. F + D + 1 = 1D. F + E + 1 = 1E.

Prompt 14: 220 rules learned by GPT4 on base-11. Precision: 90.5%. Recall: 99.5%.

0 + 0 = 0. 0 + 1 = 1. 0 + 2 = 2. 0 + 3 = 3. 0 + 4 = 4. 0 + 5 = 5. 0 + 6 = 6. 0 + 7 = 7. 0 + 8 = 8. 0 + 9 = 9. 0 + A = A. 1 + 0 = 1. 1 + 1 = 2. 1 + 2 = 3. 1 + 3 = 4. 1 + 4 = 5. 1 + 5 = 6. 1 + 6 = 7. 1 + 7 = 8. 1 + 8 = 9. 1 + 9 = A. 1 + A = 10. 2 + 0 = 2. 2 + 1 = 3. 2 + 2 = 4. 2 + 3 = 5. 2 + 4 = 6. 2 + 5 = 7. 2 + 6 = 8. 2 + 7 = 9. 2 + 8 = A. 2 + 9 = 10. 2 + 9 = 11. 2 + A = 11. 3 + 0 = 3. 3 + 1 = 4. 3 + 2 = 5. 3 + 3 = 6. 3 + 4 = 7. 3 + 5 = 8. 3 + 6 = 9. 3 + 7 = A. 3 + 8 = 10. 3 + 8 = 11. 3 + 9 = 11. 3 + A = 12. 4 + 0 = 4. 4 + 1 = 5. 4 + 2 = 6. 4 + 3 = 7. 4 + 4 = 8. 4 + 5 = 9. 4 + 6 = A. 4 + 7 = 10. 4 + 8 = 11. 4 + 9 = 12. 4 + A = 13. 5 + 0 = 5. 5 + 1 = 6.

5 + 2 = 7. 5 + 3 = 8. 5 + 4 = 9. 5 + 5 = A. 5 + 6 = 10. 5 + 8 = 12. 5 + 9 = 13. 5 + 9 = 14. 5 + A = 14. 6 + 0 = 6. 6 + 1 = 7. 6 + 2 = 8. 6 + 3 = 9. 6 + 4 = A. 6 + 5 = 10. 6 + 5 = 11. 6 + 6 = 11. 6 + 6 = 12. 6 + 7 = 12. 6 + 8 = 13. 6 + 9 = 14. 6 + A = 15. 7 + 0 = 7. 7 + 1 = 8. 7 + 2 = 9. 7 + 3 = A. 7 + 4 = 10. 7 + 4 = 11. 7 + 5 = 11. 7 + 6 = 12. 7 + 7 = 13. 7 + 9 = 15. 7 + A = 16. 8 + 0 = 8. 8 + 1 = 9. 8 + 3 = 10. 8 + 3 = 11. 8 + 4 = 11. 8 + 5 = 12. 8 + 6 = 13. 8 + 7 = 14. 8 + 8 = 15. 8 + 9 = 16. 8 + A = 17. 9 + 0 = 9. 9 + 1 = A. 9 + 2 = 10. 9 + 2 = 11. 9 + 3 = 11. 9 + 4 = 12. 9 + 5 = 13. 9 + 6 = 14. 9 + 6 = 15. 9 + 7 = 15. 9 + 8 = 16. 9 + 9 = 17. 9 + 9 = 18. 9 + A = 18. A + 0 = A. A + 1 = 10. A + 1 = 11. A + 2 = 11. A + 3 = 12. A + 4 = 13. A + 5 = 14. A + 6 = 15. A + 7 = 16. A + 8 = 17. A + 9 = 18. A + 9 = 19. A + A = 19. A + A = 20. 1 + 1 + 1 = 3. 1 + 2 + 1 = 4. 1 + 3 + 1 = 5. 1 + 4 + 1 = 6. 1 + 5 + 1 = 7. 1 + 6 + 1 = 8. 1 + 7 + 1 = 9. 1 + 9 + 1 = 10. 1 + A + 1 = 11. 2 + 1 + 1 = 4. 2 + 2 + 1 = 5. 2 + 3 + 1 = 6. 2 + 4 + 1 = 7. 2 + 5 + 1 = 8. 2 + 6 + 1 = 9. 2 + 7 + 1 = A. 2 + 8 + 1 = 10. 2 + 8 + 1 = 11. 2 + A + 1 = 12. 3 + 1 + 1 = 5. 3 + 2 + 1 = 6. 3 + 3 + 1 = 7. 3 + 4 + 1 = 8. 3 + 5 + 1 = 9. 3 + 7 + 1 = 10. 3 + 7 + 1 = 11. 3 + 9 + 1 = 12. 3 + A + 1 = 13. 4 + 1 + 1 = 6. 4 + 2 + 1 = 7. 4 + 3 + 1 = 8. 4 + 4 + 1 = 9. 4 + 5 + 1 = A. 4 + 6 + 1 = 10. 4 + 8 + 1 = 12. 4 + 9 + 1 = 13. 5 + 1 + 1 = 7. 5 + 2 + 1 = 8. 5 + 3 + 1 = 9. 5 + 5 + 1 = 10. 5 + 7 + 1 = 12. 5 + 8 + 1 = 13. 5 + 8 + 1 = 14. 5 + 9 + 1 = 14. 5 + A + 1 = 15. 6 + 1 + 1 = 8. 6 + 2 + 1 = 9. 6 + 3 + 1 = A. 6 + 4 + 1 = 10. 6 + 4 + 1 = 11. 6 + 6 + 1 = 12. 6 + 7 + 1 = 13. 6 + 9 + 1 = 15. 6 + A + 1 = 16. 7 + 1 + 1 = 9. 7 + 3 + 1 = 10. 7 + 4 + 1 = 11. 7 + 5 + 1 = 12. 7 + 6 + 1 = 13. 7 + 6 + 1 = 14. 7 + 7 + 1 = 14. 7 + 8 + 1 = 15. 7 + 9 + 1 = 16. 7 + A + 1 = 17. 8 + 2 + 1 = 10. 8 + 2 + 1 = 11. 8 + 4 + 1 = 12. 8 + 5 + 1 = 13. 8 + 7 + 1 = 15. 8 + 8 + 1 = 16. 8 + 9 + 1 = 17. 8 + A + 1 = 18. 9 + 1 + 1 = 10. 9 + 2 + 1 = 11. 9 + 3 + 1 = 12. 9 + 4 + 1 = 13. 9 + 5 + 1 = 14. 9 + 6 + 1 = 15. 9 + 7 + 1 = 16. 9 + 8 + 1 = 17. 9 + 9 + 1 = 18. A + 3 + 1 = 13. A + 4 + 1 = 14. A + 5 + 1 = 15. A + 6 + 1 = 16. A + 7 + 1 = 17. A + 7 + 1 = 18. A + 8 + 1 = 18. A + 8 + 1 = 19.

Prompt 15: 124 rules learned by GPT4 on base-9. Precision: 99.2%. Recall: 85.5%.

0 + 0 = 0. 0 + 1 = 1. 0 + 2 = 2. 0 + 3 = 3. 0 + 4 = 4. 0 + 5 = 5. 0 + 6 = 6. 0 + 7 = 7. 0 + 8 = 8. 1 + 0 = 1. 1 + 1 = 2. 1 + 2 = 3. 1 + 3 = 4. 1 + 4 = 5. 1 + 5 = 6. 1 + 6 = 7. 1 + 7 = 8. 1 + 8 = 10. 2 + 0 = 2. 2 + 1 = 3. 2 + 2 = 4. 2 + 3 = 5. 2 + 4 = 6. 2 + 5 = 7. 2 + 6 = 8. 2 + 7 = 10. 2 + 8 = 11. 3 + 0 = 3. 3 + 1 = 4. 3 + 2 = 5. 3 + 3 = 6. 3 + 4 = 7. 3 + 5 = 8. 3 + 6 = 10. 3 + 7 = 11. 3 + 8 = 12. 4 + 0 = 4. 4 + 1 = 5. 4 + 2 = 6. 4 + 3 = 7. 4 + 4 = 8. 4 + 5 = 10. 4 + 6 = 11. 4 + 7 = 12. 4 + 8 = 13. 5 + 0 = 5. 5 + 1 = 6. 5 + 2 = 7. 5 + 3 = 8. 5 + 4 = 10. 5 + 5 = 11. 5 + 6 = 12. 5 + 7 = 13. 5 + 8 = 14. 6 + 0 = 6. 6 + 1 = 7. 6 + 2 = 8. 6 + 3 = 10. 6 + 4 = 11. 6 + 5 = 12. 6 + 6 = 13. 6 + 7 = 14. 6 + 8 = 15. 7 + 0 = 7. 7 + 1 = 8. 7 + 2 = 10. 7 + 3 = 11. 7 + 4 = 12. 7 + 5 = 13. 7 + 6 = 14. 7 + 7 = 15. 8 + 0 = 8. 8 + 1 = 10. 8 + 2 = 11. 8 + 3 = 12. 8 + 4 = 13. 8 + 5 = 14. 8 + 6 = 15. 8 + 7 = 16. 8 + 8 = 16. 1 + 3 + 1 = 5. 1 + 7 + 1 = 10. 1 + 8 + 1 = 11. 2 + 6 + 1 = 10. 2 + 7 + 1 = 11. 2 + 8 + 1 = 12. 3 + 1 + 1 = 5. 3 + 5 + 1 = 10. 3 + 6 + 1 = 11. 3 + 7 + 1 = 12. 3 + 8 + 1 = 13. 4 + 4 + 1 = 10. 4 + 5 + 1 = 11. 4 + 6 + 1 = 12. 4 + 7 + 1 = 13. 4 + 8 + 1 = 14. 5 + 3 + 1 = 10. 5 + 4 + 1 = 11. 5 + 5 + 1 = 12. 5 + 6 + 1 = 13. 5 + 7 + 1 = 14. 5 + 8 + 1 = 15. 6 + 2 + 1 = 10. 6 + 3 + 1 = 11. 6 + 4 + 1 = 12. 6 + 5 + 1 = 13. 6 + 6 + 1 = 14. 6 + 7 + 1 = 15. 6 + 8 + 1 = 16. 7 + 1 + 1 = 10. 7 + 2 + 1 = 11. 7 + 3 + 1 = 12. 7 + 4 + 1 = 13. 7 + 5 + 1 = 14. 7 + 6 + 1 = 15. 7 + 7 + 1 = 16. 7 + 8 + 1 = 17. 8 + 1 + 1 = 11. 8 + 2 + 1 = 12. 8 + 3 + 1 = 13. 8 + 4 + 1 = 14. 8 + 5 + 1 = 15. 8 + 6 + 1 = 16. 8 + 7 + 1 = 17.

F.2 Relational Reasoning

Prompt 16 gives the rule library learned for CLUTRR.

Prompt 16: 98 rules learned by GPT4 on CLUTRR.

Aunt's sister is aunt. Brother's aunt is aunt. Brother's brother is brother. Brother's daughter is niece. Brother's father is father. Brother's grandfather is grandfather. Brother's grandmother is grandmother. Brother's mother is mother. Brother's sister is sister. Brother's son is nephew. Brother's uncle is uncle. Brother's wife is sister-in-law. Brother-in-law's daughter is niece. Brother-in-law's father is father-in-law. Brother-in-law's mother is mother-in-law. Brother-in-law's son is nephew. Daughter's aunt is sister. Daughter's brother is son. Daughter's daughter is granddaughter. Daughter's grandfather is father. Daughter's grandmother is mother. Daughter's husband is son-in-law. Daughter's sister is daughter. Daughter's son is grandson. Daughter's uncle is brother. Daughter-in-law's daughter is granddaughter. Daughter-in-law's son is grandson. Father's brother is uncle. Father's daughter is sister. Father's father is grandfather. Father's mother is grandmother. Father's sister is aunt. Father's son is brother. Father's wife is mother. Granddaughter's brother is grandson. Granddaughter's father is son. Granddaughter's mother is daughter. Granddaughter's sister is granddaughter. Granddaughter's uncle is son. Grandfather's daughter is aunt. Grandfather's son is uncle. Grandmother's daughter is aunt. Grandmother's son is uncle. Grandson's brother is grandson. Grandson's father is son. Grandson's mother is daughter. Grandson's sister is granddaughter. Grandson's uncle is son. Husband's daughter is daughter. Husband's father is father-in-law. Husband's granddaughter is granddaughter. Husband's grandson is grandson. Husband's mother is mother-in-law. Husband's son is son. Mother's brother is uncle. Mother's daughter is sister. Mother's father is grandfather. Mother's mother is grandmother. Mother's sister is aunt. Mother's son is brother. Nephew's grandfather is father. Nephew's grandmother is mother. Nephew's sister is niece. Niece's brother is nephew. Niece's uncle is brother. Self's brother is brother. Sister's brother is brother. Sister's daughter is niece. Sister's father is father. Sister's grandfather is grandfather. Sister's grandmother is grandmother. Sister's husband is brother-in-law. Sister's mother is mother. Sister's sister is sister. Sister's son is nephew. Sister-in-law's daughter is niece. Sister-in-law's father is father-in-law. Sister-in-law's mother is mother-in-law. Sister-in-law's son is nephew. Son's aunt is sister. Son's brother is son. Son's daughter is granddaughter. Son's grandfather is father. Son's grandmother is mother. Son's sister is daughter. Son's son is grandson. Son's uncle is brother. Son's wife is daughter-in-law. Son-in-law's son is grandson. Step-daughter's grandmother is mother. Uncle's sister is aunt. Wife's brother is brother-in-law. Wife's daughter is daughter. Wife's father is father-in-law. Wife's granddaughter is granddaughter. Wife's grandson is grandson. Wife's mother is mother-in-law. Wife's son is son.

G Examples of Model Outputs

G.1 Numerical Reasoning

Prompt 17: Example outputs on Arithmetic. Errors are marked in red. Upper: CoT. Lower: CoT+HtT.

Question: In base-16, what is F55 + 85E?
Answer:
F55 is F, 5, 5. 85E is 8, 5, E. So the steps are 5 + E, 5 + 5, F + 8.
There is no carry. 5 + E = 19. 19 is 1, 9. So we set the carry to 1. Prepend 9 to the answer. So far the answer has 1 digit: 9.
The carry is 1. 5 + 5 + 1 = 11. 11 is 1, 1. So we set the carry to 1. Prepend 1 to the answer. So far the answer has 2 digits: 1, 9.
The carry is 1. F + 8 + 1 = 1E. 1E is 1, 4. So we set the carry to 1. Prepend E to the answer. So far the answer has 3 digits: E, 1, 9.
The carry is 1. Prepend 1 to the answer. So far the answer has 4 digits: 1, E, 1, 9.
Therefore, the answer is 1E19.

Question: In base-16, what is F55 + 85E?
Answer:
F55 is F, 5, 5. 85E is 8, 5, E. So the steps are 5 + E, 5 + 5, F + 8.
There is no carry. <no_carry><5><E>5 + E = 13. 13 is 1, 3. So we set the carry to 1. Prepend 3 to the answer. So far the answer has 1 digit: 3.
The carry is 1. <carry><5><5>5 + 5 + 1 = B. B is 0, B. So we clear the carry. Prepend B to the answer. So far the answer has 2 digits: B, 3.
There is no carry. <no_carry><F><8>F + 8 = 17. 17 is 1, 7. So we set the carry to 1. Prepend 7 to the answer. So far the answer has 3 digits: 7, B, 3.
The carry is 1. Prepend 1 to the answer. So far the answer has 4 digits: 1, 7, B, 3.
Therefore, the answer is 17B3.

Prompt 18: Example outputs on Arithmetic. Errors are marked in red. Upper: CoT. Lower: CoT+HtT.

Question: In base-16, what is 47D2 + D292?
Answer:
47D2 is 4, 7, D, 2. D292 is D, 2, 9, 2. So the steps are 2 + 2, D + 9, 7 + 2, 4 + D.
There is no carry. 2 + 2 = 4. 4 is 0, 4. So we clear the carry. Prepend 4 to the answer. So far the answer has 1 digit: 4.
There is no carry. D + 9 = 1C. 1C is 1, C. So we set the carry to 1. Prepend C to the answer. So far the answer has 2 digits: C, 4.
The carry is 1. 7 + 2 + 1 = A. A is 0, A. So we clear the carry. Prepend A to the answer. So far the answer has 3 digits: A, C, 4.

The carry is 0. 4 + D = 11. 11 is 1, 1. So we set the carry to 1. Prepend 1 to the answer. So far the answer has 4 digits: 1, A, C, 4.
The carry is 1. Prepend 1 to the answer. So far the answer has 5 digits: 1, 1, A, C, 4.
Therefore, the answer is 11AC4.

Question: In base-16, what is 47D2 + D292?
Answer:
47D2 is 4, 7, D, 2. D292 is D, 2, 9, 2. So the steps are 2 + 2, D + 9, 7 + 2, 4 + D.
There is no carry. <no_carry><2><2>2 + 2 = 4. 4 is 0, 4. So we clear the carry. Prepend 4 to the answer. So far the answer has 1 digit: 4.
There is no carry. <no_carry><D><9>D + 9 = 16. 16 is 1, 6. So we set the carry to 1. Prepend 6 to the answer. So far the answer has 2 digits: 6, 4.
The carry is 1. <carry><7><2>7 + 2 + 1 = A. A is 0, A. So we clear the carry. Prepend A to the answer. So far the answer has 3 digits: A, 6, 4.
There is no carry. <no_carry><4><D>4 + D = 11. 11 is 1, 1. So we set the carry to 1. Prepend 1 to the answer. So far the answer has 4 digits: 1, A, 6, 4.
The carry is 1. Prepend 1 to the answer. So far the answer has 5 digits: 1, 1, A, 6, 4.
Therefore, the answer is 11A64.

G.2 Relational Reasoning

Prompt 19: Example outputs on CLUTRR. Errors are marked in red. Upper: CoT. Lower: CoT+HtT.

Context: The relations on the path from Christine to Nicole are mother, son, sister, grandmother.
Question: Nicole is Christine's what?
Answer:
For mother's son, we have mother's son is brother. So the relations are reduced to brother, sister, grandmother.
For brother's sister, we have brother's sister is sister. So the relations are reduced to sister, grandmother.
For sister's grandmother, we have sister's grandmother is great-grandmother. So the relations are reduced to great-grandmother.
Therefore, the answer is great-grandmother.

Context: The relations on the path from Christine to Nicole are mother, son, sister, grandmother.
Question: Nicole is Christine's what?
Answer:
For mother's son, we retrieve <mother><son>mother's son is brother. So the relations are reduced to brother, sister, grandmother.
For brother's sister, we retrieve <brother><sister>brother's sister is sister. So the relations are reduced to sister, grandmother.
For sister's grandmother, we retrieve <sister><grandmother>sister's grandmother is grandmother. So the relations are reduced to grandmother.
Therefore, the answer is grandmother.

Prompt 20: Example outputs on the textual version of CLUTRR. Errors are marked in red. Upper: CoT. Lower: CoT+HtT.

Context: The relations on the path from Sarah to Christine are husband, son, son, uncle, son, brother, brother, sister.
Question: Christine is Sarah's what?
Answer:
The path is quite complex, let's break it down:
For husband's son, we have husband's son is son. So the relations become son, son, uncle, son, brother, brother, sister.
For son's son, we have son's son is grandson. So the relations become grandson, uncle, son, brother, brother, sister.
For grandson's uncle, we have grandson's uncle is father or uncle. We choose father, given that uncle will introduce a sibling relationship, which is not consistent with the remaining relationships. So the relations

become father, son, brother, brother, sister.
For father's son, we have father's son is brother. So the relations become brother, brother, brother, sister.
For brother's brother, we have brother's brother is brother. So the relations become brother, brother, sister.
For brother's brother, we still have brother's brother is also brother. So we are left with brother, sister.
For brother's sister, we have brother's sister is sister. So the final relation is sister.
Therefore, the answer is sister.

Context: The relations on the path from Sarah to Christine are husband, son, son, uncle, son, brother, brother, sister.
Question: Christine is Sarah's what?
Answer:
For husband's son, we retrieve <husband><son>husband's son is son. So the relations are reduced to son, son, uncle, son, brother, brother, sister.
For son's son, we retrieve <son><son>son's son is grandson. So the relations are reduced to grandson, uncle, son, brother, brother, sister.
For grandson's uncle, we retrieve <grandson><uncle>grandson's uncle is son. So the relations are reduced to son, son, brother, brother, sister.
Reapplying son's son, we again have <son><son>son's son is grandson. So the relations are reduced to grandson, brother, brother, sister.
For grandson's brother, we retrieve <grandson><brother>grandson's brother is grandson. So the relations are reduced to grandson, brother, sister.
Reapplying grandson's brother, we have <grandson><brother>grandson's brother is grandson. So the relations are reduced to grandson, sister.
For grandson's sister, we retrieve <grandson><sister>grandson's sister is granddaughter. So the relations are reduced to granddaughter.
Therefore, the answer is granddaughter.