

2024

Reasoning or reciting? Exploring the capabilities and limitations of language models through counterfactual tasks

Z. Wu, L. Qiu, A. Ross, E. Akyürek, B. Chen, B. Wang, N. Kim, J. Andreas, Y. Kim. 2024. "Reasoning or Reciting? Exploring the Capabilities and Limitations of Language Models Through Counterfactual Tasks" Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pp.1819-1862. <https://doi.org/10.18653/v1/2024.naacl-long.102>
<https://hdl.handle.net/2144/50152>

"Downloaded from OpenBU. Boston University's institutional repository."

Reasoning or Reciting? Exploring the Capabilities and Limitations of Language Models Through Counterfactual Tasks

Zhaofeng Wu[Ⓜ] Linlu Qiu[Ⓜ] Alexis Ross[Ⓜ] Ekin Akyürek[Ⓜ] Boyuan Chen[Ⓜ]
 Bailin Wang[Ⓜ] Najoung Kim[Ⓜ] Jacob Andreas[Ⓜ] Yoon Kim[Ⓜ]
[Ⓜ]MIT [Ⓜ]Boston University
 zfw@csail.mit.edu

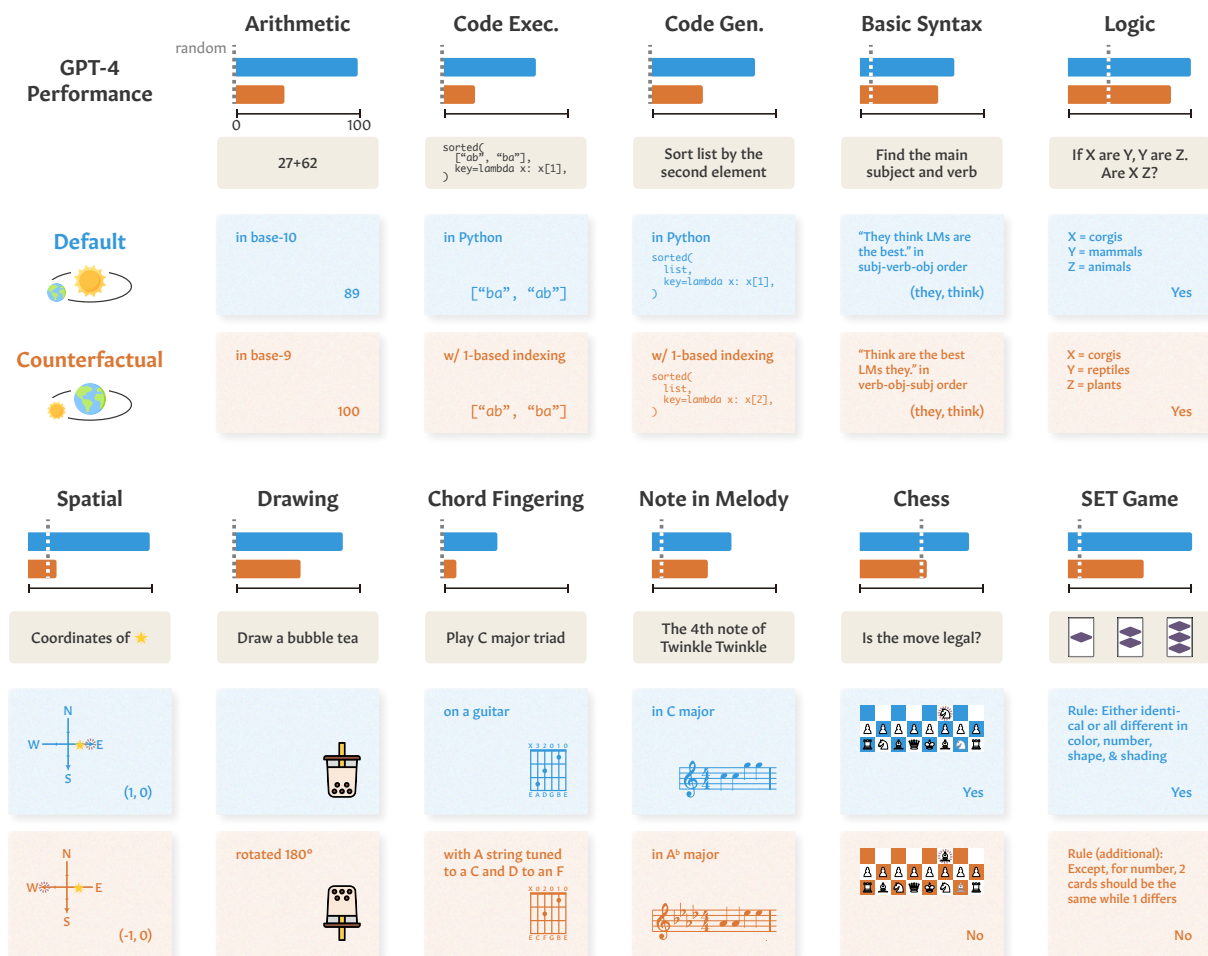


Figure 1: GPT-4’s performance on the default version of various tasks (blue) and counterfactual counterparts (orange). The shown results use 0-shot chain-of-thought prompting (§4; Kojima et al., 2023). GPT-4 consistently and substantially underperforms on counterfactual variants compared to default task instantiations.

Abstract

The impressive performance of recent language models across a wide range of tasks suggests that they possess a degree of abstract reasoning skills. Are these skills general and transferable, or specialized to specific tasks seen during pretraining? To disentangle these effects, we propose an evaluation framework based on “counterfactual” task variants that deviate from the default assumptions underlying standard tasks. Across a suite of 11 tasks, we observe nontrivial performance on the counterfactual

variants, but nevertheless find that performance substantially and consistently degrades compared to the default conditions. This suggests that while current LMs may possess abstract task-solving skills to an extent, they often also rely on narrow, non-transferable procedures for task-solving. These results motivate a more careful interpretation of language model performance that teases apart these aspects.

We release our code, all synthetically generated data, and LM interactions (prompts and responses) at <https://github.com/ZhaofengWu/counterfactual-evaluation>.

1 Introduction

The striking empirical successes of language models (LMs) suggest that next-word prediction at scale may be a viable approach for distilling the knowledge embedded in large-scale text corpora into general-purpose interactive agents. LMs obtain impressive results on various NLP benchmarks (OpenAI, 2023; Anil et al., 2023; Anthropic, 2023; *i.a.*) and display surprising abilities that suggest a non-trivial understanding of the world (Bubeck et al., 2023), even purportedly surpassing human performance on tasks that require nontrivial reasoning (Chowdhery et al., 2022; Hoffmann et al., 2022; Malinka et al., 2023; *i.a.*).

Ideally, we expect a general-purpose LM to be able to *generalize* not only to unseen instances of known tasks, but to new tasks. Humans, for example, can transfer their knowledge to new instances and also flexibly adapt to novel tasks (Singley and Anderson, 1989). While much past work has focused on instance-level generalization, we ask: to what extent does the performance of current LMs derive from their ability to deploy task-general reasoning skills (Li et al., 2022; Mishra et al., 2022), versus their ability to recognize and recall specific tasks seen frequently in pre-training?

We propose to measure such task-level generalizability by taking tasks on which LMs perform well, and altering the conditions under which these tasks are performed. The general reasoning procedure for these variants remains the same, but the input-output mapping functions differ. We call the new tasks *counterfactual tasks*, as they deviate from the default, generally assumed conditions for these tasks. For example, while arithmetic is by default performed in base-10, we consider a counterfactual task requiring additions in base-9. If models implement a general and transferable task-solving procedure, we expect comparable performance on counterfactual and default tasks; if they employ procedures tailored to default task conditions, we expect a drop in the counterfactual performance.

We design a suite of 11 counterfactual evaluation tasks, illustrated in Figure 1 (appendix), to measure an LM’s flexibility to adapt to new task variants across multiple categories and domains. In each, the original task under the default conditions and its counterfactual variants share the same reasoning procedure. We consider traditional tasks such as deductive reasoning, non-language tasks that are nonetheless commonly evaluated such as code gen-

eration, as well as non-standard tasks such as drawing and spatial reasoning. The latter extralinguistic tasks test whether LMs learn conceptual structures that mirror the structure of the non-linguistic world, which has been suggested by recent work (Patel and Pavlick, 2022; Bubeck et al., 2023; *i.a.*).

We evaluate GPT-4 (OpenAI, 2023), GPT-3.5, Claude (Anthropic, 2023), and PaLM-2 (Anil et al., 2023) on tasks under both the default and counterfactual conditions. We observe above-random counterfactual performance for most tasks, indicating some degree of task generalizability. However, performance on counterfactual task variants consistently and substantially degrades relative to the performance on the default settings. A systematic study of model behavior on default and counterfactual tasks (§5) reveals that while chain-of-thought prompting and few-shot demonstrations generally improve performance, they do not close the default-counterfactual gap. We further find that counterfactual performance is affected by the commonness of the counterfactual conditions, but correlates with the default task performance. These results collectively suggest that LMs’ abilities on these tasks are supported at least in part by non-transferable, default-condition-specific behaviors rather than generalizable reasoning skills.

2 Counterfactual Tasks

We informally conceptualize a task as a function $f_w : X \rightarrow Y$ that maps an **input** $x \in X$ under a **world model** $w \in W$ to an **output** $y \in Y$. World models encapsulate the conditions for the function evaluation. For example, in arithmetic, w could represent the set of conditions required for an arithmetic operation, such as the number base. We refer to the set of assumed default conditions, including but not limited to the base’s being 10, as the **default world**, or w^{default} . Intuitively, for any task, w^{default} corresponds to the set of conditions underlying most task instances in pretraining corpora.

Traditional machine learning evaluations assess how closely a model’s learned hypothesis h estimates f_w by independently sampling training and test sets from the population distribution and only exposing the model to the former for learning h . However, in datasets of web text, these evaluations are subject to potential data contamination issues (Brown et al., 2020; Magar and Schwartz, 2022; *i.a.*). These issues may be more severe in recent LMs: the ever-growing pretraining datasets

potentially expose the models to more evaluation instances, and recent LMs with increasing sizes are more capable of memorizing these instances.

We hence consider another dimension of generalization: to new task variants in **counterfactual worlds** w^{cf} , instead of new inputs x . This allows us to measure the extent to which a model’s $f_{w^{\text{default}}}$ performance is specific to w^{default} or attributable to a general implementation of the task f . For arithmetic, w^{cf} could be one that was the same as w^{default} but assumed a base other than base-10. We expect a model with general arithmetic ability to perform similarly in other bases. We stress that our goal is not to find counterfactual world models outside human experience. Base-9 addition, for example, is not a novel concept. Nor do we aim that the counterfactual world models are unobserved in pretraining. Instead, counterfactuals are simply variations on the *default* conditions for a task.

We assess an LM’s task performance with 0-shot prompting. We specify the task f , the test instance x , and the world model w in a prompt, and parse the LM’s output. We denote the LM’s implementation of f_w for a given instance x to be,

$$h(f, w, x) = \arg \max_{y'} P_{\text{LM}}(y' \mid \text{prompt}_f(f, x), \text{prompt}_w(w)),$$

where the $\arg \max$ is computed with approximate decoding and prompt_f and prompt_w describe tasks and world models respectively. For each task, we devise one or more w^{cf} that deviate from the default world conditions and compare $h(f, w^{\text{default}}, x)$ and $h(f, w^{\text{cf}}, x)$. If we control $f_w(x)$ to be similarly hard between w^{default} and w^{cf} , we can attribute the performance difference to an LM overfitting to the default instantiation of the task.

Counterfactual Comprehension Check. One potential confounder is that an LM may fail at a particular counterfactual task by failing to understand the prompt component that specifies the counterfactual conditions $\text{prompt}_w(w^{\text{cf}})$, and keep reasoning in w^{default} . This failure mode would not reflect a lack of generalizable task abilities. We control for this with task-specific **counterfactual comprehension checks (CCCs)** that test an LM’s surface understanding of the specified counterfactual world. For each $(w^{\text{default}}, w^{\text{cf}})$ pair, we introduce another control task g_w with input x' and output y' that is much simpler than f_w but still allows for the discrimination of w^{default} from w^{cf} (i.e., $g_{w^{\text{cf}}}(x') \neq g_{w^{\text{default}}}(x')$). A high performance of

$P_{\text{LM}}(y' \mid \text{prompt}_g(g, x'), \text{prompt}_w(w^{\text{cf}}))$ would indicate that prompt_w is effective at making the LM perform a task in w^{cf} . In the arithmetic example, for a base-9 counterfactual world, we use the same $\text{prompt}_w(\text{base-9})$ to specify the counterfactual world, and check that it facilitates an understanding of $w = \text{base-9}$ by asking what the next integer after x' is. If, for example, it consistently carries over digits greater than 8 and does not carry over otherwise, this would show the effectiveness of $\text{prompt}_w(\text{base-9})$. Our CCC designs are heuristic: as with control tasks in the probing literature (Hewitt and Liang, 2019), we rely on intuition to craft a g_w that is “simpler” than f_w .¹

3 Tasks

In this section, we give a quick overview of the tasks we consider. See §A for the full description of each task and §C for all the prompts used.

Arithmetic. Modern LMs have been shown to possess basic numerical reasoning abilities (Lewkowycz et al., 2022), with Brown et al. (2020) even reporting near-perfect GPT-3 accuracy for two-digit additions. On the other hand, Razeghi et al. (2022) find that LMs perform significantly better with numbers that occur more frequently in the pretraining data, and Li et al. (2023b) show that symbol replacement affects LMs’ mathematical ability; both findings point to overfitting and memorization effects. We consider the same two-digit addition task, the simplest arithmetic task in Brown et al. (2020), but inspect a model’s accuracy in different bases. We use base-8, 9, 11, and 16 as the counterfactual setup, which were chosen to control for task difficulty and also to test for how relatively uncommon (9 & 11) and common (8 & 16) bases affect performance (see §5.1). The CCC evaluates the successor relation under each base.

Programming. The inclusion of code corpora in LM pretraining (Gao et al., 2021; Chowdhery et al., 2022; Touvron et al., 2023; *i.a.*) allows LMs to possess coding capabilities, sometimes even at state-of-the-art level (Sobania et al., 2023). Nevertheless, Miceli-Barone et al. (2023) show that GPT-3 and related models are fragile under identifier swaps in programs, suggesting only a shallow understanding of code. Here, we inspect an LM’s programming

¹For some tasks, it is more natural to query about the task and CCC jointly in the same prompt, i.e., $P_{\text{LM}}(y, y' \mid \text{prompt}_f(f, x), \text{prompt}_g(g, x'), \text{prompt}_w(w^{\text{cf}}))$.

ability through a deeper counterfactual perturbation: contrary to the traditional 0-based indexing in Python, we instruct the LM to evaluate or generate Python programs under 1-based indexing using the HumanEval dataset (Chen et al., 2021). 1-based indexing is a common assumption for other programming languages such as MATLAB and R and hence provides a fair testbed. The CCC here involves the same program execution task but on much simpler inputs, such as simple list indexing.

Basic Syntactic Reasoning. Mahowald et al. (2023) distinguish between two types of LM capabilities: *formal competence* that encompasses the knowledge of language, and *functional competence* which involves using language, potentially combined with extralinguistic capacities, to interact with the world. While our other tasks assess functional competence, we include an evaluation on formal competence. We revisit the attested syntactic knowledge of LMs (Yu et al., 2020; Linzen and Baroni, 2021; Belinkov, 2022; *i.a.*) with a meta-linguistic task (Beguš et al., 2023; Hu and Levy, 2023; *i.a.*): evaluating LMs in synthetic versions of English with different word orders from English’s subject-verb-object (SVO) ordering, obtained from manipulating dependency trees (Ravfogel et al., 2019). We ask the LM to identify the main subject and the main verb of a sentence under both the original and counterfactual orders. The CCC requires the model to revert simple reordered sentences to the original SVO ordering, equivalent to identifying these elements in a sentence.

Natural Language Reasoning with First-Order Logic. We next consider a deductive reasoning task, again based on natural language. Deductive logical reasoning is a prerequisite ability for many complex tasks (McCarthy, 1959) and has received much recent focus (Clark et al., 2020; Tafjord et al., 2021; Saparov and Mitchell, 2022; Saparov and He, 2023; *i.a.*). Nevertheless, LMs struggle with reasoning with premises that are inconsistent with common sense (Dasgupta et al., 2022; Yu et al., 2023; Tang et al., 2023). Here, we undertake a similar study from the perspective of counterfactual analysis to disentangle the effect of common sense from a model’s actual logical reasoning capability. Following prior work, we ask LMs if a series of premises entails a conclusion. We use the FOLIO dataset (Han et al., 2022), whose premises are mostly consistent with common sense. For the counterfactual version we manually rewrite

them to violate common sense, and study if LM performance is affected by the truthfulness of the premises under which they operate. The CCC directly asks the model if the original or post-rewrite premise is true, when presented both as options.

Spatial Reasoning. A major debate around LMs is whether grounded representations of meaning can be learned from form alone (Bender and Koller, 2020; Piantadosi and Hill, 2022). Studies have shown that LMs can learn meaningful representations of certain concepts through text-only training, with Patel and Pavlick (2022) finding that LMs learn representations of spatial relations and cardinal directions that can be aligned to grounded conceptual spaces. We similarly investigate an understanding of cardinal directions, but instead of evaluating whether a model can *induce* structured conceptual spaces, we ask if it can *apply* conceptual spaces to reason about object locations. Specifically, we ask an LM for the coordinates of objects whose positions are described using cardinal directions, under a conventional 2D coordinate system (e.g., where east corresponds to $(1, 0)$) versus coordinate systems with swapped, rotated, and randomly permuted axes. We expect a robust representation to not be sensitive to such transformations. The CCC involves asking the model to directly output the counterfactual cardinal directions.

Drawing. LMs have been shown to structure their representations of perceptual concepts such as size and color in a way that credibly mirrors the physical world (Abdou et al., 2021; Patel and Pavlick, 2022; Zhang et al., 2020; Ilharco et al., 2021; *i.a.*). Recent LMs can even directly generate plausible drawings of objects using code such as TikZ and SVG (Bubeck et al., 2023; Zhang et al., 2023b). We evaluate the visual understanding of LMs by asking them to generate code for drawing various objects programmatically. Inspired by psychological studies that show humans’ ability to rotate mental representations of objects (Shepard and Metzler, 1971; Vandenberg and Kuse, 1978), we ask the LM to generate code that draws the same object, but rotated or vertically flipped as our counterfactual settings. For the CCC, we ask the model to draw a straight line at the top of the canvas in addition to the object; a flipped/rotated line thus signifies an understanding of the transformations.

Music. Recent work has shown the potential of large-scale models for music infilling (Huang et al., 2019a,b) and generation (Agostinelli et al., 2023;

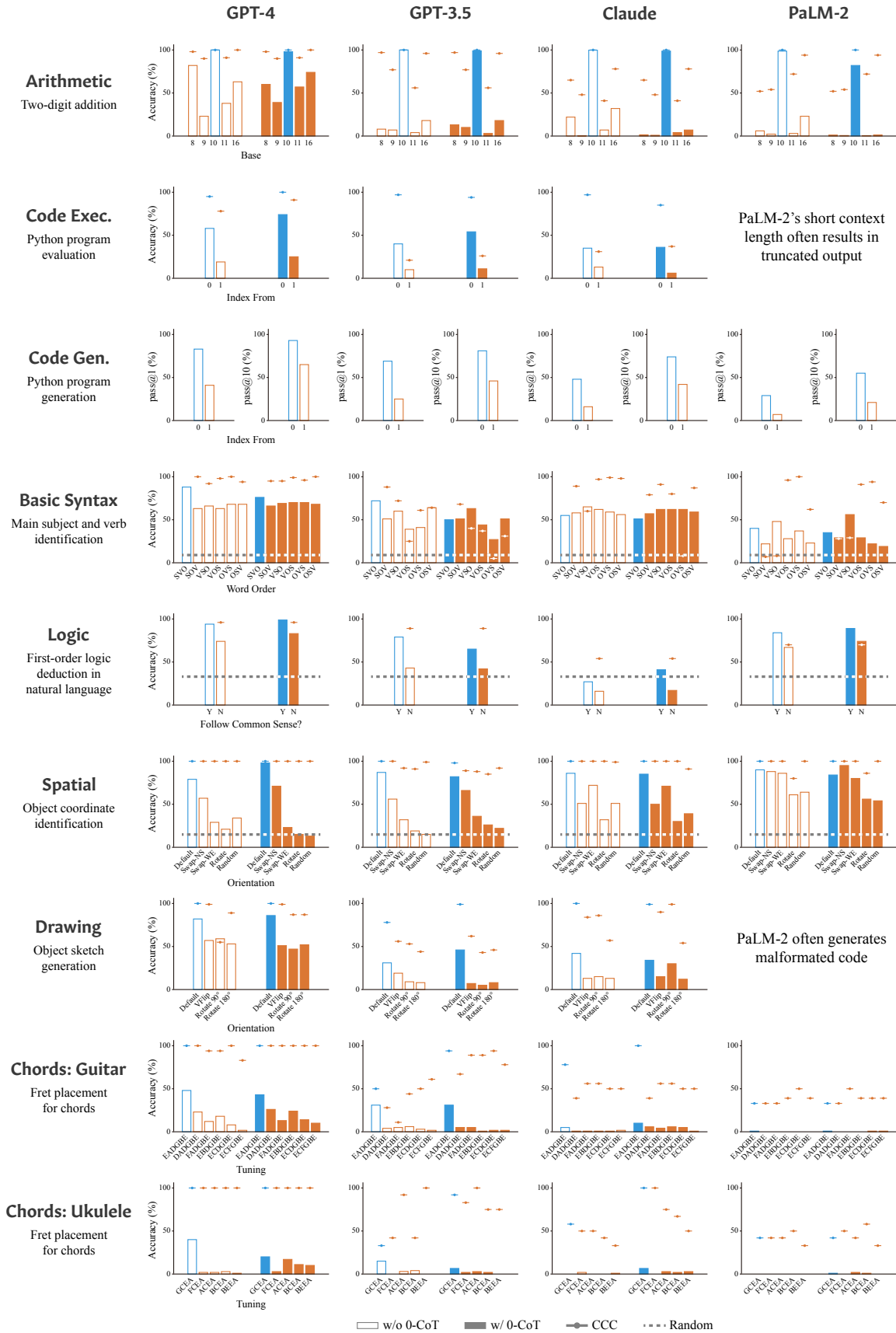


Figure 2: Main results. The blue and orange bars represent the default and counterfactual conditions respectively, either with or without 0-shot chain-of-thought (0-CoT) (except code generation; see §A.2). CCC is the counterfactual comprehension check (§2), but when applicable, we report it for the default setting too. Random performance is marked whenever nontrivial. Counterfactual performance is consistently lower than the default task performance, while CCC is usually high. §D reports numeric results.

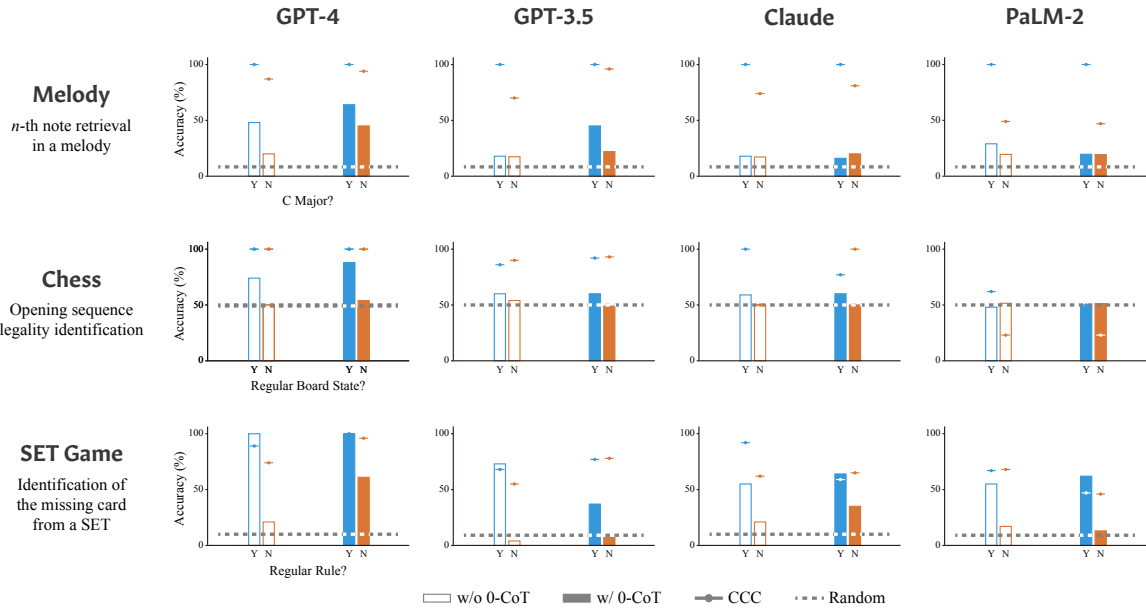


Figure 3: Main results (continued). The blue and orange bars represent the default and counterfactual conditions respectively, either with or without 0-shot chain-of-thought (0-CoT). CCC is the counterfactual comprehension check (§2), but when applicable, we report it for the default setting too. Counterfactual performance is consistently lower than the default task performance, while CCC is usually high. §D reports numeric results.

Copet et al., 2023; Ren et al., 2020). Bubeck et al. (2023) show that even a text-only LM exhibits some musical abilities. We investigate the extent of LMs’ musical abilities through two tasks.

In the *chord placement* task, we evaluate whether LMs can provide the correct chord fret placements for string instruments with standard or altered string tunings. The altered tunings, known as *scordatura*, are typical in music to evoke specific effects. In the counterfactual setting, we instruct LMs to provide fret placements for a special guitar/ukulele where one or two of the strings are altered. To check whether the model has understood the tunings, we ask for the first three notes on each string (including open string) as the CCC.

In the *note retrieval* task, we evaluate whether LMs can retrieve notes from famous melodies. The process of re-writing melodies in different keys, referred to as “transposition,” is common in music. We evaluate LMs’ musical abilities under transpositions by prompting them to retrieve the n -th note in a melody in either its canonical key or a different key. We ask the LMs to retrieve the n -th note of the scale of the given key as the CCC.

Chess. Chess-playing has long been regarded as a testbed for AI (Silver et al., 2017; Tomasev et al., 2020), and modern LMs have exhibited abilities that imply an understanding of chess rules (Srivastava et al., 2023; Du et al., 2023). We test this understanding by asking for the legality of a 4-move

opening. In the counterfactual setting, we swap the initial positions of knights and bishops—a setup present in a real-world chess variant “Chess 960”—and similarly ask LMs for opening legality under this new starting configuration. The CCC asks for the starting positions of the knights and bishops.

SET Game. SET is a popular card game where each card has 4 attributes (color, shape, shading, and number) with 3 different values for each attribute. A player finds a SET of 3 cards on a board whose values for each attribute are **either all the same or all unique**. This game has been thoroughly studied in computer science, from the perspective of coding theory and combinatorics (Davis and Maclagan, 2003), linear algebra (Coleman and Hartshorn, 2012), and complexity theory (Chaudhuri et al., 2003). We suspect this popularity makes it susceptible to overfitting by LMs and investigate this possibility. We ask the LM to identify the card on a board that completes a 3-card SET with two given cards. In the counterfactual setup, we invert the rule for the *number* attribute, requiring its values to be **neither all the same nor all unique**. For the CCC, we ask the model for the validity of a SET under the original and the counterfactual rules.

4 Results

We evaluate GPT-4 (gpt-4-0314; OpenAI, 2023), GPT-3.5 (gpt-3.5-turbo-0301), Claude (claude-v1.3; Anthropic, 2023), and PaLM-2

(text-bison-001; Anil et al., 2023).² We experiment both with and without encouraging step-by-step reasoning with the phrase “Let’s think step by step.” (Kojima et al., 2023; Reynolds and McDonnell, 2021). We refer to this step-by-step setup as zero-shot chain-of-thought prompting (**0-CoT**; Kojima et al., 2023). We include all prompts in §C.

Figures 2 and 3 show our results. The numeric version are in §D. LMs consistently and substantially perform worse on the counterfactual variants, both with and without 0-shot CoT. For most cases, LMs exhibit an above-random counterfactual performance, suggesting some degree of the targeted ability. However, when the CCC accuracy is high, as is usually the case for GPT-4 and select settings for other models too, the gaps in default vs. counterfactual performance demonstrate limitations in their abstract capacity for the target task. When the CCC accuracy is lower, the failure of counterfactual world comprehension confounds this conclusion, but often the gaps are so large (sometimes between near-perfect and near-zero, e.g. for arithmetic) that they are nonetheless strongly indicative of non-transferable, default-condition-specific task implementations. The fact that the LMs sometimes do not evaluate the CCC well under counterfactual conditions, but can do so under the default conditions (for arithmetic, programming, drawing, etc.) itself also points to overfitting to the latter.

5 Analysis

We investigate how a variety of factors affect the performance trends in §4. Unless otherwise specified, we only consider GPT-4 with 0-shot CoT as this is generally the best-performing setup. We provide additional analyses in §B.

5.1 “Commonness” of Counterfactuals

Our counterfactual worlds are not designed to be completely alien to the LMs but only less common than the assumed default case. The counterfactualness of task variants is hence only relative, and here we take a more nuanced look at how the commonness of these counterfactual conditions affects the default-counterfactual performance gap. For arithmetic, all models perform better in bases 8 and 16, likely due to their relative abundance compared to bases 9 and 11. In spatial reasoning, the smallest counterfactual performance degradation is usually from when the north and south directions

²The largest PaLM-2 model is not publicly accessible, and we can only test the second-largest version.

are swapped, potentially because some programming libraries use an inverted y -axis (matplotlib, ggplot, D3, etc.; see §A.5). For chord fingering, the common alternative drop-D tuning of guitars (DADGBE) leads to the highest counterfactual performance for GPT-4. These correlations between the counterfactual performance and the commonness of these worlds paint a more fine-grained picture than a binary real vs. counterfactual distinction and point to a memorization-like effect where LMs perform better under more common conditions.

5.2 Proximity between Default and Counterfactual Conditions

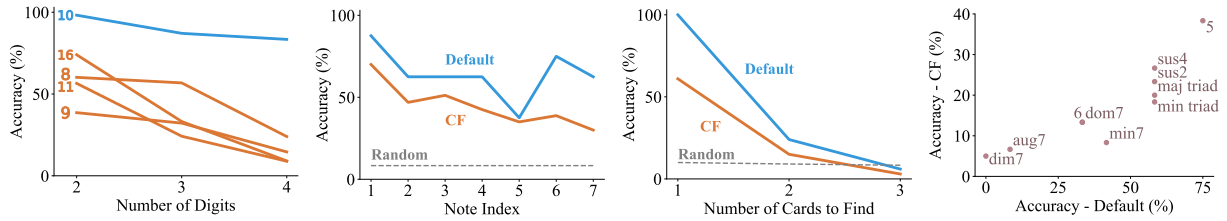
Another axis along which the counterfactual worlds differ is in their proximity to the default conditions. For example, for the different arithmetic bases, bases 9 and 11 are *closer* to base 10, but *less common* than bases 8 and 16. While the default-counterfactual gap is most affected by commonness for the arithmetic task, for the guitar and ukulele tunings (other than the drop-D tuning), the LM performance generally decreases monotonically with increasing distance from the original tunings.

The FOLIO dataset (Han et al., 2022) enables another analysis of how proximity to the default conditions affects the model performance, *without* counterfactual perturbations. This dataset was constructed to mostly follow common sense, with premises and conclusions that are deemed true in the real world. However, this is not always the case, with premises such as “John can make meals which are popular at the party,” whose factuality cannot be determined alone.

In §B.1, we use this feature to show that a LM’s reasoning ability is inversely related to the distance between the (LM-believed) real world and the world state described by the premises (occasionally counterfactual to the LM), by training a predictive model given features approximating this distance. Overall, these results show that LMs tend to perform better on task variants that are closer to the default instantiation of a task.

5.3 Default vs. Counterfactual Performance

Recalling our formalization $h_{\text{LM}}(f, w, x)$ in §2, §5.1 analyzed how the commonness of w affects the observed patterns. Here, we explore how the counterfactual performance correlates with the default task performance when the other three elements vary: the task f , the input x , and the LM.



(a) Addition accuracy with varying numbers of digits in the operands. (b) Melody note recall accuracy broken down by the index of the note. (c) SET identification accuracy when needing to find different numbers of cards in a SET. (d) Fret placement accuracy by chord type. The y -axis averages over all altered tunings.

Figure 4: Investigating the relationship between the default task performance and counterfactual performance, broken down by different factors. Only GPT-4 with 0-shot CoT results are shown. There is a consistent default-counterfactual correlation across task variants when varying different factors.

We first consider different task variants with various difficulties. For arithmetic, beyond 2-digit addition, we also measure GPT-4’s 3- and 4-digit addition performance (Figure 4a). For note retrieval from melodies, we use the index of the inquired note as the proxy for difficulty (Figure 4b). For SET, while our original task shows two cards and asks a model to find the missing one from a 3-card SET, we change the task to instead show one or none of the cards in a SET, while still requiring the model to identify the SET (Figure 4c). For all these task variants, we see a strong correlation between the original and counterfactual world performance.

We also see this effect when breaking down results by test instances x . In Figure 4d, we separate the chord types, and observe that the default and counterfactual task performance correlates. Similarly, for most tasks, stronger models under default conditions are also stronger under counterfactual conditions, and vice versa. These correlations indicate that the default task performance can be a good indicator of its counterfactual performance, and hence its utility should not be discounted.

Occasionally, this default-counterfactual correlation is reversed. In the spatial reasoning task, for example, GPT-4 has the best default-condition accuracy with 0-shot CoT, but it also degrades the most facing counterfactuals. PaLM-2 performs worse under default conditions, but is the most robust to counterfactual perturbations. McKenzie et al. (2023), who found a similar trend with respect to pretraining FLOPs and termed it “inverse scaling,” provided a memorization-based explanation: they observed that when a task contradicts with pretraining texts, similar to how our counterfactuals deviate from the default conditions in pretraining, larger LMs tend to rely on the pretraining text and, in turn, fail at the contradictory task.

5.4 The Effect of Prompts

0-shot CoT. Consistent with prior findings (Chen et al., 2022; Dasgupta et al., 2022; *i.a.*), we generally observe 0-shot CoT to be helpful for most cases. There are, however, exceptions, for example on PaLM-2’s base-10 and 16 addition, and GPT-4’s chord-playing performance. This may be due to a model pragmatically inferring that a task is more difficult than it actually is when explicitly asked to “think step by step”, and this “overthinking” on simple tasks could lead to mistakes (Kojima et al., 2023). It is also possible that these are due to memorization: the model could have memorized the specific input-output mapping of a task without understanding how to derive the output from the input, and when explicitly asked to spell out that process, it makes more errors (Zhang et al., 2023a).

Few-shot Demonstrations. We study if additional demonstration examples using in-context learning (Brown et al., 2020) bridges the default-counterfactual gap. For the arithmetic task, we construct few-shot CoT prompts (Nye et al., 2021; Wei et al., 2022) and prepend up to 16 samples. As shown in Figure 5 (appendix), while the gap is reduced, it is still substantial for bases 9, 11, and 16, and plateauing towards the 16-shot setting. Therefore, the default-counterfactual gap is unlikely to be eliminated simply with more demonstrations.

6 Discussion

Do humans also perform worse with unfamiliar counterfactual conditions? It is possible that humans may have lower performance under the counterfactual conditions with a fixed time budget, but not necessarily when given ample time to reason and revise. Analogous to the classic competence/performance distinction in linguistics (Chomsky, 1965, §1.1), we hypothesize that humans have the *competence* to generalize to new task condi-

tions, even though it may sometimes require sufficient execution budget to realize it as robust *performance*.³ In fact, there is increasing evidence from cognitive science that human reasoning is scaffolded by rich causal models of the world (Pearl, 1988; Lake et al., 2017), and that humans can intervene on these models to perform rapid and flexible counterfactual simulations (Lagnado et al., 2013; Gerstenberg et al., 2017, 2021). However, stepping back, replicating or modeling human intelligence need not be a main goal of LMs in the first place, and human behavior is largely orthogonal to the desiderata we set for these models.

Is task-specific reasoning bad? It is not necessarily bad when solving familiar tasks, but an ideal system should also possess general reasoning abilities that, when prompted, can be used to generalize to novel situations. Our point is that memorization is an often-overlooked confounding factor in interpreting LMs’ reasoning abilities.

Why do we care about counterfactual worlds? Isn’t the default-task model still useful? It is certainly useful. However, many of the counterfactual worlds that we investigate are not very distant so that model performance under them still bears utility. Moreover, we are mostly only interested in the counterfactual tasks insofar as performance on these tasks can serve as a measurable proxy for the generalizability of these models and their underlying reasoning capabilities.

Aren’t these trends trivial? The default task variant is likely the most frequent in pretraining, so of course an LM performs better under it. Indeed, our results parallel the classic train-test gap in machine learning. However, an ideal learner with the right inductive biases should be able to structure their internal parameters and representations to implement general-purpose abstractions (e.g., the concept of addition), and use these abstractions to generalize to counterfactual conditions, analogous to physicists using mathematics to make predictions about universes that are substantially different from our own, or more generally to humans who can generalize to new stimuli (Lagnado et al., 2013; Gerstenberg et al., 2017, 2021).

Can some more carefully designed prompts eliminate the default-counterfactual gap? We can never tractably rule out this possibility. Never-

theless, given the consistency of the gap across the different tasks (which use different prompts) and the setting of 0-shot CoT, we believe that a prompt that *completely* bridges the default-counterfactual gap is unlikely. Our in-context learning experiment (§5.4) further shows that while this gap could be reduced by more informative prompts, it cannot be not fully removed.

7 Related Work

Causal Analysis. Our counterfactual perturbations can be informally viewed as interventions in causal inference (Pearl, 2009). This relationship has been explored in machine learning and NLP for commonsense reasoning (Kıcıman et al., 2023), interpretability (Elazar et al., 2021; Geiger et al., 2021, 2022), spurious correlation detection (Veitch et al., 2021; Eisenstein, 2022), and fairness (Kusner et al., 2017; Nabi and Shpitser, 2018). Under this perspective, the degradation from counterfactuals can be viewed as a failure to robustly learn the causal effects of world states.

Counterfactual Evaluation. “Counterfactuals” is an informally-used term in NLP and has been used to refer to different types of perturbations. One line of work concerns counterfactuals to a certain event or situation that is still licensed in a default world model (Qin et al., 2019, 2020; Yang et al., 2020; Frohberg and Binder, 2022; *i.a.*), in contrast to our counterfactual world states that deviate from the default. Another body of work examines the robustness of model predictions using counterfactual data (Kaushik et al., 2020, 2021; Gardner et al., 2020). Closer to us, Li et al. (2023a) showed that while LMs seem to be able to perform some reasoning in counterfactual worlds, this is largely affected by superficial lexical cues.

8 Conclusion

Through our counterfactual evaluation on 11 tasks, we identified consistent and substantial degradation of LM performance under counterfactual conditions. We attribute this gap to overfitting to the default task variants, and thus encourage future LM analyses to explicitly consider abstract task ability as detached from observed task performance. Furthermore, insofar as this degradation is a result of the LMs’ being trained only on surface form text, it would also be interesting future work to see if more grounded LMs (grounded in the “real” world, or some semantic representation, etc.) are more robust to task variations.

³It is arguable if our evaluation setting provides sufficient execution budget (Lampinen, 2023). Our in-context learning experiment (§5.4) may be thought of as increasing this budget, and yet the default-counterfactual gap is still sizeable there.

Limitations

Despite our attempt to devise novel counterfactual conditions to gauge an LM’s “true” reasoning ability, it may not be precisely reflected by the counterfactual performance due to several factors.

Underestimation

For our main evaluations, we aim to construct counterfactual tasks that have the same difficulty as the default variants so that task difficulty does not confound our comparisons. This is not always possible—in fact, an objective difficulty measure may not even exist. One could, for example, argue that base-11 addition is harder than base-10 because it requires reasoning with one additional digit, or base-9 is harder than base-10 because on average the sums would consist of more digits.

Retrieving notes in melodies in different keys faces a similar issue. We expect similar retrieval difficulty under different keys if the model recalls a melody as a series of abstract relations in a scale and directly maps them onto notes in a target key. However, an alternative strategy would be to first retrieve the note in a canonical key and then *transpose* it to the desired uncommon key. This 2-step process is a natural one that is often employed by musicians. And with this strategy, the counterfactual task consists of 2 steps and is harder than (and requires first) completing the 1-step original task. The counterfactual setup thus introduces a confounder: low performance may be driven by the increased difficulty of the counterfactual task, rather than overfitting to melodies in their canonical keys, *if* models are employing two-step strategy. However, since both strategies are available to models and we do not prompt them to use a particular one, reliance on this two-step strategy may itself be indicative of overfitting to the original canonical keys.

Overestimation

We can never be certain of how rarely particular counterfactual conditions are encountered during pretraining. It is quite likely that there is text online that, for example, draws rotated versions of various objects used in our study. Consequently, the effect of overfitting could also manifest in our counterfactual conditions, and the default-counterfactual gap could actually be larger for some genuinely unseen conditions.

We also distinguish between two types of counterfactual perturbations. One type fundamentally affects the operation of the world model and necessitates an understanding of the counterfactual world to perform the task in it (e.g., arithmetic base or 1-based indexing⁴). On the other hand, some perturbations are more superficial and may admit a shortcut where the model first figures out a simple mapping of the input back to the default conditions and performs the task (potentially leveraging instance-level memorization) under those. In some of our tasks, this mapping may be simple, such as the word replacements in the natural language logical reasoning task⁵ and the transformation functions for the drawing task (see §A.6), which could potentially be exploited by the models. We explicitly disallow this in our prompt for the drawing task (Table 7) but did not identify a good way to forbid this for logical reasoning, potentially accounting for its generally high counterfactual performance.

Finally, we reiterate from §4 that a non-perfect CCC accuracy does not allow us to perfectly tease apart counterfactual performance and a failure of counterfactual condition comprehension. But often the default-counterfactual gap is so prominent that it is still strongly suggestive of overfitting to the default conditions. Also, recall from §2 that the CCC itself is also a nontrivial task. For ThonPy, for example, the CCC also involves program evaluation, albeit with simpler statements that involve less reasoning, such as `print("qrstu"[4])`. We do not see an easy way to introduce ThonPy CCC that is entirely disentangled from program evaluation. This conflation would result in the CCC accuracy’s being lower than what would reflect the model’s understanding of the counterfactual conditions.

Acknowledgments

We thank, alphabetically, Alex Gu, Alisa Liu, Belinda Li, Chenghao Yang, Han Guo, Hao Peng,

⁴It may be tempting to consider a simple replacement strategy `[i]→[i-1]` to map back to 0-based indexing. But this does not work for the dictionary type. There are other complications; see Table 2.

⁵To be more concrete, imagine that a model memorizes an instance with nine premises on dogs involving complex logical relationships, and that it entails a given conclusion. For the counterfactual instance, we replace the word “dogs” with another object, say “headphones,” to make the premises no longer factually true. Instead of performing the reasoning over premises with headphones such as how they are, counterfactually, the cutest creatures, a model could identify the mapping “dogs” → “headphones”, revert it (i.e., replace all “headphones” back to “dogs”), and perform the task under the default common-sense-complying conditions.

Heyun Li, Jesse Dodge, Pratyusha Sharma, Tiwa Eisape, and Yizhong Wang for helpful discussions and feedback for this work. We are also grateful to Simeng Han for providing us with an updated version of the FOLIO dataset. Our drawing evaluation would not have been possible without our annotators Alex Hu, Ananya Harsh Jha, Belinda Li, Erjia Cao, Ha-na Park, Huirong Wen, Jiangjie Chen, Kabir Swain, Ka Wai Chan, Lucy Li, Simran Swain, Tejas Srinivasan, Tianyu Liu, Yue Bai, Yutaro Yamada, and Ziwei Wei. Zhaofeng would like to thank Jiamin Zhang for the guitar lessons, which were short but helpful for the relevant components of this paper. Figure 1 uses icons from flaticon.com. This study was supported by funds from the MIT-IBM Watson AI Lab, the MIT Quest for Intelligence, and the National Science Foundation under grants IIS-2212310 and IIS-2238240.

References

- Mostafa Abdou, Artur Kulmizev, Daniel Hershcovich, Stella Frank, Ellie Pavlick, and Anders Søgaard. 2021. [Can language models encode perceptual structure without grounding? a case study in color](#). In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 109–132. Online. Association for Computational Linguistics.
- Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. 2023. [MusicLM: Generating music from text](#).
- Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. [PaLM 2 technical report](#).
- Anthropic. 2023. [Introducing Claude](#).
- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022. [Training a helpful and harmless assistant with reinforcement learning from human feedback](#).
- Gašper Beguš, Maksymilian Dąbkowski, and Ryan Rhodes. 2023. [Large linguistic models: Analyzing theoretical linguistic abilities of LLMs](#). *ArXiv preprint*, abs/2305.00948.
- Yonatan Belinkov. 2022. [Probing classifiers: Promises, shortcomings, and advances](#). *Computational Linguistics*, 48(1):207–219.
- Emily M. Bender and Alexander Koller. 2020. [Climbing towards NLU: On meaning, form, and understanding in the age of data](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

- Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrike, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. [Sparks of artificial general intelligence: Early experiments with GPT-4](#).
- Kamalika Chaudhuri, Brighten Godfrey, and David Ratajczak. 2003. On the complexity of the game of SET.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. [Evaluating large language models trained on code](#).
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. [Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks](#). *ArXiv preprint*, abs/2211.12588.
- Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. The MIT Press, Cambridge.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. [PaLM: Scaling language modeling with pathways](#).
- Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. [Transformers as soft reasoners over language](#). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3882–3890. ijcai.org.
- Ben Coleman and Kevin Hartshorn. 2012. [Game, set, math](#). *Mathematics Magazine*, 85(2):83–96.
- Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. 2023. [Simple and controllable music generation](#).
- Ishita Dasgupta, Andrew K. Lampinen, Stephanie C. Y. Chan, Antonia Creswell, Dharshan Kumaran, James L. McClelland, and Felix Hill. 2022. [Language models show human-like content effects on reasoning](#).
- Benjamin Lent Davis and Diane Maclagan. 2003. [The card game SET](#). *The Mathematical Intelligencer*, 25:33–40.
- Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. [Improving factuality and reasoning in language models through multi-agent debate](#). *ArXiv preprint*, abs/2305.14325.
- Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Number 57 in Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton, Florida, USA.
- Jacob Eisenstein. 2022. [Informativeness and invariance: Two perspectives on spurious correlations in natural language](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4326–4331, Seattle, United States. Association for Computational Linguistics.
- Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2021. [Amnesic probing: Behavioral explanation with amnesic counterfactuals](#). *Transactions of the Association for Computational Linguistics*, 9:160–175.
- Jörg Froberg and Frank Binder. 2022. [CRASS: A novel data set and benchmark to test counterfactual reasoning of large language models](#). In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2126–2140, Marseille, France. European Language Resources Association.
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2021. [The Pile: An 800GB dataset of diverse text for language modeling](#). *ArXiv preprint*, abs/2101.00027.
- Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco,

- Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. [Evaluating models' local decision boundaries via contrast sets](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323, Online. Association for Computational Linguistics.
- Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. [Causal abstractions of neural networks](#). In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 9574–9586.
- Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah D. Goodman, and Christopher Potts. 2022. [Inducing causal structure for interpretable neural networks](#). In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 7324–7338. PMLR.
- Tobias Gerstenberg, Noah D. Goodman, David A. Lagnado, and Joshua B. Tenenbaum. 2021. A counterfactual simulation model of causal judgments for physical events. *Psychological review*.
- Tobias Gerstenberg, Matthew Peterson, Noah D. Goodman, David A. Lagnado, and Joshua B. Tenenbaum. 2017. Eye-tracking causality. *Psychological Science*, 28:1731 – 1744.
- Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, David Peng, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Shafiq Joty, Alexander R. Fabri, Wojciech Kryscinski, Xi Victoria Lin, Caiming Xiong, and Dragomir Radev. 2022. [FOLIO: Natural language reasoning with first-order logic](#).
- John Hewitt and Percy Liang. 2019. [Designing and interpreting probes with control tasks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. [Training compute-optimal large language models](#).
- Jennifer Hu and Roger Levy. 2023. [Prompt-based methods may underestimate large language models' linguistic generalizations](#). *ArXiv preprint, abs/2305.13264*.
- Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron C. Courville, and Douglas Eck. 2019a. [Counterpoint by convolution](#). *ArXiv preprint, abs/1903.07227*.
- Cheng-Zhi Anna Huang, Curtis Hawthorne, Adam Roberts, Monica Dinulescu, James Wexler, Leon Hong, and Jacob Howcroft. 2019b. [The bach doodle: Approachable music composition with machine learning at scale](#).
- Gabriel Ilharco, Rowan Zellers, Ali Farhadi, and Hananeh Hajishirzi. 2021. [Probing contextual language models for common ground with visual representations](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5367–5377, Online. Association for Computational Linguistics.
- Divyansh Kaushik, Eduard H. Hovy, and Zachary Chase Lipton. 2020. [Learning the difference that makes a difference with counterfactually-augmented data](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- Divyansh Kaushik, Amrith Setlur, Eduard H. Hovy, and Zachary Chase Lipton. 2021. [Explaining the efficacy of counterfactually augmented data](#). In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. [Large language models are zero-shot reasoners](#).
- Matt J. Kusner, Joshua R. Loftus, Chris Russell, and Ricardo Silva. 2017. [Counterfactual fairness](#). In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4066–4076.
- Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. 2023. [Causal reasoning and large language models: Opening a new frontier for causality](#).
- David A. Lagnado, Tobias Gerstenberg, and Ro'i Zultan. 2013. Causal responsibility and counterfactuals. *Cognitive Science*, 37:1036 – 1073.
- Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. 2017. [Building machines that learn and think like people](#). *Behavioral and Brain Sciences*, 40.
- Andrew Kyle Lampinen. 2023. [Can language models handle recursively nested grammatical structures? A case study on comparing models and humans](#).

- Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. [Solving quantitative reasoning problems with language models](#).
- Belinda Li, Jane Yu, Madian Khabsa, Luke Zettlemoyer, Alon Halevy, and Jacob Andreas. 2022. [Quantifying adaptability in pre-trained language models with 500 tasks](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4696–4715, Seattle, United States. Association for Computational Linguistics.
- Jiaxuan Li, Lang Yu, and Allyson Ettinger. 2023a. [Counterfactual reasoning: Testing language models’ understanding of hypothetical scenarios](#).
- Weixian Waylon Li, Yftah Ziser, Maximin Coavoux, and Shay B. Cohen. 2023b. [BERT is not the count: Learning to match mathematical statements with proofs](#). In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3581–3593, Dubrovnik, Croatia. Association for Computational Linguistics.
- Tal Linzen and Marco Baroni. 2021. Syntactic structure from deep learning. *Annual Review of Linguistics*, 7:195–212.
- Inbal Magar and Roy Schwartz. 2022. [Data contamination: From memorization to exploitation](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 157–165, Dublin, Ireland. Association for Computational Linguistics.
- Kyle Mahowald, Anna A Ivanova, Idan A Blank, Nancy Kanwisher, Joshua B Tenenbaum, and Evelina Fedorenko. 2023. [Dissociating language and thought in large language models: a cognitive perspective](#). *ArXiv preprint*, abs/2301.06627.
- Kamil Malinka, Martin Perešíni, Anton Firc, Ondřej Hujňák, and Filip Januš. 2023. [On the educational impact of ChatGPT: Is artificial intelligence ready to obtain a university degree?](#)
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. [Building a large annotated corpus of English: The Penn Treebank](#). *Computational Linguistics*, 19(2):313–330.
- John McCarthy. 1959. [Programs with common sense](#). In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 75–91.
- Ian R. McKenzie, Alexander Lyzhov, Michael Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, Euan McLean, Aaron Kirtland, Alexis Ross, Alisa Liu, Andrew Gritsevskiy, Daniel Wurgaft, Derik Kauffman, Gabriel Recchia, Jiacheng Liu, Joe Cavanagh, Max Weiss, Sicong Huang, The Floating Droid, Tom Tseng, Tomasz Korbak, Xudong Shen, Yuhui Zhang, Zhengping Zhou, Najoung Kim, Samuel R. Bowman, and Ethan Perez. 2023. [Inverse scaling: When bigger isn’t better](#).
- Antonio Valerio Miceli-Barone, Fazl Barez, Ioannis Konstas, and Shay B. Cohen. 2023. [The larger they are, the harder they fail: Language models do not recognize identifier swaps in Python](#).
- Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. [Cross-task generalization via natural language crowdsourcing instructions](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland. Association for Computational Linguistics.
- Razieh Nabi and Ilya Shpitser. 2018. [Fair inference on outcomes](#). In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1931–1940. AAAI Press.
- Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. [Universal Dependencies v1: A multilingual treebank collection](#). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).
- Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. [Show your work: Scratchpads for intermediate computation with language models](#).
- OpenAI. 2023. [GPT-4 technical report](#).
- Roma Patel and Ellie Pavlick. 2022. [Mapping language models to grounded conceptual spaces](#). In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.
- Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Judea Pearl. 2009. *Causality*, 2nd edition. Cambridge University Press.
- Steven Piantadosi and Felix Hill. 2022. [Meaning without reference in large language models](#). In *NeurIPS 2022 Workshop on Neuro Causal and Symbolic AI (nCSI)*.

- Lianhui Qin, Antoine Bosselut, Ari Holtzman, Chandra Bhagavatula, Elizabeth Clark, and Yejin Choi. 2019. [Counterfactual story reasoning and generation](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5043–5053, Hong Kong, China. Association for Computational Linguistics.
- Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena D. Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. 2020. [Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 794–805, Online. Association for Computational Linguistics.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. [Learning transferable visual models from natural language supervision](#). In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.
- Shauli Ravfogel, Yoav Goldberg, and Tal Linzen. 2019. [Studying the inductive biases of RNNs with synthetic variations of natural languages](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3532–3542, Minneapolis, Minnesota. Association for Computational Linguistics.
- Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 2022. [Impact of pretraining term frequencies on few-shot numerical reasoning](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 840–854, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Yi Ren, Jinzheng He, Xu Tan, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2020. [Popmag: Pop music accompaniment generation](#). In *MM '20: The 28th ACM International Conference on Multimedia, Virtual Event / Seattle, WA, USA, October 12-16, 2020*, pages 1198–1206.
- Laria Reynolds and Kyle McDonell. 2021. [Prompt programming for large language models: dddd the few-shot paradigm](#). In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems, CHI EA '21*, New York, NY, USA. Association for Computing Machinery.
- Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *Proceedings of ICLR*.
- Abulhair Saparov and Tom M. Mitchell. 2022. [Towards general natural language understanding with probabilistic worldbuilding](#). *Transactions of the Association for Computational Linguistics*, 10:325–342.
- Sebastian Schuster and Christopher D. Manning. 2016. [Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks](#). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2371–2378, Portorož, Slovenia. European Language Resources Association (ELRA).
- Pratyusha Sharma, Tamar Rott Shaham, Manel Baradad, Stephanie Fu, Adrian Rodriguez-Munoz, Shivam Duggal, Phillip Isola, and Antonio Torralba. 2024. [A vision check-up for language models](#).
- Roger N Shepard and Jacqueline Metzler. 1971. Mental rotation of three-dimensional objects. *Science*, 171(3972):701–703.
- David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharmas Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. [Mastering chess and Shogi by self-play with a general reinforcement learning algorithm](#). *ArXiv preprint*, abs/1712.01815.
- Mark K Singley and John Robert Anderson. 1989. *The transfer of cognitive skill*. 9. Harvard University Press.
- Dominik Sobania, Martin Briesch, Carol Hanna, and Justyna Petke. 2023. An analysis of the automatic bug fixing performance of ChatGPT. In *Proceedings of the 45th International Conference on Software Engineering*.
- Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabasum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, César Ferri Ramírez,

Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Chris Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodola, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Wang, Gonzalo Jaimovitch-López, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Shevlin, Hinrich Schütze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B. Simon, James Koppel, James Zheng, James Zou, Jan Kocoń, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh D. Dhole, Kevin Gimpel, Kevin Omondi, Kory Mathewson, Kristen Chifullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros Colón, Luke Metz, Lütfi Kerem Şenel, Maarten Bosma, Maarten Sap, Maartje ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramírez Quintana, Marie Tolkienn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L. Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael A. Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Star-

ritt, Michael Strube, Michał Swędrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimeo Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdah Gheini, Mukund Varma T, Nanyun Peng, Nathan A. Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan LeBras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Ruslan Salakhutdinov, Ryan Chi, Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel S. Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima, Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven T. Piantadosi, Stuart M. Shieber, Summer Mishnerghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsu Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Ramasesh, Vinay Uday Prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. 2023. *Beyond the imitation game: Quantifying and extrapolating the capabilities of language models.*

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. *ProofWriter: Generating implications, proofs, and*

- abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.
- Xiaojuan Tang, Zilong Zheng, Jiaqi Li, Fanxu Meng, Song-Chun Zhu, Yitao Liang, and Muhan Zhang. 2023. [Large language models are in-context semantic reasoners rather than symbolic reasoners](#).
- Nenad Tomasev, Ulrich Paquet, Demis Hassabis, and Vladimir Kramnik. 2020. [Assessing game balance with AlphaZero: Exploring alternative rule sets in chess](#). *ArXiv preprint*, abs/2009.04374.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. [LLaMA: Open and efficient foundation language models](#).
- Steven G Vandenberg and Allan R Kuse. 1978. Mental rotations, a group test of three-dimensional spatial visualization. *Perceptual and motor skills*, 47(2):599–604.
- Victor Veitch, Alexander D’Amour, Steve Yadlowsky, and Jacob Eisenstein. 2021. [Counterfactual invariance to spurious correlations in text classification](#). In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 16196–16208.
- Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. [How far can camels go? Exploring the state of instruction tuning on open resources](#).
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. [Chain of thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*.
- Frank F. Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. 2022. [A systematic evaluation of large language models of code](#). In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming, MAPS 2022*, page 1–10, New York, NY, USA. Association for Computing Machinery.
- Xiaoyu Yang, Stephen Obadinma, Huasha Zhao, Qiong Zhang, Stan Matwin, and Xiaodan Zhu. 2020. [SemEval-2020 task 5: Counterfactual recognition](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 322–335, Barcelona (online). International Committee for Computational Linguistics.
- Charles Yu, Ryan Sie, Nicolas Tedeschi, and Leon Bergen. 2020. [Word frequency does not predict grammatical knowledge in language models](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4040–4054, Online. Association for Computational Linguistics.
- Wenhao Yu, Meng Jiang, Peter Clark, and Ashish Sabharwal. 2023. [IfQA: A dataset for open-domain question answering under counterfactual presuppositions](#).
- Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. 2023a. [How language model hallucinations can snowball](#).
- Tianjun Zhang, Yi Zhang, Vibhav Vineet, Neel Joshi, and Xin Wang. 2023b. [Controllable text-to-image generation with GPT-4](#). *ArXiv preprint*, abs/2305.18583.
- Xikun Zhang, Deepak Ramachandran, Ian Tenney, Yanai Elazar, and Dan Roth. 2020. [Do language embeddings capture scales?](#) In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 292–299, Online. Association for Computational Linguistics.

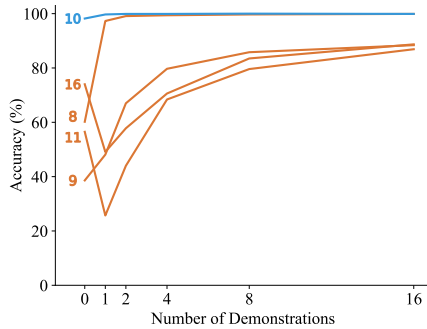


Figure 5: Two-digit addition accuracy when given different numbers of demonstration examples. The default-counterfactual gap reduces, but is not eliminated.

A Full Setups

Unless otherwise specified, we use temperature=0 when sampling from the LMs.

A.1 Arithmetic

We randomly sample 1,000 two-digit addition expressions and evaluate them in bases 8, 9, 10, 11, and 16. Each base is sampled separately—for bases other than base-10, we make sure all expressions evaluate to a different result in that base compared to base-10 so that these expressions discriminate between the bases. To ensure the LMs understand these bases, we design the CCC to ask the model what the number following a given number is. We want the model to know when to carry over and when not to, so we take the 100 smallest numbers in the given basis that ends with the maximum digit in that base, and 100 that end with 0.

A.2 Programming

We use the HumanEval dataset (Chen et al., 2021) which has short Python programs and is commonly used to assess the coding ability of LMs (Bai et al., 2022; Xu et al., 2022; Wang et al., 2023; *i.a.*). It was designed as a code-generation dataset, where a model writes a function from a specification and is evaluated against test cases with input-output pairs. Different from our other tasks, we follow prior work (Touvron et al., 2023; Wang et al., 2023) and (1) use temperature 0.1 when evaluating pass@1 and 0.8 for pass@10, (2) sample 50 responses, and (3) only evaluate without 0-shot CoT. While the original work (Chen et al., 2021) recommended sampling 200 responses, this is very expensive, and we follow Wang et al. (2023) and only sample 50. In Figure 2, we only show the performance on the subset of HumanEval where a 1-based execution of the ground-truth program fails the unit tests. These

are the instances that distinguish between 0- and 1-based indexing. We also report results on the full HumanEval dataset in Table 21.

We also consider another setup—code *execution*, where we give the LM the ground-truth program and ask the LM for the output of the test cases given the input. We remove four programs in HumanEval that are not compatible with this format (ID: 32, 38, 50, and 53), only for this execution task. Because the program would have a different functionality under 1-based indexing, we remove the docstring that is the function description, and also rename the function to the uninformative `function`, to avoid confusing the LM. Some programs also become invalid under 1-based indexing, specifically, those that perform any indexing using \emptyset . We remove all test cases that involve indexing with \emptyset and programs that do not have test cases left after this removal. 150 programs and 969 test cases remain. Some of these test cases may not distinguish between 0- and 1-based indexing. So for our main task (i.e., not CCC), we only consider test cases whose outputs are different under 0- vs. 1-based indexing, and there are 113 of them.

Because we use the same prompt to indicate the counterfactual conditions for both code generation and execution, and because we want to maintain comparability with prior work for the former, we only include CCC in the execution setup. We believe they reflect the LMs’ understanding of 1-based indexing in the generation setup too. We ask the LM for the output of simple tests about 1-based indexing such as `"qrstu"[4]` and `"qrs"[:2]`. They do not require sophisticated reasoning under the counterfactual conditions and yet are sufficient to discriminate between the default and the counterfactual conditions. We append 5 such checks after each of the 150 programs, totaling 750 CCC.

For the execution task, we do not consider PaLM-2, because it only has a maximum of 1,024 output context length and leads to truncated, unparseable results for most test instances, especially under 0-shot CoT.

A.3 Basic Syntactic Reasoning

We follow Ravfogel et al. (2019) and create synthetic variants of English with all six orderings of the subject, verb, and object. Given a dependency tree of a regular English sentence, we alter the order of subject and object nodes with respect to the

corresponding verb. The subtrees rooted at subject or object nodes are moved as a whole, whereas other non-core dependent nodes (e.g., prepositional phrases) are kept in the original positions. We use 100 sentences from English Penn Treebank (Marcus et al., 1993), and convert the original phrase-structure trees into Universal Dependencies (Nivre et al., 2016) using the Stanford converter (Schuster and Manning, 2016).

Our task is to identify the main verb and the main subject of a sentence. We only choose sentences where the main subject contains a single word. Ravfogel et al. (2019)’s data generation procedure sometimes results in sentences in the SVO order to be unnatural English sentences. To eliminate this complexity, we retain only sentences whose SVO variant according to Ravfogel et al. (2019)’s data generation procedure is identical to the original English sentence.

We designed the CCC to assess how well LMs understand the instruction that explains the difference of word orders in the counterfactual settings. We synthetically generate 100 simple three-word sentences (e.g., “anna saw john”) in five counterfactual word orders (e.g., “anna john saw” in SOV), and ask LMs to reconstruct the original English sentences in SVO order. Conceptually, this is equivalent to asking the model to identify the subject, verb, and object in the perturbed order, but using a format that is more familiar to the LM.

To generate the simple sentences for the CCC, we designed a simple context-free grammar where the subject and the object are sampled from the vocabulary of person names, and the verb is sampled from the set {saw, loves, calls, knows, sees}. A key feature of the sentences generated from this approach is their retained plausibility when the subject and object are interchanged. This means that given a counterfactual sentence (e.g., “anna john saw”), there are two natural English sentences as candidates for reconstruction (i.e., “anna saw john” and “john saw anna”). Due to this inherent ambiguity, LMs cannot default to the heuristic of treating the synthetic sentence as bag-of-words and then reconstructing the most natural ordering of those words in real English. The random baseline chooses a random noun as the main subject and a random verb as the main verb.

A note on CCC results. The results for this task are shown in Table 22. Generally, the models pass our crafted CCC challenge with decent ac-

curacy, but we observed that, in a few cases, the LMs are confused by the reconstruction ambiguity explained above. GPT-3.5 and Claude fail in the OVS settings where they often directly copy the original sentence—e.g., instead of reconstructing “anna saw john” to “john saw anna”, they simply copy the original sentence “anna saw john” as the output. Similarly, PaLM-2 often incorrectly reverses the subject and object in the SOV and VSO settings—e.g., instead of reconstructing “calls tom lucas” to “tom calls lucas”, it outputs “lucas calls tom”.

A.4 Natural Language Reasoning with First-Order Logic

We use the FOLIO dataset (Han et al., 2022) that contains premises most of which are consistent with common sense and are hence amenable to our counterfactual study. We use the full dataset, combining the training and development sets for a total of 1,204 instances, for the logistic regression analysis in §5.1. But for our counterfactual study, automatically altering the premises to violate common sense is not trivial, so one author manually rewrote the premises of a subset of 81 instances to be counterfactual, and another author verified the rewrite. Considering the analysis in §5.1, we chose this subset by including every instance with premises all of which GPT-4 believes to be true and whose conclusion whose GPT-4-believed truth value matches the entailment label.

We explicitly instruct the model to use no common sense or world knowledge (§C), thereby requiring symbolic reasoning. For the CCC, we ask the model if the unaltered or the altered premise is true, when both are presented as options, and expect the latter.

While the FOLIO dataset has a public release, the authors have made subsequent updates which, at the time of this paper, have not been made public. We hence do not release the LM interaction data for this task, and use a fictional example in Table 5.

A.5 Spatial Reasoning

We ask the LM for the coordinates of objects in a room. We randomly sample 100 rooms, each with 3 different objects placed in 3 different cardinal directions specified using unit vectors (out of north $(0, -1)$, south $(0, 1)$, east $(1, 0)$, and west $(-1, 0)$ as the default conditions). Though using a downward-facing y -axis as the default condition may be counter-intuitive, it is natural when draw-

ing top-to-bottom and is the convention in most image processing libraries such as OpenCV (Python), Pillow (Python), and Processing (Java, JavaScript, Python), as well as graphic design applications such as Adobe Illustrator. We believe this system is the most often encountered during LM pretraining. However, other libraries with an upward-facing y -axis also exist, such as matplotlib (Python), ggplot (R), and D3 (JavaScript).

For the counterfactual setting, we alter the direction-unit vector mapping, and ask for the object coordinates in the new system. We consider two direction-swapped worlds (north-south and east-west), three rotated worlds (by 90° , 180° , and 270°), and a randomly permuted world. We evaluate the relative positions of objects and report the instance-level accuracy that requires all 3 objects in a room to be located correctly as the main metric. The random accuracy is around 16.7%.⁶ We also report the object-level accuracy in Table 24. As the CCC, we make sure that the LM understands the permuted world by asking it to also specify the coordinates of the unit vectors representing the 4 cardinal directions in the output.

A.6 Drawing

We choose 100 objects from five Emoji⁷ categories: activity, travel & places, animals & nature, food & drink, and objects. Since LMs cannot generate images at the pixel level, we use code as an intermediate abstraction for sketch generation. We do our best to select objects that are easy to draw using code, verified by multiple authors. We consider the Processing language for our experiment which supports a variety of shapes and colors and is widely used in visualization and which Sharma et al. (2024) found the LMs to be more adept in. Our initial experiments found this language to achieve the best drawing performance compared to other graphics and image processing frameworks, including TikZ, SVG, and matplotlib.

For the counterfactual settings, we ask the LMs to draw the same object, but vertically flipped (i.e., upside-down), or rotated by 90° or 180° . We also ask the LMs to avoid using any transformation functions such as `rotate` and `scale` to avoid shortcuts. Before our quantitative evaluation, we flip/rotate back the generated drawing.

⁶When not considering cases where objects are placed in the same line, there are 24 permutations for placing 3 objects in 4 different directions, of which 4 can be considered correct.

⁷<https://getemoji.com>

We use human evaluation by asking human annotators to determine whether the drawing matches the object (detailed in §E). We instruct the annotators to consider orientation as part of correctness and for objects that have a canonical orientation, they must be drawn in that orientation. We average the results over 4 annotators. We also show a breakdown of accuracy depending on whether an object has a canonical orientation or not, as judged by the annotators, in Table 26. In addition, we consider multi-class classification accuracy using CLIP (Radford et al., 2021) as an automatic metric, where we ask CLIP to classify the drawing into our 100 categories in a 0-shot fashion. We include the CLIP multi-class classification accuracy in Table 25. We note that the accuracy of the CLIP model for our setup is not guaranteed: first, our generated sketches may be distributionally different from the predominantly photorealistic images in CLIP’s training data; also, CLIP might be insensitive to the object’s orientation, but that distinguishes between our default and counterfactual settings. Therefore, to verify the reliability of this automatic evaluation, we randomly sample 10 objects for each model and for each default/counterfactual setting, and perform human evaluation on the 240 generated images. We find that CLIP’s judgment aligns with human annotators’ 84% of the time, suggesting the reliability of this evaluation.

For this task, we do not consider PaLM-2 due to its limited context length. Our preliminary experiments also found PaLM-2 to struggle in generating parseable Processing code, even in the default setting.

We construct the CCC baseline by requiring the LMs to additionally draw a line at the top of the figure and flip/rotate it as well. A successful flipping/rotation of the line, as judged by the annotators and verified in the generated code if necessary, demonstrates an understanding of the counterfactual world.

A.7 Music

A.7.1 Playing Chords on Instruments

We measure LMs’ abilities to give correct fret placements for ukulele and guitar chords in an existing database.^{8,9} We include the following kinds of

⁸<https://github.com/tombatossals/chords-db>

⁹We heuristically filter out incorrect datapoints by filtering out chords that either have the wrong number of notes or lack the root note.

chords from the database: sus2 (suspended second chord), sus4 (suspended fourth chord), min triad (minor triad), maj triad (major triad), dim7 (diminished seventh chord), aug7 (augmented seventh chord), maj7 (major seventh chord), min7 (minor seventh chord), dom7 (dominant seventh chord), 5 (fifth interval), and 6 (sixth chord).

In the counterfactual setting, we instruct LMs to provide fret placements for a “special” ukulele or guitar where one of the strings is altered. We experiment with perturbations of different sizes: For guitar, we experiment with one-string changes by one note ($EADGBE \rightarrow EBDGBE$; $EADGBE \rightarrow FADGBE$), one-string changes by two notes ($\rightarrow ECDGBE$), and two string changes ($\rightarrow ECFGBE$). We also experiment with a one-string change that corresponds to a common alternate tuning of a guitar called drop-D tuning ($\rightarrow DADGBE$). For ukulele, we experiment with one-string changes by one note ($GCEA \rightarrow FCEA$; $\rightarrow ACEA$), one-string change by two notes ($\rightarrow BCEA$), and two-string changes by two notes ($\rightarrow BEEA$). The generated fret placements for a chord are considered *correct* if all and only the notes in the corresponding chord (e.g., *C, E, G* for a C major triad) are produced, irrespective of order.

As the CCC, we assess LMs’ understanding of the given instrument’s strings by asking them to identify what notes a given sequence of frets corresponds to; for the CCC, the sequences are either all fret 0, all fret 1, or all fret 2. We compute CCC accuracy at the fret level (as opposed to the sequence level).

A.7.2 Retrieving Notes of Famous Melodies

For 8 famous melodies, we prompt LMs to retrieve the n -th note in the melody, where n is between 1 and 7 (inclusive). In the counterfactual setting, we prompt the LM to do the same but in a different key. The list of melodies and keys we experiment with is below.

We use *C Major* as the key for songs as the default condition given its popularity for famous melodies like children’s songs. We use other keys as the counterfactual keys.¹⁰

¹⁰We note that some songs may have multiple canonical keys (e.g., “Twinkle Twinkle Little Star” is also frequently performed in keys like G major or D Major.) In some initial exploration, we validated that C Major was at least one of the canonical keys for the melodies chosen, both by verifying that popular sheet music for these songs was written in C Major, and by asking GPT-3.5 to generate the melodies in an unspecified key and verifying that the generated key was C

As the CCC, we assess LMs’ understanding of the given keys by asking them to retrieve the n -th note of the scale of the given key.

Melodies: Twinkle Twinkle Little Star, Mary Had a Little Lamb, Happy Birthday to You, Somewhere Over the Rainbow, Row Row Row Your Boat, Old Macdonald Had a Farm, Itsy Bitsy Spider, London Bridge is Falling Down.

Counterfactual Keys: B# major, C# major, Db major, D major, D# major, Eb major, Fb major, E major, E# major, F major, F# major, Gb major, G major, G# major, Ab major, A major, A# major, Bb major, Cb major, B major.

A.8 Chess

We evaluate an LM’s ability to understand chess rules by checking if it can determine whether a 4-move opening follows the rules of chess or not. In the counterfactual setting, we swap the position of bishops and knights on the board and evaluate the same task. For each setting, we randomly sample 400 unique chess openings via a procedural generation algorithm: 200 are legal for the default setting but not for the counterfactual setting, and vice versa for the other 200, ensuring a more balanced and fair classification problem. We represent the moves as the LM input using the PGN format, the standard for chess moves description.

For the CCC, we ask an LM for the starting positions of the four knights and four bishops on the board to make sure it understands the new initial board. For both the default and counterfactual settings, we ask for the positions of white knights, white bishops, black knights, and black bishops, totaling 8 pieces, and evaluate using accuracy. Since concluding the effectiveness of our counterfactual prompt using merely 8 CCC may not be statistically significant, we sample 15 LM responses using temperature=0.1 for asking about each piece.

A.9 SET Game

We synthetically generate SET boards, consisting of 12 cards, each with exactly one 3-card SET that satisfies the game rules in §3. We represent each card with a string representation, e.g., (3|open|red|diamond). In preliminary experiments, we tried to ask the LMs to find the SET directly, but found that they cannot perform this task well (see Figure 4c, “Number of Cards to Major.

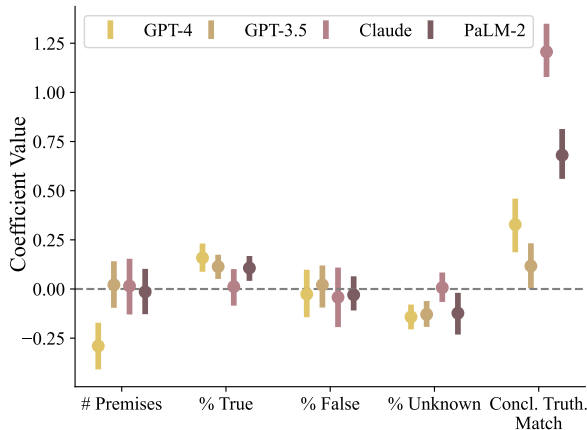


Figure 6: Logistic regression coefficients of features that predict whether an LM correctly predicts the label of an instance. “Concl. Truth. Match” is a binary feature that is 1 iff the instance label matches the (LM-believed) truthfulness of the conclusion. The 95% confidence intervals are also shown. LMs tend to predict more correctly when there are more true premises, when the instance label matches the conclusion truthfulness, but less correctly with more false and unknown premises.

Find”= 3). Therefore, in our main evaluation, we expose 2 cards in the SET and ask the LM to identify the missing one that completes the SET.

In the counterfactual setting, we invert the rule for the *number* attribute to require that two cards in the SET should have the same number but the other card should be different. For the CCC, we ask the model to verify the validity of a given SET instead of finding it. In each CCC instance, we either give a valid SET from the board, or 3 randomly sampled cards that do not constitute a valid SET. We ask the model to classify whether the given combination is valid or invalid. We note that our counterfactual perturbation ensures that the each SET cannot be simultaneously valid in the default setting and the counterfactual setting, and hence this CCC is discriminative between the two settings.

B Additional Analysis

In this section, we provided additional analysis to §5.

B.1 LMs’ Logical Ability Correlates With the Proximity between Default and Counterfactual Conditions

In §5.2, we mentioned that the commonsense-abiding tendency of most statements in the FOLIO dataset (Han et al., 2022) enables us to quantify how the a LM performs when the premises and the conclusion increasingly deviate from the real

world (as believed by the LM), without artificial counterfactual perturbations.

Concretely, for each test instance in FOLIO, we ask the LMs whether the premises and conclusion are true, false, or uncertain. We train a logistic regression model to predict LM correctness on each test instance, using as features the total number of premises in an input, the proportion of the premises that are true/false/uncertain, as encoded by the LM, as well as whether the LM-predicted truthfulness of the conclusion matches the label of the instance.

Figure 6 shows the learned coefficients of these features, as well as their 95% confidence interval bootstrapping with 1,000 iterations (Efron and Tibshirani, 1993). Ideally, a robust model should predict solely based on symbolic deduction and extralinguistic truthfulness information should not affect its accuracy. In other words, these features should all have coefficients 0 and have no predictive power with respect to the model’s correctness. However, all LMs predict more correctly with more realistic (true) premises, and when the conclusion’s LM-predicted truthfulness matches the label (indicating a tendency to predict the label solely based on the conclusion, ignore premises). On the other hand, they tend to perform worse when there are more false or uncertain premises. Most of these trends are statistically significant. This means that the reasoning ability of LMs is affected by the distance between the (LM-believed) real world and the world state under which the LMs are expected to reason.

B.2 Qualitative Analysis of Drawing Results

We conduct a qualitative error analysis on the drawing task and show some examples in Figure 7. We first note that GPT-4 successfully passes the CCC for these cases (see §3; but not displayed here), indicating that it understands the flip/rotation instructions. However, the objects in the counterfactual worlds are often not flipped or rotated. Even when they are transformed appropriately, the resulting drawing is often simplified or of worse quality (e.g., Unicorn, Cake). We also observed much more syntactically invalid programs in the counterfactual cases for GPT-3.5.¹¹ These results indicate that even when a model can perform a task in the counterfactual setup, its capabilities are reduced.

¹¹On average, the number of parseable programs generated by GPT-3.5 drops from 99% in the default condition to 62%, 71%, and 75% for the vertically flipped, 90° rotated, and 180° rotated settings, respectively.

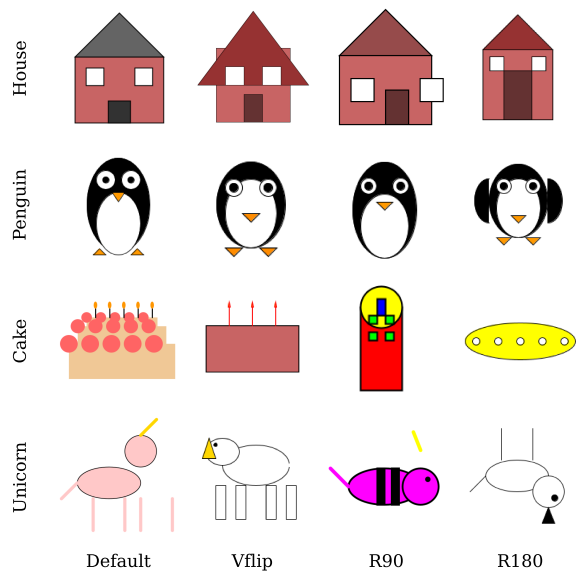


Figure 7: Visualizations of objects drawn by GPT-4 under the default (upright) and counterfactual conditions: vertical flip (Vflip, i.e. upside-down), rotates 90 degrees (R90), and 180 degrees (R180). In all cases, the CCC (not shown) passes. We show the original output, without flipping/rotating back as in our quantitative evaluation (§A.6). For the counterfactual settings, GPT-4 either does not transform the objects as instructed (e.g., house and penguin) or struggles to draw meaningful objects (e.g., cake and unicorn).

C Prompts

We provide the exact prompts that we used to query the LMs in Tables 1 to 17. For clarity, we give a concrete prompt that embeds a test instance, rather than a template. We explain minor design decisions in the respective captions. We do not use the system message field for any model.

D Raw Results

We show the numeric results in Tables 18 to 34.

E Human Annotation Details

Our human annotators are all current students from American, Asian, and European institutions, to whom we explained the purpose of the annotation. We show the exact instruction of our drawing evaluation in Figure 8.

Mode	Prompt
Test	You are a mathematician. Assuming that all numbers are in base-11 where the digits are "0123456789A", what is $59+37$? {Let's think step by step, and } end the response with the result in " <code>\boxed{result}</code> ".
CCC	You are a mathematician. Assuming that all numbers are in base-11 where the digits are "0123456789A", what is the next number after 11A? Do this by counting the few preceding numbers and completing the sequence. End the response with the result.
Few-Shot CoT	<p>You are a mathematician. Assuming that all numbers are in base-11 where the digits are "0123456789A", what is $25+68$? Let's think step by step, and end the response with the result in "<code>\boxed{result}</code>". We add the ones digits first. In base-11, $5+8=12$. So the ones digit of the final sum is 2. We need to carry over the 1 to the tens place. Then we add the tens digits. In base-11, $2+6=8$. Since we carried over the 1, $8+1=9$. So the tens digit of the final sum is 9. Putting the digits of the final sum together, we get <code>\boxed{92}</code>. ...[optionally more demonstrations in the same format]...</p> <p>You are a mathematician. Assuming that all numbers are in base-11 where the digits are "0123456789A", what is $59+37$? Let's think step by step, and end the response with the result in "<code>\boxed{result}</code>".</p>

Table 1: Prompts for the arithmetic task. **{Let's think step by step, and }** is added only if 0-shot CoT is used (and the following e is capitalized without 0-shot CoT). We use the `\boxed{result}` syntax to wrap results because we found in preliminary experiments that the models tend to use this format even without this specification. The Few-Shot CoT prompt is used for the analysis in §5.4.

Mode	Prompt
Default	<p>You are an expert programmer. What does the following code snippet in Python 3.7 print?</p> <pre> ```python def function(lst): return sum([lst[i] for i in range(1, len(lst), 2) if lst[i] % 2 == 0]) print([function([4, 88])]) print([function([4, 5, 6, 7, 2, 122])]) print([function([4, 0, 6, 7])]) print([function([4, 4, 6, 8])]) print([list(range(3))]) print([[4, 5, 6].pop(2)]) print(["qrs"[:2]]) print(["qrstu"[4]]) print([list(enumerate("qrstuv"))]) ``` </pre> <p>{Let's think step by step. Write out intermediate results and reasoning processes as needed. }End the response by saying "The final output is:" and a unified summary ```python``` code block with <i>*ALL*</i> the output, in which each line represents the output of each print statement.</p>
CF	<p>You are an expert programmer who can readily adapt to new programming languages. There is a new programming language, ThonPy, which is identical to Python 3.7 except all variables of the `list`, `tuple`, and `str` types use 1-based indexing, like in the MATLAB and R languages, where sequence indices start from 1. That is, index `n` represents the `n`-th element in a sequence, NOT the `n+1`-th as in 0-based indexing. This change only affects when the index is non-negative. When the index is negative, the behavior is the same as Python 3.7. This also affects methods of these classes such as `index` and `pop`. The built-in functions `enumerate` and `range` also use 1-based indexing: by default, the index of `enumerate` starts from 1, and so does the lower bound of `range` when not supplied (the higher bound is unchanged).</p> <p>For example,</p> <pre> ```thonpy assert (7, 8, 9)[1] == 7 assert ["abc", "def", "ghi"][3] == "ghi" assert "abcde"[4] == "d" assert "abc"[:2] == "a" assert [7, 8, 9][1:] == [7, 8, 9][1:5] == [7, 8, 9][1::1] == [7, 8, 9][:4] == [9, 8, 7][::-1] == [9, 8, 7, 6][3::-1] == [7, 8, 9] assert list(enumerate([7, 8, 9])) == [(1, 7), (2, 8), (3, 9)] assert list(range(2)) == [1] assert list(range(2, 4)) == [2, 3] assert {0: 7, 1: 8, 2: 9}[1] == 8 assert [7, 8, 9].index(8) == 2 ``` </pre> <p>What does the following code snippet in ThonPy print?</p> <pre> ```thonpy def function(lst): return sum([lst[i] for i in range(1, len(lst), 2) if lst[i] % 2 == 0]) print([function([4, 88])]) print([function([4, 5, 6, 7, 2, 122])]) print([function([4, 0, 6, 7])]) print([function([4, 4, 6, 8])]) print([list(range(3))]) print([[4, 5, 6].pop(2)]) print(["qrs"[:2]]) print(["qrstu"[4]]) print([list(enumerate("qrstuv"))]) ``` </pre> <p>{Let's think step by step. Write out intermediate results and reasoning processes as needed. }End the response by saying "The final output is:" and a unified summary ```thonpy``` code block with <i>*ALL*</i> the output, in which each line represents the output of each print statement.</p>

Table 2: Prompts for the program execution task. **{Let's think step by step. Write out intermediate results and reasoning processes as needed. }** is added only if 0-shot CoT is used. All the print statements wrap the expression in a singleton list for the ease of parsing, so that (a) each output always takes a single line even with line breaks in the middle, and (b) we can distinguish between a string representation of e.g. an integer and the integer type.

Mode	Prompt
	<p>You are an expert programmer. Complete the following function in Python 3.7. Please only output the code for the completed function.</p>
Default	<pre>def add(lst): """Given a non-empty list of integers lst. add the even elements that are at odd indices.. Examples: add([4, 2, 6, 7]) ==> 2 """</pre>
CF	<p>You are an expert programmer who can readily adapt to new programming languages. There is a new programming language, ThonPy, which is identical to Python 3.7 except all variables of the `list`, `tuple`, and `str` types use 1-based indexing, like in the MATLAB and R languages, where sequence indices start from 1. That is, index `n` represents the `n`-th element in a sequence, NOT the `n+1`-th as in 0-based indexing. This change only affects when the index is non-negative. When the index is negative, the behavior is the same as Python 3.7. This also affects methods of these classes such as `index` and `pop`. The built-in functions `enumerate` and `range` also use 1-based indexing: by default, the index of `enumerate` starts from 1, and so does the lower bound of `range` when not supplied (the higher bound is unchanged).</p> <p>For example,</p> <pre>```thonpy assert (7, 8, 9)[1] == 7 assert ["abc", "def", "ghi"][3] == "ghi" assert "abcde"[4] == "d" assert "abc"[:2] == "a" assert [7, 8, 9][1:] == [7, 8, 9][1:5] == [7, 8, 9][1::1] == [7, 8, 9][:4] == [9, 8, 7][::-1] == [9, 8, 7, 6][3::-1] == [7, 8, 9] assert list(enumerate([7, 8, 9])) == [(1, 7), (2, 8), (3, 9)] assert list(range(2)) == [1] assert list(range(2, 4)) == [2, 3] assert {0: 7, 1: 8, 2: 9}[1] == 8 assert [7, 8, 9].index(8) == 2 ```</pre> <p>Complete the following function in ThonPy. Please only output the code for the completed function.</p> <pre>def add(lst): """Given a non-empty list of integers lst. add the even elements that are at odd indices.. Examples: add([4, 2, 6, 7]) ==> 2 """</pre>

Table 3: Prompts for the program generation task.

Mode	Prompt
Default	<p>You are an expert in linguistics. Your task is to identify the main verb and the main subject of a sentence in English. Show the main verb (a single word) and its subject (also a single word) after the prefix 'Main verb and subject: '.</p> <p>Sentence: japan had just opened its doors to the world after about 250 years of isolation .</p> <p>{Let's think step by step. }</p>
CF	<p>You are an expert in linguistics. Imagine a language that is the same as English with the only exception being that it uses the verb-object-subject order instead of the subject-verb-object order. Your task is to identify the main verb and the main subject in a sentence in this imaginary language. Show the main verb (a single word) and its subject (also a single word) after the prefix 'Main verb and subject: '.</p> <p>Sentence: had just opened its doors japan to the world after about 250 years of isolation .</p> <p>{Let's think step by step. }</p>
CCC	<p>You are an expert in linguistics. Imagine a language that is the same as English with the only exception being that it uses the verb-subject-object order instead of the subject-verb-object order. Your task is to reconstruct the original sentence in English. You should only use the words in the same form as they appear in the given sentence.</p> <p>Sentence: saw anna john</p> <p>Show the original sentence at the end after the prefix 'Original sentence: '.</p> <p>{Let's think step by step. }</p>

Table 4: Prompts for the basic syntactic reasoning task. **{Let's think step by step. }** is added only if 0-shot CoT is used.

Mode	Prompt
Test	<p>Consider the following premises: "All corgis are reptiles. All reptiles are plants." Assuming no other commonsense or world knowledge, is the sentence "All corgis are plants." necessarily true, necessarily false, or neither? {Let's think step by step, and }end the response with either "necessarily true", "necessarily false", or "neither".</p>
CCC	<p>Consider the following premises: "All corgis are reptiles. All reptiles are plants." Assuming no other commonsense or world knowledge, which sentence between (a) "All corgis are reptiles." and (b) "All corgis are mammals." is definitely true? Answer just "(a)" or "(b)" and nothing else. You MUST choose one and only one, so DO NOT say neither or both.</p>

Table 5: Prompts for the natural language reasoning task. **{Let's think step by step, and }** is added only if 0-shot CoT is used (and the following e is capitalized without 0-shot CoT). We only use a made-up example here rather than one in the dataset due to the non-publicness of the dataset (§A.4). Default and counterfactual tasks share the same test template, but the instances themselves are changed to be counterfactual. For the CCC, we separate each changed premise in an instance into a separate prompt. The default statement and the counterfactual statement are matched to (a) and (b) randomly. We do not distinguish between CCC with or without 0-shot CoT.

Mode	Prompt
Default	<p>You are in the middle of a kitchen. There is a microwave on the south side. There is a fridge on the west side. There is a coffee machine on the north side. We define the following directions. The north direction is $(0, -1)$. The south direction is $(0, 1)$. The east direction is $(1, 0)$. The west direction is $(-1, 0)$. What's the layout of the room in the following format? You can estimate the size of the objects.</p> <pre> ```json {"name": "??", "width": 500, "height": 500, "directions": {"north": "??", "south": "??", "east": "??", "west": "??"}, "objects": [{"name": "??", "x": "??", "y": "??"}]} ``` </pre> <p>{Let's think step by step.}</p>
CF	<p>You are in the middle of a kitchen. There is a microwave on the south side. There is a fridge on the west side. There is a coffee machine on the north side. We define the following directions. The north direction is $(-1, 0)$. The south direction is $(1, 0)$. The east direction is $(0, -1)$. The west direction is $(0, 1)$. What's the layout of the room in the following format? You can estimate the size of the objects.</p> <pre> ```json {"name": "??", "width": 500, "height": 500, "directions": {"north": "??", "south": "??", "east": "??", "west": "??"}, "objects": [{"name": "??", "x": "??", "y": "??"}]} ``` </pre> <p>{Let's think step by step.}</p>

Table 6: Prompts for the spatial reasoning task. **{Let's think step by step.}** is added only if 0-shot CoT is used.

Mode	Prompt
Default 1	<p>You are an expert programmer and drawer. Imagine an image: there is a line at the top and a house in the middle. Can you try your best to draw this image using the `processing` language? Please do not use any built-in transformation functions, such as `rotate` and `scale`. Also, avoid defining any custom transformation functions yourself. Do not load any existing images. Please include as many details of the house as possible and put everything together in the end.</p> <p>{Let's think step by step.}</p>
Default 2	<p>You are an expert programmer and drawer. Imagine an image: there is a line at the top and a house in the middle. Can you try your best to draw the line and the house using the `processing` language? Please do not use any built-in transformation functions, such as `rotate`, `scale`, and `translate`. Also, avoid defining any custom transformation functions yourself. Do not load any existing images. Please include as many details of the house as possible and put everything together in the end.</p> <p>{Let's think step by step.}</p>
CF 1	<p>You are an expert programmer and drawer. Imagine an image: there is a line at the top and a house in the middle. Can you rotate this image 180 degrees and try your best to draw it using the `processing` language? Please do not use any built-in transformation functions, such as `rotate` and `scale`. Also, avoid defining any custom transformation functions yourself. Do not load any existing images. Do not draw the original objects. Please include as many details of the house as possible and put everything together in the end.</p> <p>{Let's think step by step.}</p>
CF 2	<p>You are an expert programmer and drawer. Imagine an image: there is a line at the top and a house in the middle. Can you rotate this image 180 degrees and try your best to draw the 180-degree rotated line and the 180-degree rotated house using the `processing` language? Please do not use any built-in transformation functions, such as `rotate`, `scale`, and `translate`. Also, avoid defining any custom transformation functions yourself. Do not load any existing images. Do not draw the original objects. Please include as many details of the house as possible and put everything together in the end.</p> <p>{Let's think step by step.}</p>

Table 7: Prompts for the drawing task. **{Let's think step by step.}** is added only if 0-shot CoT is used. We use prompt 1 for GPT-4 and prompt 2 for GPT-3.5 and Claude. We chose the prompt based on the best CCC accuracy for each respective model. In our preliminary experiments, we found that switching the prompt hurts CCC accuracy by more than 20% for both GPT-4 and GPT-3.5. Claude does not follow our instructions when using prompt 1, leading to almost 0% CCC's accuracy.

Mode	Prompt
Default	<p>You are an expert guitar player. I have a guitar with standard strings E-A-D-G-B-E.</p> <p>I want you to tell me how I could play the E minor triad on this guitar.</p> <p>Make sure to choose one final answer, which you should start with 'ANSWER:' and specify in the following format:</p> <p>ANSWER: E string: fret FRET A string: fret FRET D string: fret FRET G string: fret FRET B string: fret FRET E string: fret FRET</p> <p>Use fret 0 to indicate an open string and fret X to indicate not playing a string. Each increase in fret corresponds to an increase in half a note.</p> <p>{Let's think step by step.}</p>
CF	<p>You are an expert guitar player. I have a special guitar with strings tuned to E-C-F-G-B-E instead of the standard E-A-D-G-B-E. Note that what is the standard A string is instead tuned to C, and the standard D string is instead tuned to F. All other strings are the same.</p> <p>I want you to tell me how I could play the E minor triad on this guitar.</p> <p>Make sure to choose one final answer, which you should start with 'ANSWER:' and specify in the following format:</p> <p>ANSWER: E string: fret FRET C string: fret FRET F string: fret FRET G string: fret FRET B string: fret FRET E string: fret FRET</p> <p>Use fret 0 to indicate an open string and fret X to indicate not playing a string. Each increase in fret corresponds to an increase in half a note.</p> <p>{Let's think step by step.}</p>

Table 8: Prompts for chord fingering: guitar. **{Let's think step by step.}** is added only if 0-shot CoT is used.

Mode	Prompt
Default	<p>You are an expert guitar player. I have a guitar with standard strings E-A-D-G-B-E.</p> <p>I want you to tell me what notes the following sequences of finger positions corresponds to:</p> <p>E string: fret 0 A string: fret 0 D string: fret 0 G string: fret 0 B string: fret 0 E string: fret 0</p> <p>Note that fret 0 indicates an open string, and each increase in fret corresponds to an increase in half a note.</p> <p>Make sure to choose one final answer, which you should start with 'ANSWER:' and format with dash-separated notes in the order of strings E-A-D-G-B-E.</p> <p>{Let's think step by step.}</p>
CF	<p>You are an expert guitar player. I have a special guitar with strings tuned to E-C-F-G-B-E instead of the standard E-A-D-G-B-E. Note that what is the standard A string is instead tuned to C, and the standard D string is instead tuned to F. All other strings are the same.</p> <p>I want you to tell me what notes the following sequences of finger positions corresponds to:</p> <p>E string: fret 0 C string: fret 0 F string: fret 0 G string: fret 0 B string: fret 0 E string: fret 0</p> <p>Note that fret 0 indicates an open string, and each increase in fret corresponds to an increase in half a note.</p> <p>Make sure to choose one final answer, which you should start with 'ANSWER:' and format with dash-separated notes in the order of strings E-C-F-G-B-E.</p> <p>{Let's think step by step.}</p>

Table 9: CCC prompts for chord fingering: guitar. **{Let's think step by step.}** is added only if 0-shot CoT is used.

Mode	Prompt
Default	<p>You are an expert ukulele player. I have a ukulele with standard strings G-C-E-A.</p> <p>I want you to tell me how I could play the E minor triad on this ukulele.</p> <p>Make sure to choose one final answer, which you should start with 'ANSWER:' and specify in the following format:</p> <p>ANSWER: G string: fret FRET C string: fret FRET E string: fret FRET A string: fret FRET</p> <p>Use fret 0 to indicate an open string and fret X to indicate not playing a string. Each increase in fret corresponds to an increase in half a note.</p> <p>{Let's think step by step.}</p>
CF	<p>You are an expert ukulele player. I have a special ukulele with strings tuned to F-C-E-A instead of the standard G-C-E-A. Note that what is the standard G string is instead tuned to F. All other strings are the same.</p> <p>I want you to tell me how I could play the E minor triad on this ukulele.</p> <p>Make sure to choose one final answer, which you should start with 'ANSWER:' and specify in the following format:</p> <p>ANSWER: F string: fret FRET C string: fret FRET E string: fret FRET A string: fret FRET</p> <p>Use fret 0 to indicate an open string and fret X to indicate not playing a string. Each increase in fret corresponds to an increase in half a note.</p> <p>{Let's think step by step.}</p>

Table 10: Prompts for chord fingering: ukulele. **{Let's think step by step.}** is added only if 0-shot CoT is used.

Mode	Prompt
Default	<p>You are an expert ukulele player. I have a ukulele with standard strings G-C-E-A.</p> <p>I want you to tell me what notes the following sequences of finger positions corresponds to:</p> <p>G string: fret 0 C string: fret 0 E string: fret 0 A string: fret 0</p> <p>Note that fret 0 indicates an open string, and each increase in fret corresponds to an increase in half a note.</p> <p>Make sure to choose one final answer, which you should start with 'ANSWER:' and format with dash-separated notes in the order of strings G-C-E-A.</p> <p>{Let's think step by step.}</p>
CF	<p>You are an expert ukulele player. I have a special ukulele with strings tuned to F-C-E-A instead of the standard G-C-E-A. Note that what is the standard G string is instead tuned to F. All other strings are the same.</p> <p>I want you to tell me what notes the following sequences of finger positions corresponds to:</p> <p>F string: fret 0 C string: fret 0 E string: fret 0 A string: fret 0</p> <p>Note that fret 0 indicates an open string, and each increase in fret corresponds to an increase in half a note.</p> <p>Make sure to choose one final answer, which you should start with 'ANSWER:' and format with dash-separated notes in the order of strings F-C-E-A.</p> <p>{Let's think step by step.}</p>

Table 11: CCC prompts for chord fingering: ukulele. **{Let's think step by step.}** is added only if 0-shot CoT is used.

Mode	Prompt
Default	<p>You are an expert musician. What is the second note of the melody of the song 'Twinkle Twinkle Little Star' in C major? Make sure to choose one final answer, which you should start with 'ANSWER:' and specify in the following format: NOTE={note}.</p> <p>{Let's think step by step.}</p>
CF	<p>You are an expert musician. What is the second note of the melody of the song 'Twinkle Twinkle Little Star' in Db major? Make sure to choose one final answer, which you should start with 'ANSWER:' and specify in the following format: NOTE={note}.</p> <p>{Let's think step by step.}</p>

Table 12: Prompts for melody retrieval. **{Let's think step by step.}** is added only if 0-shot CoT is used.

Mode	Prompt
Default	<p>You are an expert musician. What is the second note of the C major scale? Make sure to choose one final answer, which you should start with 'ANSWER:' and specify in the following format: NOTE={note}.</p> <p>{Let's think step by step.}</p>
CF	<p>You are an expert musician. What is the second note of the Db major scale? Make sure to choose one final answer, which you should start with 'ANSWER:' and specify in the following format: NOTE={note}.</p> <p>{Let's think step by step.}</p>

Table 13: CCC prompts for melody retrieval. **{Let's think step by step.}** is added only if 0-shot CoT is used.

Mode	Prompt
Default	<p>You are a chess player. Given an opening, determine whether the opening is legal. The opening doesn't need to be a good opening. Answer "yes" if all moves are legal. Answer "no" if the opening violates any rules of chess. Is the new opening "1. e4 e6 2. Be2 Bc5" legal?</p> <p>{Let's think step by step}</p>
CF	<p>You are a chess player. You are playing a chess variant where the starting positions for knights and bishops are swapped. For each color, the knights are at placed that where bishops used to be and the bishops are now placed at where knights used to be. Given an opening, determine whether the opening is legal. The opening doesn't need to be a good opening. Answer "yes" if all moves are legal. Answer "no" if the opening violates any rules of chess. Under the custom variant, is the new opening "1. e4 e6 2. Nfe2 Nc5" legal?</p> <p>{Let's think step by step}</p>

Table 14: Prompts for the chess task. **{Let's think step by step}** is added only if 0-shot CoT is used.

Mode	Prompt
Default	<p>You are a chess player. Question: The two bishops on the board should be initially at which squares? Answer: {Let's think step by step}</p>
CF	<p>You are a chess player. You are playing a chess variant where the starting positions for knights and bishops are swapped. For each color, the knights are at placed that where bishops used to be and the bishops are now placed at where knights used to be. Question: In this chess variant, the two bishops on the board should be initially at which squares? Answer: {Let's think step by step}</p>

Table 15: CCC prompts for the chess task. **{Let's think step by step}** is added only if 0-shot CoT is used.

Mode	Prompt
Default	<p>You will be shown 12 cards. Each card has a figure and a number. A figure is a combination of a color, a shape, and a fill. Set of colors are: red , green , blue . Set of shapes are: squiggle , diamond , oval .</p> <p>Set of fills are: solid , striped , open .</p> <p>-THE RULE OF THE GAME- A GAME-SET is set of three cards: For each attribute, (color, shape, fill, number), the three cards should either be ALL the SAME or NONE the SAME (=ALL DIFFERENT, e.g. if 2 of the cards have the same value, and 1 of them has a different value, the set is NOT valid; for example, (blue, green, blue) is MIXED and does not satisfy any of the rule, whereas (oval, diamond, squiggle) is all different.</p> <p>Here is the board: (2 green oval open) ... [truncated]</p> <p>You can pick a set by typing the cards in the below format: First card: CARD1 Second card: CARD2 Third card: CARD3 Now remember the rule and tell me which three cards here constitutes a GAME-SET in the same format. I will give you 2 cards as a hint, and you tell me the third one. First card: (2 green oval open) Second card: (1 green diamond solid) {Let's think step by step.}</p>
CF	<p>You will be shown 12 cards. Each card has a figure and a number. A figure is a combination of a color, a shape, and a fill. Set of colors are: red , green , blue . Set of shapes are: squiggle , diamond , oval .</p> <p>Set of fills are: solid , striped , open .</p> <p>-THE RULE OF THE GAME- (This is not the original SET game. It has a tweaked rule.) In this version, a GAME-SET is a set of three cards: - For each figure attribute except the number (color, shape, fill), the three cards should either be ALL the SAME or NONE the SAME (=ALL DIFFERENT, e.g. if 2 of the cards have the same value, and 1 of them has a different value, the set is NOT valid; for example, (blue, green, blue) is MIXED and does not satisfy any of the rule, whereas (oval, diamond, squiggle) is all different. - But only for the number attribute, 2 of the cards should have the same number, and 1 of them should have a different number in order for the set to be valid.</p> <p>Here is the board: (2 green oval open) ... [truncated]</p> <p>You can pick a set by typing the cards in the below format: First card: CARD1 Second card: CARD2 Third card: CARD3 Now remember the rule and tell me which three cards here constitutes a GAME-SET in the same format. I will give you 2 cards as a hint, and you tell me the third one. First card: (2 green oval open) Second card: (1 green diamond solid) {Let's think step by step.}</p>

Table 16: Prompts for the SET task. **{Let's think step by step}** is added only if 0-shot CoT is used.

Mode	Prompt
Default	<p>Each card has a figure and a number. A figure is a combination of a color, a shape, and a fill. Set of colors are: red , green , blue . Set of shapes are: squiggle , diamond , oval .</p> <p>Set of fills are: solid , striped , open .</p> <p>-THE RULE OF THE GAME- A GAME-SET is set of three cards: For each attribute, (color, shape, fill, number), the three cards should either be ALL the SAME or NONE the SAME (=ALL DIFFERENT, e.g. if 2 of the cards have the same value, and 1 of them has a different value, the set is NOT valid; for example, (blue, green, blue) is MIXED and does not satisfy any of the rule, whereas (oval, diamond, squiggle) is all different.</p> <p>I will give you three cards from the board, and you will tell me whether this constitutes a GAME-SET.</p> <p>First card: (1 blue oval striped) Second card: (2 red squiggle striped) Third card: (3 green diamond striped)</p> <p>Is this a GAME-SET? {Answer with yes or no in the last line. Let's verify rules for each attribute step-by-step:}</p>
CF	<p>Each card has a figure and a number. A figure is a combination of a color, a shape, and a fill. Set of colors are: red , green , blue . Set of shapes are: squiggle , diamond , oval .</p> <p>Set of fills are: solid , striped , open .</p> <p>-THE RULE OF THE GAME- (This is not the original SET game. It has a tweaked rule.) In this version, a GAME-SET is a set of three cards: - For each figure attribute except the number (color, shape, fill), the three cards should either be ALL the SAME or NONE the SAME (=ALL DIFFERENT, e.g. if 2 of the cards have the same value, and 1 of them has a different value, the set is NOT valid; for example, (blue, green, blue) is MIXED and does not satisfy any of the rule, whereas (oval, diamond, squiggle) is all different. - But only for the number attribute, 2 of the cards should have the same number, and 1 of them should have a different number in order for the set to be valid.</p> <p>I will give you three cards from the board, and you will tell me whether this constitutes a GAME-SET.</p> <p>First card: (1 blue oval striped) Second card: (2 red squiggle striped) Third card: (3 green diamond striped)</p> <p>Is this a GAME-SET? {Answer with yes or no in the last line. Let's verify rules for each attribute step-by-step:}</p>

Table 17: CCC prompts for the SET experiments. **{bold text}** is added only if 0-shot CoT is used. Note that we removed the board information for simplicity as it is not required for this CCC test.

Base	Tests										CCC				
	w/o 0-CoT					w/ 0-CoT									
	8	9	10	11	16	8	9	10	11	16	8	9	10	11	16
# instances	1,000										200				
GPT-4	82.3	23.4	100.0	38.4	63.0	60.2	38.6	98.2	56.5	74.0	98.0	90.0	100.0	91.0	100.0
GPT-3.5	8.3	6.6	100.0	3.8	17.7	12.6	9.8	99.0	2.7	17.7	96.5	77.0	100.0	56.0	95.5
Claude	22.3	0.2	99.8	6.6	32.4	1.4	0.9	98.7	4.0	6.6	64.5	47.5	100.0	41.0	77.5
PaLM-2	6.4	2.2	98.7	3.4	23.4	1.1	0.6	82.2	0.5	1.2	51.5	53.5	100.0	72.0	93.5

Table 18: Results for the arithmetic task (in accuracy; %).

# digits	# shots	8	9	10	11	16
2	0	60.2	38.6	98.2	56.5	74.0
3	0	56.8	32.2	87.1	24.2	33.2
4	0	24.0	14.6	83.4	8.9	9.1
2	1	97.3	48.1	99.7	25.7	49.1
2	2	99.1	67.0	99.9	44.0	57.8
2	4	99.4	79.7	99.9	68.4	70.6
2	8	99.7	85.8	100.0	79.6	83.5
2	16	99.9	88.4	99.9	86.9	88.7

Table 19: Results for the arithmetic task for various analyses in §5 (in accuracy; %). Only for GPT-4 with 0-shot CoT.

# instances	Tests				CCC			
	w/o 0-CoT		w/ 0-CoT		w/o 0-CoT		w/ 0-CoT	
	Default	CF	Default	CF	Default	CF	Default	CF
# instances	113				750			
GPT-4	58.4	18.6	73.5	24.8	95.3	78.1	99.7	90.9
GPT-3.5	39.8	9.7	54.0	10.6	97.1	21.3	94.1	25.9
Claude	35.4	13.3	36.3	6.2	96.5	31.1	85.1	37.3

Table 20: Results for the programming execution task (in accuracy; %).

# instances	HumanEval (All)				HumanEval (Subset)			
	pass@1		pass@10		pass@1		pass@10	
	Default	CF	Default	CF	Default	CF	Default	CF
# instances	164				53			
GPT-4	87.4	68.2	95.3	83.4	82.5	40.5	93.3	64.9
GPT-3.5	73.8	41.8	88.4	67.6	68.9	25.1	81.0	45.8
Claude	53.7	39.6	78.1	64.2	47.6	15.7	74.0	41.9
PaLM-2	27.3	20.8	55.8	42.6	29.2	7.4	55.3	21.0

Table 21: Results for the programming generation task (in pass@1 and pass@10; %). We report both the results on the entire HumanEval dataset for comparability with other work, as well as the subset on which evaluating the original program under 1-based indexing would not pass the test cases. Figure 2 only showed the results on this subset.

Test Accuracy												
w/o 0-CoT						w/ 0-CoT						
SVO	SOV	VSO	VOS	OVS	OSV	SVO	SOV	VSO	VOS	OVS	OSV	
# instances						100						
GPT-4	88.0	63.0	66.0	63.0	68.0	68.0	76.0	66.0	69.0	70.0	70.0	68.0
GPT-3.5	72.0	51.0	60.0	39.0	41.0	64.0	50.0	51.0	63.0	44.0	27.0	51.0
Claude	55.0	58.0	65.0	62.0	59.0	56.0	51.0	57.0	62.0	62.0	62.0	59.0
PaLM-2	40.0	22.0	48.0	28.0	37.0	23.0	35.0	29.0	56.0	29.0	22.0	19.0

CCC												
w/o 0-CoT						w/ 0-CoT						
SVO	SOV	VSO	VOS	OVS	OSV	SVO	SOV	VSO	VOS	OVS	OSV	
# instances						100						
GPT-4	–	100.0	92.0	98.0	100.0	94.0	–	95.0	95.0	99.0	96.0	100.0
GPT-3.5	–	88.0	72.0	25.0	61.0	64.0	–	68.0	40.0	37.0	5.0	31.0
Claude	–	89.0	60.0	97.0	99.0	98.0	–	79.0	91.0	80.0	8.0	87.0
PaLM-2	–	7.0	8.0	96.0	100.0	62.0	–	28.0	29.0	91.0	94.0	70.0

Table 22: Results for the basic syntactic reasoning task (in accuracy; %).

Tests											
w/o 0-CoT				w/ 0-CoT		CCC					
Default		CF		Default			CF				
# instances				81		310					
GPT-4		93.8		74.1		98.8		82.7		97.4	
GPT-3.5		79.0		43.2		65.4		42.0		90.3	
Claude		27.2		16.0		40.7		17.3		55.2	
PaLM-2		84.0		66.7		88.9		74.1		70.6	

Table 23: Results for the logical reasoning task (in accuracy; %).

Tests Accuracy														
w/o 0-CoT								w/ 0-CoT						
Default	S-NS	S-WE	R90	R180	R270	Rand.	Default	S-NS	S-WE	R90	R180	R270	Rand.	
# instances			100					100						
GPT-4	79.0	57.0	29.0	34.0	6.0	22.0	34.0	98.0	71.0	23.0	24.0	9.0	13.0	13.0
GPT-3.5	87.0	56.0	32.0	27.0	12.0	17.0	15.0	82.0	66.0	36.0	27.0	29.0	22.0	22.0
Claude	86.0	51.0	72.0	35.0	45.0	15.0	51.0	85.0	50.0	71.0	30.0	49.0	11.0	39.0
PaLM-2	90.0	88.0	86.0	50.0	93.0	39.0	64.0	84.0	95.0	80.0	38.0	91.0	38.0	54.0

Tests Object-level Accuracy														
w/o 0-CoT								w/ 0-CoT						
Default	S-NS	S-WE	R90	R180	R270	Rand.	Default	S-NS	S-WE	R90	R180	R270	Rand.	
# instances			100					100						
GPT-4	86.0	74.3	55.7	56.0	34.0	53.0	61.7	99.0	85.3	57.0	49.7	36.0	46.3	46.0
GPT-3.5	92.3	77.7	62.3	54.7	41.3	42.3	47.0	92.7	82.7	64.0	53.0	50.7	54.7	53.3
Claude	93.7	75.3	87.7	65.7	70.3	46.7	76.0	91.7	74.7	86.3	63.0	73.0	44.0	69.3
PaLM-2	96.3	95.7	94.3	71.3	97.7	64.7	79.7	94.0	98.3	91.0	63.7	96.0	64.0	75.0

CCC														
w/o 0-CoT								w/ 0-CoT						
Default	S-NS	S-WE	R90	R180	R270	Rand.	Default	S-NS	S-WE	R90	R180	R270	Rand.	
# instances			100					100						
GPT-4	100.0	99.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
GPT-3.5	100.0	100.0	92.0	83.0	90.0	100.0	99.0	98.0	89.0	88.0	83.0	78.0	94.0	92.0
Claude	100.0	100.0	100.0	100.0	100.0	100.0	99.0	100.0	100.0	100.0	100.0	100.0	100.0	91.0
PaLM-2	100.0	100.0	100.0	100.0	100.0	39.0	100.0	100.0	100.0	100.0	100.0	100.0	59.0	100.0

Table 24: Results for the spatial reasoning task (in accuracy; %). The first section (Tests Accuracy) requires all 3 objects to be correctly placed. The second section (Test Object-Level Accuracy) refers to accuracy averaged over objects. ‘S’ denotes to swapping, ‘R’ denotes to rotation, ‘Rand.’ denotes to random permutation.

Human Evaluation Binary Classification Test Accuracy								
	w/o 0-CoT				w/ 0-CoT			
	Default	VFlip	R90	R180	Default	Flip	R90	R180
# instances	100							
GPT-4	82.0	57.0	59.0	53.0	86.0	51.0	47.0	52.0
GPT-3.5	31.0	19.0	9.0	8.0	46.0	7.0	5.0	8.0
Claude	42.0	13.0	15.0	13.0	34.0	15.0	30.0	12.0

CLIP Evaluation Multi-class Classification Tests Accuracy								
	w/o 0-CoT				w/ 0-CoT			
	Default	VFlip	R90	R180	Default	Flip	R90	R180
# instances	100							
GPT-4	58.0	30.0	23.0	31.0	53.0	26.0	33.0	28.0
GPT-3.5	20.0	9.0	7.0	8.0	32.0	4.0	6.0	6.0
Claude	19.0	11.0	9.0	8.0	18.0	9.0	13.0	8.0

CCC Accuracy								
	w/o 0-CoT				w/ 0-CoT			
	Default	VFlip	R90	R180	Default	Flip	R90	R180
# instances	100							
GPT-4	100.0	99.0	55.0	89.0	100.0	99.0	87.0	87.0
GPT-3.5	78.0	56.0	53.0	44.0	99.0	62.0	43.0	46.0
Claude	100.0	84.0	86.0	57.0	99.0	90.0	99.0	54.0

Table 25: Results for the drawing task (in accuracy; %). VFlip corresponds to vertical flipping, R90 and R180 correspond to rotation by 90 degrees and 180 degrees, respectively.

Accuracy for Objects w/o Canonical Orientation								
	w/o 0-CoT				w/ 0-CoT			
	Default	VFlip	R90	R180	Default	Flip	R90	R180
GPT-4	72.1	68.6	67.3	57.4	76.2	58.0	69.8	63.3
GPT-3.5	31.8	22.0	14.3	13.0	39.0	4.8	7.1	7.0
Claude	51.1	22.9	20.9	23.3	39.5	27.9	34.8	19.1

Accuracy for Objects w/ Canonical Orientation								
	w/o 0-CoT				w/ 0-CoT			
	Default	VFlip	R90	R180	Default	Flip	R90	R180
GPT-4	89.5	44.9	50.0	49.1	93.1	44.0	21.3	41.2
GPT-3.5	30.4	16.0	5.2	3.7	50.8	8.6	3.4	8.8
Claude	34.5	3.8	10.5	5.3	29.8	5.3	25.9	5.7

Table 26: Results for the drawing task, as measured by human evaluation accuracy (%), broken down by objects with or without a canonical orientation as judged by human annotators. If an object has a canonical orientation, such as the house in Figure 7, it is only considered correct if the orientation is correct.

Test Accuracy												
w/o 0-CoT						w/ 0-CoT						
Default	DAD-	FAD-	EBD-	ECD-	ECF-	Default	DAD-	FAD-	EBD-	ECD-	ECF-	
# instances	120											
GPT-4	47.5	22.5	11.7	18.3	7.5	1.7	42.5	25.8	12.5	24.2	14.2	10.0
GPT-3.5	30.8	4.2	5.0	5.8	3.3	1.7	30.8	5.0	5.0	0.8	1.7	1.7
Claude	5.0	0.8	0.8	0.8	0.8	1.7	10.0	5.8	4.2	5.8	5.0	0.8
PaLM-2	0.8	0.0	0.0	0.0	0.0	0.0	0.8	0.0	0.0	0.0	0.8	0.8

CCC												
w/o 0-CoT						w/ 0-CoT						
Default	DAD-	FAD-	EBD-	ECD-	ECF-	Default	DAD-	FAD-	EBD-	ECD-	ECF-	
# instances	18											
GPT-4	100.0	100.0	94.4	94.4	100.0	83.3	100.0	100.0	100.0	100.0	100.0	100.0
GPT-3.5	50.0	27.8	11.1	44.4	50.0	61.1	94.4	66.7	88.9	88.9	94.4	77.8
Claude	77.8	38.9	55.6	55.6	50.0	50.0	100.0	38.9	55.6	55.6	50.0	50.0
PaLM-2	33.3	33.3	33.3	38.9	50.0	38.9	33.3	33.3	50.0	38.9	38.9	38.9

Table 27: Results for the chord fingering task (in accuracy; %): guitar. Default corresponds to EADGBE. Counterfactuals show the first three strings (the remaining three strings, GBE, are the same).

Test Accuracy												
w/o 0-CoT						w/ 0-CoT						
Default	DAD-	FAD-	EBD-	ECD-	ECF-	Default	DAD-	FAD-	EBD-	ECD-	ECF-	
# instances	12											
maj triad	66.7	50.0	33.3	41.7	8.3	0.0	58.3	41.7	8.3	41.7	8.3	0.0
min triad	58.3	33.3	25.0	41.7	25.0	0.0	58.3	25.0	16.7	25.0	16.7	8.3
5	83.3	41.7	33.3	16.7	0.0	0.0	75.0	50.0	25.0	33.3	33.3	50.0
dom7	25.0	25.0	16.7	8.3	25.0	0.0	33.3	16.7	8.3	25.0	16.7	0.0
6	25.0	16.7	0.0	16.7	0.0	0.0	33.3	25.0	8.3	16.7	8.3	8.3
sus4	50.0	33.3	0.0	33.3	0.0	0.0	58.3	33.3	16.7	33.3	25.0	25.0
dim7	25.0	0.0	0.0	0.0	8.3	16.7	0.0	0.0	8.3	16.7	0.0	0.0
aug7	0.0	0.0	0.0	0.0	0.0	0.0	8.3	0.0	0.0	16.7	16.7	0.0
sus2	58.3	0.0	8.3	16.7	0.0	0.0	58.3	50.0	16.7	25.0	16.7	8.3
min7	83.3	25.0	0.0	8.3	8.3	0.0	41.7	16.7	16.7	8.3	0.0	0.0

Table 28: Results broken down by chords for the chord fingering task as analyzed in §5 (in accuracy; %): guitar, GPT-4. Default corresponds to EADGBE. Counterfactuals show the first three strings (the remaining three strings, GBE, are the same).

Test Accuracy										
	w/o 0-CoT					w/ 0-CoT				
	Default	FC-	AC-	BC-	BE-	Default	FC-	AC-	BC-	BE-
# instances	108									
GPT-4	39.8	1.9	1.9	2.8	0.9	20.4	2.8	16.7	11.1	10.2
GPT-3.5	14.8	0.0	2.8	3.7	0.0	6.5	1.9	2.8	1.9	0.0
Claude	0.0	1.9	0.0	0.0	0.9	6.5	0.0	2.8	1.9	2.8
PaLM-2	0.0	0.0	0.0	0.0	0.0	0.9	0.0	1.9	0.9	0.0

CCC										
	w/o 0-CoT					w/ 0-CoT				
	Default	FC-	AC-	BC-	BE-	Default	FC-	AC-	BC-	BE-
# instances	12									
GPT-4	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
GPT-3.5	33.3	41.7	91.7	41.7	100.0	91.7	83.3	100.0	75.0	75.0
Claude	58.3	50.0	50.0	41.7	33.3	100.0	100.0	75.0	66.7	50.0
PaLM-2	41.7	41.7	41.7	50.0	33.3	41.7	50.0	41.7	58.3	33.3

Table 29: Results for the chord fingering task (in accuracy; %): ukulele. Default corresponds to GCEA. Counterfactuals show the first two strings (the remaining two strings, EA, are the same).

	Tests				CCC			
	w/o 0-CoT		w/ 0-CoT		w/o 0-CoT		w/ 0-CoT	
	Default	CF	Default	CF	Default	CF	Default	CF
# instances	56	1120	56	1120	7	140	7	140
GPT-4	48.2	20.4	64.3	44.9	100.0	87.1	100.0	94.3
GPT-3.5	17.9	17.4	44.6	21.8	100.0	70.0	100.0	95.7
Claude	17.9	17.2	16.1	20.1	100.0	74.3	100.0	80.7
PaLM-2	28.6	19.5	19.6	19.3	100.0	48.6	100.0	47.1

Table 30: Results for the melody retrieval task (in accuracy; %). Default corresponds to C major, and CF corresponds to other keys.

	Tests				CCC			
	w/o 0-CoT		w/ 0-CoT		w/o 0-CoT		w/ 0-CoT	
	Default	CF	Default	CF	Default	CF	Default	CF
# instances	8	160	8	160	1	20	1	20
1	87.5	63.1	87.5	70.0	100.0	95.0	100.0	95.0
2	87.5	10.6	62.5	46.9	100.0	95.0	100.0	100.0
3	12.5	11.9	62.5	51.2	100.0	80.0	100.0	90.0
4	25.0	21.2	62.5	42.5	100.0	100.0	100.0	85.0
5	50.0	3.1	37.5	35.0	100.0	95.0	100.0	100.0
6	37.5	5.0	75.0	38.8	100.0	75.0	100.0	95.0
7	37.5	27.5	62.5	30.0	100.0	70.0	100.0	95.0

Table 31: Results broken down by n for the melody retrieval task as analyzed in §5 (in accuracy; %): GPT-4. Default corresponds to C major, and CF corresponds to other keys.

	Tests				CCC			
	w/o 0-CoT		w/ 0-CoT		w/o 0-CoT		w/ 0-CoT	
	Default	CF	Default	CF	Default	CF	Default	CF
# instances	400				120			
GPT-4	73.8	50.0	87.8	53.6	100.0	100.0	100.0	100.0
GPT-3.5	60.1	54.4	60.2	48.9	85.8	90.0	91.7	93.3
Claude	59.2	50.0	60.2	50.4	100.0	50.0	76.7	100.0
PaLM-2	48.0	51.5	48.5	51.2	61.7	23.3	50.0	23.3

Table 32: Results for the chess task with 4 moves (in accuracy; %). CF refers to the setting where the initial positions of knights and bishops are swapped. We generate a balanced classification problem with 400 openings via procedure generation.

	Tests				CCC			
	w/o 0-CoT		w/ 0-CoT		w/o 0-CoT		w/ 0-CoT	
	Default	CF	Default	CF	Default	CF	Default	CF
# instances	100				100			
GPT-4	100.0	21.0	100.0	61.0	89.0	74.0	100.0	96.0
GPT-3.5	73.0	4.0	37.0	7.0	68.0	55.0	77.0	78.0
Claude	55.0	21.0	64.0	35.0	92.0	62.0	59.0	65.0
PaLM-2	55.0	17.0	62.0	13.0	67.0	68.0	47.0	46.0

Table 33: Results for the SET game (in accuracy; %).

	Tests					
	Default (c=1)	CF (c=1)	Default (c=2)	CF (c=2)	Default (c=3)	CF (c=3)
# instances	100					
GPT-4	100.0	61.0	24.0	6.0	15.0	3.0
GPT-3.5	37.0	7.0	7.0	0.0	1.0	0.0
Claude	64.0	35.0	10.0	4.0	5.0	1.0
PaLM-2	62.0	13.0	10.0	1.0	3.0	1.0

Table 34: Breakdown for the SET game test results (with 0-CoT) when the model needs to find different number of cards (c) in a SET as analyzed in §5 (in accuracy; %).

In the beginning, you need to specify:

- Canonical Orientation? Use 1 if you believe the object has one or more canonical orientation, such as a “house”. Use 0 if it doesn’t, such as a “pen”.

If an object is considered to have a canonical orientation, it is ONLY CORRECT if it’s displayed in the right orientation.

For each image, you need to specify the following:

- Correct? (Input: 0 or 1) Determine if the object is correct. Use 0 for False and 1 for True.
- Control? (Input: 0 or 1 or 2) Determine if the control has been passed. Use 1 if there is an almost horizontal line at the TOP. Use 0 if there is a standalone line but is either not horizontal or is at the center or bottom (even if the object appears beneath it). Use 2 if there is no such line.

Examples: *[some examples, omitted here for brevity]*

Figure 8: The drawing evaluation instruction given to our human annotators. “Control” was a prior name for our CCC in this project.