LLMs and the Abstraction and Reasoning Corpus: Successes, Failures, and the Importance of Object-based Representations

Anonymous authors
Paper under double-blind review

Abstract

Can a Large Language Model (LLM) solve simple abstract reasoning problems? We explore this broad question through a systematic analysis of GPT on the Abstraction and Reasoning Corpus (ARC) (Chollet, 2019), a representative benchmark of abstract reasoning ability from limited examples in which solutions require some "core knowledge" of concepts such as objects, goal states, counting, and basic geometry. GPT-4 solves only 13/50 of the most straightforward ARC tasks when using textual encodings for their two-dimensional input-output grids. Our failure analysis reveals that GPT-4's capacity to identify objects and reason about them is significantly influenced by the sequential nature of the text that represents an object within a text encoding of a task. To test this hypothesis, we design a new benchmark, the 1D-ARC, which consists of one-dimensional (array-like) tasks that are more conducive to GPT-based reasoning, and where it indeed performs better than on the (2D) ARC. To alleviate this issue, we propose an object-based representation that is obtained through an external tool, resulting in nearly doubling the performance on solved ARC tasks and near-perfect scores on the easier 1D-ARC. Although the state-of-the-art GPT-4 is unable to "reason" perfectly within non-language domains such as the 1D-ARC or a simple ARC subset, our study reveals that the use of object-based representations can significantly improve its reasoning ability.

1 Introduction

It has been recently claimed that Large Language Models (LLMs) such as GPT-4(OpenAI, 2023a) exhibit "sparks of artificial general intelligence" (Bubeck et al., 2023). As a result, the impressive question-answering and text generation abilities of pre-trained LLMs are already being deployed in rather consequential e-commerce and educational settings[1]. If LLMs are to be used to reliably solve complex, noisy, real-world problems, one would expect them to be capable of reasoning in simple, unambiguous, idealized settings. By "reasoning", we here mean "using evidence, arguments, and logic to arrive at conclusions or make judgments", as defined in (Huang and Chang, 2022). While the performance of LLMs on arithmetic and language-based commonsense reasoning benchmarks has been the subject of recent analyses (see for example Section 4.1 of (Huang and Chang, 2022) for a brief survey), it is unclear whether LLMs exhibit the ability to generate abstract concepts based on a handful of "training" samples (Odouard and Mitchell, 2022).

To quantitatively measure the gap between machine and human learning, the Abstraction and Reasoning Corpus (ARC) was introduced in (Chollet, 2019). The author advocates for leveraging human-level intelligence as a frame of reference for evaluating general intelligence. To that end, he draws upon the work of developmental psychologists (Spelke and Kinzler, 2007) on the theory of Core Knowledge to determine axes along which human-like intelligence should be measured. Core Knowledge identifies four broad categories of innate assumptions that form the foundation of human cognition:

- **Objectness:** The ability to perceive the surroundings as consisting of cohesive, persistent, and non-interpenetrating objects.

---

[1] `https://openai.com/blog/introducing-chatgpt-and-whisper-apis`

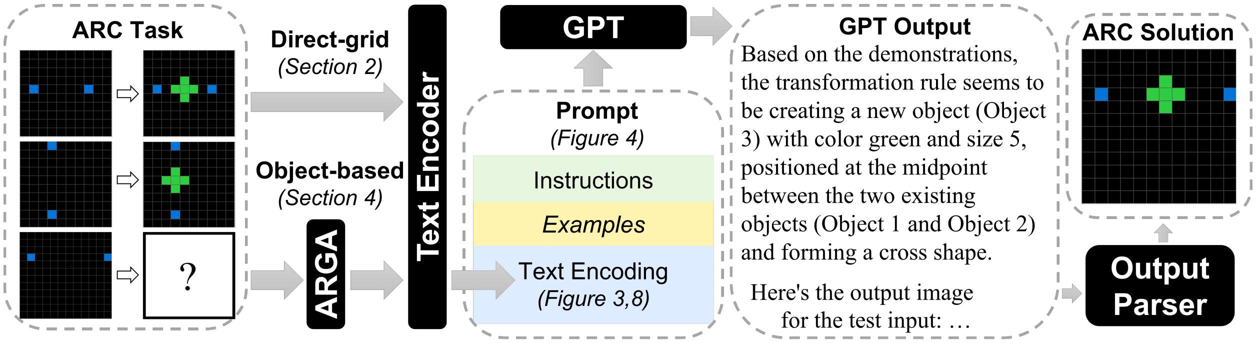

ARC Task
?
Direct-grid
(Section 2)
Object-based
(Section 4)
ARGA
Text Encoder
GPT
Prompt
(Figure 4)
Instructions
Examples
Text Encoding
(Figure 3,8)
GPT Output
Based on the demonstrations, the transformation rule seems to be creating a new object (Object 3) with color green and size 5, positioned at the midpoint between the two existing objects (Object 1 and Object 2) and forming a cross shape.
Here's the output image for the test input: …
ARC Solution
Output Parser

Figure 1: **Example of solving an ARC task with GPT.** An "ARC Task" consists of a set of training input-output pairs followed by a test input for which GPT should produce a correct output. To do so, a prompt is created. It includes high-level instructions on what GPT should do, optionally with additional in-context examples. A text encoding of the ARC task of interest is also included in the prompt. The encoding may be a direct representation of the 2-dimensional grids or an object representation produced by an external ARC solver, ARGA. GPT must then "reason" about the prompt to produce an answer. The output is then parsed and checked for correctness.

- **Agentness and goal-directedness:** The tendency to perceive certain objects in the environment as intentional agents with goals, capable of contingent and reciprocal actions, while distinguishing them from inanimate objects.
- **Numerical knowledge:** Innate knowledge of abstract number representations for small numbers and the concepts of addition, subtraction and comparison between those numbers.
- **Elementary geometry and topology:** Knowledge of distance and basic 2D and 3D shapes.

The ARC, a benchmark of 1,000 image-based reasoning tasks, is thus proposed as a test of the above four core knowledge systems in humans or AI systems. Each task requires the production of an output image given a specific input, with 2 to 5 input-output image pairs provided as training instances to "learn" the underlying procedure (Figure 2). The training inputs are different from the actual test input, though they are solvable using the same (unspecified) procedure. Crucially, no acquired knowledge outside of the aforementioned priors is required to solve these tasks. Note that although these priors are explicitly described, the ARC tasks remain completely open-ended: objects can have different shapes and colors and form various relations with one another, and the grid size can also vary between tasks. This feature makes these problems not amenable to solving through search. This is in contrast to games like Go and chess (Silver et al., 2018; 2017) where the search space is large but the set of moves is finite and fixed. In fact, so far the approaches that have employed a heuristic search via a predetermined set of transformations have all fallen short of generalizing to the hidden tasks; see Section 5.

Given the vast corpus of human knowledge that LLMs are typically trained on, one might wonder whether they could have acquired the priors listed above. Towards answering this question, we conduct a comprehensive study of GPT-3.5 and GPT-4 on the ARC. We contribute three high-level findings that we believe shed light on some intrinsic limitations of the LLM framework and, to some extent, how to resolve them:

1. **GPT fails on simple ARC tasks:** Using pure text encodings (Figure 3) of tasks such as those in Figure 2, GPT-4 can only solve 13/50 of the simplest ARC tasks (Section 2). We conduct a failure analysis which reveals that the culprit is the LLM's inability to maintain "object cohesion" across the lines of text that represent the ARC image grids.
2. **GPT does better when objects can be easily detected in text:** We hypothesize that the issue raised in point 1 is tied to the two-dimensional nature of the ARC grids. In Section 3, we introduce the 1D-ARC benchmark, a set of ARC-like tasks that can be represented as a single line of text. Relative to the 50 ARC tasks, GPT performs better on 1D-ARC, but is far from perfect.

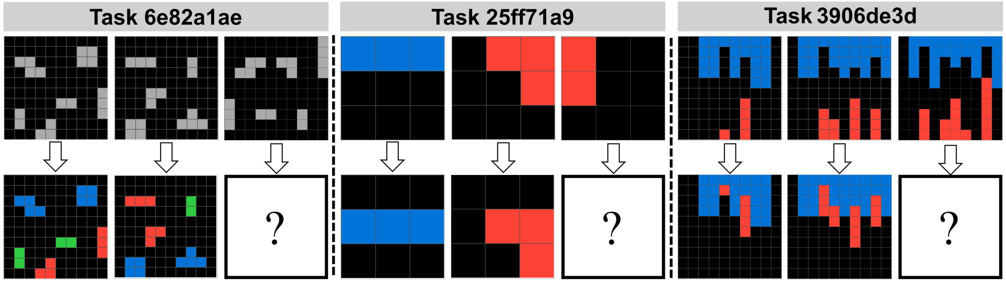


Figure 2: **Sample ARC Tasks.** Three tasks (separated by dashed lines) are shown. For a given task, each column contains one example input-output pair. The first two columns contain the "training" instances and the third column contains the "test" instance. The goal is to use the training instances to solve the test instance. The left task ("Recolor by size") requires recoloring the grey objects to green, red, or blue based on if their size is 2, 3 or 4. The middle task ("Static movement") requires moving the non-black object down 1 pixel. The right task ("Dynamic movement") requires moving the red objects up towards the blue objects until they make contact.

3. **An object-based representation boosts GPT's performance significantly:** Leveraging the object-centric graph abstractions of the ARC solver Abstract Reasoning with Graph Abstractions (ARGA) (Xu et al., 2023), we provide the LLMs with a more structured object-based representation of the input-output ARC grids (Section 4). This results in a significant jump in performance, where GPT-4 solves 23 instead of 13/50 tasks, and achieves near-perfect scores on many 1D-ARC task types. Further experiments on additional datasets such as the full ARC training set and the Mini-ARC, although not the focus of this paper, produce a similar pattern.

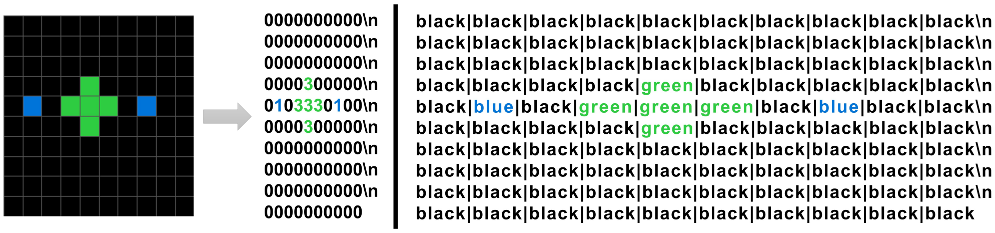
Given the increasing interest from the artificial intelligence community in the reasoning capabilities of pre-trained LLMs and the unique characteristics of the ARC (and 1D-ARC), we believe that our work contributes to research on imbuing LLMs with such capabilities. We demonstrate that the use of an external tool that produces appropriate representations is crucial. We hope that our experimental design on the ARC, the new 1D-ARC dataset, and the integration of a domain-specific external tool for improved representation will be useful in generating new ideas at the intersection of LLMs and reasoning. Our code and data are available in the supplementary materials and will be open-sourced upon publication.

2 A first attempt at solving ARC with an LLM

The task of solving the ARC with an LLM necessitates the encoding of two-dimensional (2D) input-output images using a textual representation[2]. A text-encoded ARC task is incorporated into an LLM prompt, which then generates the solution. This section proposes a straightforward pipeline and evaluates it before mining the results to understand failure modes.

2.1 Textual encoding

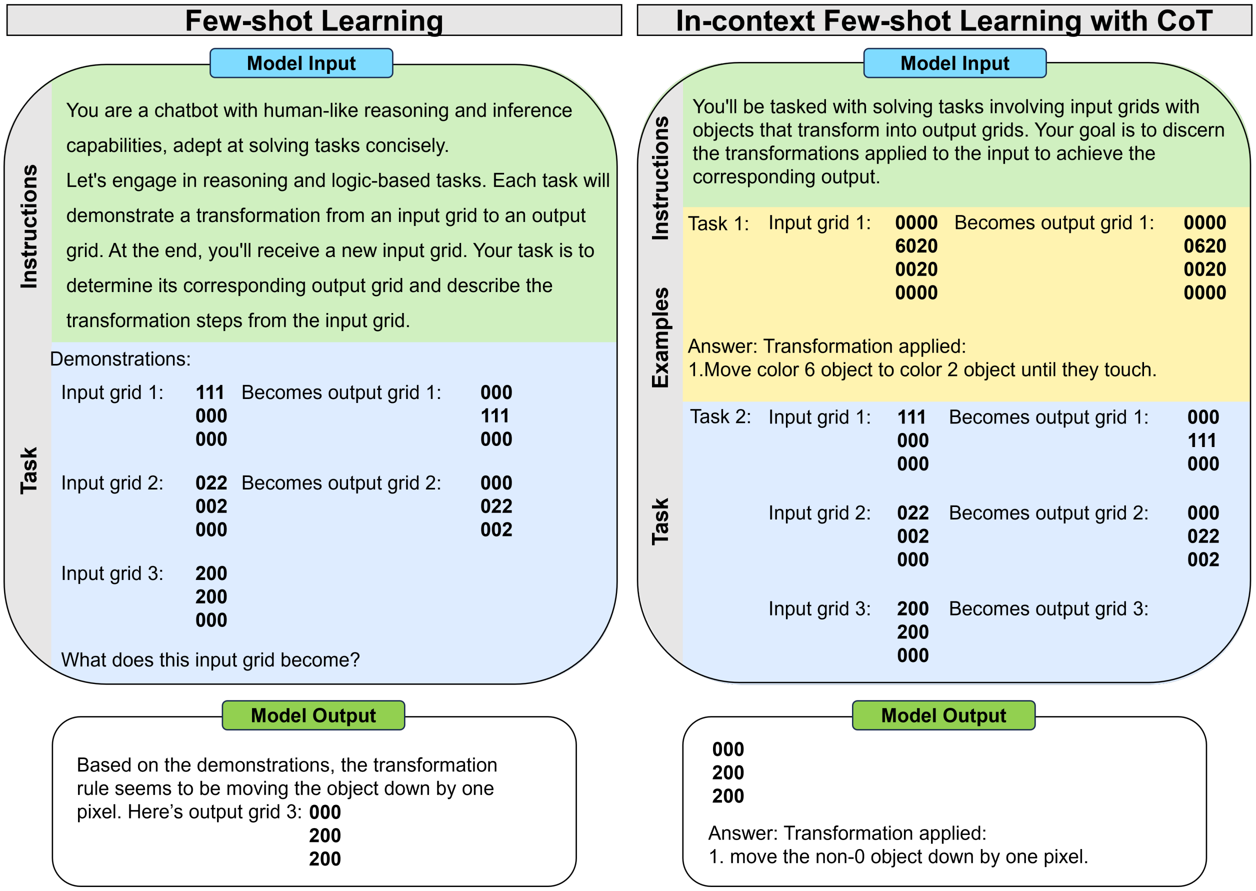
A 2D grid with colored pixels can be directly encoded into text by representing each pixel's color either numerically (using values from 0 to 9, each representing one of the ten colors) or with color descriptors (e.g., "blue", "green", "black"). A delimiter delineates between adjacent pixels and "newline" characters were used to separate the rows in an image. We assessed the impact of different delimiters (“`,`”, “`|`”, or no delimiter) on LLM performance. Figure 3 provides two visual examples of this *direct-grid encoding*.



```
0000000000\n
0000000000\n
0000000000\n
0000300000\n
0103330100\n
0000300000\n
0000000000\n
0000000000\n
0000000000\n
0000000000
```

```
black|black|black|black|black|black|black|black|black|black\n
black|black|black|black|black|black|black|black|black|black\n
black|black|black|black|black|black|black|black|black|black\n
black|black|black|black|green|black|black|black|black|black\n
black|blue|black|green|green|green|black|blue|black|black\n
black|black|black|black|green|black|black|black|black|black\n
black|black|black|black|black|black|black|black|black|black\n
black|black|black|black|black|black|black|black|black|black\n
black|black|black|black|black|black|black|black|black|black\n
black|black|black|black|black|black|black|black|black|black
```

Figure 3: **Visualization of direct-grid encodings.** Left: each pixel is represented with a number corresponding to the pixel color, with no delimiters. Right: each pixel is represented with the color descriptor, separated by the delimiter "|". The text string has been formatted for easier reading.



You are a chatbot with human-like reasoning and inference capabilities, adept at solving tasks concisely.

Let's engage in reasoning and logic-based tasks. Each task will demonstrate a transformation from an input grid to an output grid. At the end, you'll receive a new input grid. Your task is to determine its corresponding output grid and describe the transformation steps from the input grid.

Demonstrations:

Input grid 1: 111 000 000 Becomes output grid 1: 000 111 000

Input grid 2: 022 002 000 Becomes output grid 2: 000 022 002

Input grid 3: 200 200 000

What does this input grid become?

Based on the demonstrations, the transformation rule seems to be moving the object down by one pixel. Here's output grid 3: 000 200 200

You'll be tasked with solving tasks involving input grids with objects that transform into output grids. Your goal is to discern the transformations applied to the input to achieve the corresponding output.

Task 1: Input grid 1: 0000 6020 0020 0000 Becomes output grid 1: 0000 0620 0020 0000

Answer: Transformation applied:
1.Move color 6 object to color 2 object until they touch.

Task 2: Input grid 1: 111 000 000 Becomes output grid 1: 000 111 000

Input grid 2: 022 002 000 Becomes output grid 2: 000 022 002

Input grid 3: 200 200 000 Becomes output grid 3:

000
200
200

Answer: Transformation applied:
1. move the non-0 object down by one pixel.

Figure 4: **Example prompts.** Left: Few-shot Learning. Right: In-context Few-shot Learning with CoT. The prompt texts have been formatted for easier reading (for example, the "Task" string on the left is provided to the LLM as "Demonstrations:\nInput Grid 1: 111\n000\n000...").

2.2 Prompting and strategy

After encoding ARC images into text, we incorporate the latter into prompts that instruct the LLM to solve the task at hand. We explored two single-stage strategies for prompting the LLM.

---

[2] At the time of our research, GPT-4's vision API, which enables the model to accept image inputs, was not yet available. Consequently, this paper does not focus on the multimodal capabilities of GPT-4. However, we provide glimpses into these capabilities and their potential in Section 6 and Appendix G.

Table 1: **Direct-grid variants, performance comparison.** Each row corresponds to a variant of a direct-grid encoding. Each column corresponds to a combination of a prompting method with either GPT-3.5 or GPT-4. GPT solutions were obtained through OpenAI's API with temperature set to 0. The values correspond to the number of tasks, out of 50, solved by each method; higher is better and top-performers are bolded.

Direct-grid encoding		Few-shot		In-context Few-shot w/ CoT	
Pixel	Delimeter	GPT-3.5	GPT-4	GPT-3.5	GPT-4
Number	n/a	3	5	2	9
Number	|	4	11	5	12
Word	,	3	12	5	**13**
Word	|	4	8	2	**13**

Few-shot learning: By design, an ARC task is a few-shot learning task, providing a handful of training examples for generating a solution for a test example. Therefore, adopting the few-shot learning strategy is the most straightforward and intuitive initial approach for leveraging the inherent structure of ARC tasks. The prompt created using this strategy has two main sections: "instructions" and "task". The "instructions" section outlines the nature of an ARC task and the expected behavior of the LLM; it is the same across all tasks. The "task" section provides information about the single ARC task of interest, including its few-shot examples. This approach, in line with the classic few-shot learning concept, encourages the LLM to leverage the provided examples to solve the task. An example of this prompting strategy can be found in Figure 4.

In-context few-shot learning with chain-of-thought: Building on the few-shot learning strategy and drawing inspiration from chain-of-thought (CoT) prompting introduced in (Wei et al., 2022), we investigate a natural combination thereof. This approach enriches the learning context for the LLM by augmenting the original prompt with an "examples" section, which includes two simple ARC-like tasks—different from the actual task of interest—and their step-by-step solutions. This strategy not only leverages the inherent task examples but also provides a stable learning base of *in-context examples* for the LLM, encouraging a CoT response. As such, it assesses the LLM's capacity to generalize and apply knowledge acquired from a limited set of contextual examples to solve a similar task. An example prompt using this approach can be found in Figure 4.

2.3 Results

Given that the most advanced ARC solvers achieve only a 30% accuracy on the hidden test set of the ARC, we strategically selected a subset of 50 ARC tasks. These tasks were among the "easiest", allowing our resources to be more efficiently allocated for in-depth experimentation. Our goal was to upper-bound the LLM's performance on the ARC as a whole; note that we do discuss results on all 400 ARC training tasks in the following paragraph and in Section 4.2, but we restrict much of the analysis in this paper to the selected 50 tasks. We define "easy" tasks as those that have been previously addressed using the symbolic search-based method, ARGA (Xu et al., 2023). This designation stems from the fact that ARGA's implementation confines its solution space to 15 functions, in contrast to state-of-the-art models which can have a more expansive function set, with the Kaggle first-place solution, for example, encompassing 42 functions. Thus, studying this subset provides a clearer understanding of how a purely search-based solver, with a restricted function set, tackles ARC tasks.

The top-performing pixel representation and delimiter combination (Word + "|") solves only 13 out of the 50 tasks, as shown in Table 1. Using the top-performing representation, we assessed the complete ARC training set and GPT-4 managed to solve 81 out of the 400 tasks. Furthermore, on the Mini-ARC dataset—a simplified version of ARC detailed in Section 5—this approach solved 35 out of the 149 tasks. These accuracy rates are consistent with the performance observed on the subset of 50 tasks. In the following section, we will delve into the reasons why the LLM struggled with these tasks given that they are easy for a non-LLM method.

2.4 Analysis

We started our analysis by extracting key attributes such as pixel and color counts from each ARC task. We then applied logistic regression to explore potential relationships between these features and the performance of the LLM.

An intriguing finding from our analysis is that the number of colored pixels in a *test* image is associated with a notable negative coefficient, indicating a potential inverse relationship with the LLM's ability to solve tasks. Since a set of adjacent colored pixels often corresponds to an object in the ARC, this finding suggests that tasks with fewer objects are *more likely* to be solved by the LLM. Conversely, we find a positive coefficient associated with the average number of colored pixels in *training* images, implying a possible positive correlation with task solvability. This could suggest that more colored pixels in training images provide more learning material for the LLM, potentially improving performance. The full set of features studied can be found in Appendix E.

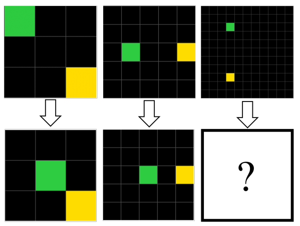
A closer examination of the tasks that GPT solved correctly using the direct-grid approach reveals some interesting patterns in the reasoning provided by the model. Out of the 13 tasks that were correctly solved, only three tasks were accompanied by the correct reasoning steps. Surprisingly, for some tasks, GPT did not provide any reasoning at all, despite the presence of reasoning examples within the In-context Few-shot Learning with CoT prompts. This inconsistency in the application of reasoning illustrates a possible gap in GPT's understanding and application of the reasoning process, which further complicates the task of solving ARC problems. An example of a task where the reasoning provided by the model was incorrect despite achieving the correct output is illustrated in Figure 5. Further examples can be found in Appendix D.

2.5 Object cohesion

To further understand the limitations of GPT on ARC tasks, we explored the concept of *object cohesion* in text, defined as the *"ability to parse grids into* 'objects' *based on continuity criteria including color continuity or spatial contiguity, and the ability to parse grids into zones, partitions"*(Chollet, 2019). Object cohesion is an integral part of human cognition (Spelke and Kinzler, 2007) and is assumed to be a significant part of the Core Knowledge priors required for ARC solving (Chollet, 2019).

Our objective was to investigate how the textual representation of objects influences GPT's problem-solving capacity. Given that the initial identification and abstraction of objects are pivotal in resolving the ARC (Acquaviva et al., 2022), understanding the impact of textual object depiction on the performance of language models is critical. We discovered that GPT's performance deteriorates significantly when objects are not sequentially represented within the text. To further demonstrate this, we selected tasks with clear horizontal or vertical objects and manipulated them to adopt the opposite orientation. A visualization of the difference between sequential and non-sequential object representation can be found in Figure 6.

For each "horizontal" or "vertical" original ARC task, we generated a rotated version and compared performance in both the horizontal and vertical configurations. An example is visualized in Figure 6 with more examples shown in Appendix C. The results in the leftmost part of Table 2 show a significant performance



Direct-grid approach:
Move color green object 1 pixel down
and color yellow object 1 pixel up.

Object-based approach :
Move color 3 object 1 pixel towards
color 4 object.

Figure 5: **Reasoning provided by GPT-4 for an example task.** Both approaches produced the correct output grid. The direct-grid approach produced the wrong reasoning while the object-based approach produced the correct one.

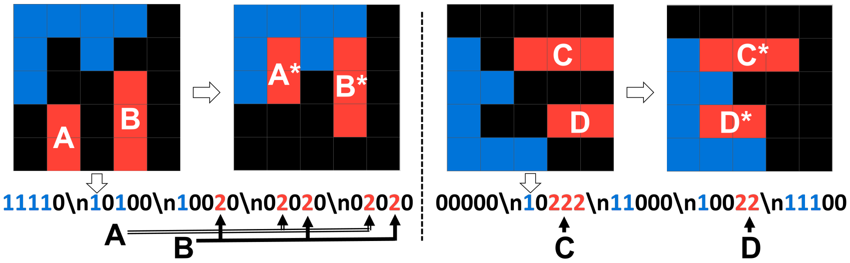

A
B
A*
B*
C
D
C*
D*
11110\n10100\n10020\n02020\n02020
A
B
00000\n10222\n11000\n10022\n11100
C
D

Figure 6: **Object cohesion analysis: two example tasks and their textual representations.** Generated based on ARC task seen in Figure 2 (Right), the two tasks are identical modulo the 90-degree rotation. Left: objects A and B are vertical and become non-sequential when represented in text. Right: objects C and D are horizontal and become sequential when represented in text.

drop in the vertical case, reinforcing our hypothesis. It is evident that GPT struggles with object cohesion when objects are not sequentially arranged within the text. This insight not only deepens our understanding of the model's limitations but also guides us toward potential solutions. In the following section, we delve further into this finding by generating a new dataset that guarantees object sequentialness, and assessing the performance of LLMs on this dataset.

3 Does reduced task dimensionality improve LLM performance?

We introduce 1D-ARC, a novel variation on the original ARC that reduces its dimensionality to facilitate future research and provide a more approachable benchmark for LLMs. The 1D-ARC maintains the same Core Knowledge priors as the ARC but restricts the dimensionality of the input and output images to one dimension. Consequently, the images comprise only a single row of pixels, significantly reducing the complexity of tasks and enabling all objects to be represented within a single sequence. This modification effectively removes the challenge of maintaining object cohesion in non-sequential text. We visualize some example tasks in Figure 7 but the visualization for the full dataset is included in Appendix A.

The 1D-ARC dataset was strategically designed to adapt transformation types from the original ARC dataset to a one-dimensional format. This methodology effectively preserves the core knowledge priors inherent to the original ARC. An example of the design and generation process for a 1D-ARC task is illustrated in Appendix B.

Our data generators have been developed to be capable of creating a variety of 1D-ARC tasks. They rely on task-specific parameters such as the maximum width of the 1D sequence, the maximum number of objects, and the maximum size of the objects. This parametric approach ensures tasks can originate from the same foundational concept or transformation yet manifest with varying complexities.

These generators can be seen as "1-concept" in design, each tailored for a specific transformation. However, their modular construction allows for the combination of multiple "1-concept" generators sequentially, leading to multi-concept tasks. As an example, a task could first necessitate the relocation of objects, then require a color transformation of these objects. This layered approach augments the scope and intricacy of the tasks, serving as a comprehensive evaluation of LLM capabilities.

3.1 Results

We used the best-performing prompts from Table 1 for the direct-grid approach and documented the results in the rightmost part of Table 2. Notably, the direct-grid encoding shows a relative improvement in performance on the 1D-ARC as compared to the original ARC. This implies that reducing both task space complexity and the spatial dimensionality of the input-output pairs enhances the LLM's ability to parse and reason with the encoded information.

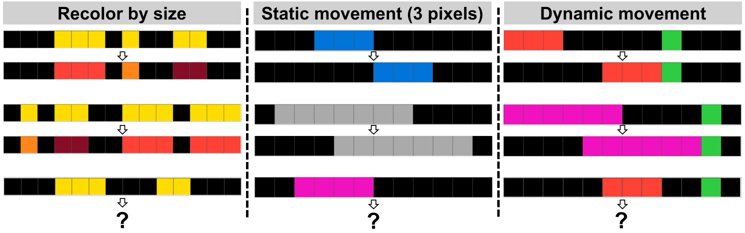


Figure 7: **Example tasks from the 1D-ARC dataset.** Each task is inspired by an ARC task; see Figure 2. From left to right: Recolor by size, Static movement by 3 pixels, Dynamic movement (move the block on the left until it touches the green pixel).

Table 2: **Results for direct-grid approach.** The number of solved tasks is out of 50. The first column is for the 50 tasks from the ARC. The second block of 5 columns is for some 1D-ARC task types; results on the full 1D-ARC can be found in Appendix A. The third block is for three task types (Fill, Move, Pile) with horizontal (H) and vertical (V) variants.

LLM	ARC Subset	Move 1 Pixel	Move 3 Pixels	Move Dynamic	Recolor by Size	Denoise	Fill		Move		Pile	
							H	V	H	V	H	V
GPT-3.5	2	10	7	6	2	13	2	0	0	0	2	0
GPT-4	13	33	12	11	14	30	46	1	12	0	32	0

Even with the significantly simpler 1D-ARC, there is still much room for improvement in performance. While GPT-4 is able to solve some tasks more effectively, it still falls short on others. This finding suggests that providing a sequential representation of objects in text alone may not be sufficient for GPT to effectively solve ARC tasks.

In light of these findings, we next explored the benefits of employing an external tool to perform object abstraction for GPT, thereby completely removing the challenge of object cohesion in text.

4 Enhancing LLM performance with an object-based representation

To address the challenges we have identified thus far and to enhance GPT's performance, we propose the integration of an external tool to aid in object representation during the ARC task-solving process. More specifically, we leverage the ARGA algorithm (Xu et al., 2023) to execute object abstraction before prompting GPT for the solution.

4.1 Object-based textual representation

ARGA is a non-learning approach that aims to solve ARC by first abstracting the images into graph representations and then conducting a search within a Domain-Specific Language (DSL) defining possible changes to the graphs to identify the solution. We leverage the first component of ARGA to acquire a *graph representation* of the images. These graph representations, in which each node (or vertex) corresponds to an object in the image grid and each edge represents relationships between the objects, are subsequently encoded into object-oriented text representations. It is worth noting that ARGA provides a suite of hand-designed abstraction methods to cater to different tasks. In our case, we apply the "best-fit" abstraction, utilizing the abstraction that ARGA deems optimal for generating the solution for each task. Essentially, our objective is to evaluate the effectiveness of an abstraction mechanism that excels at object abstraction. After transforming the images into graphs, we examine two textual representations:

Table 3: **Object-based variants, performance comparison.** The values correspond to the number of tasks solved by each method, out of 50; higher is better and top-performer is bolded.

Object-based encoding	Few-shot		In-context Few-shot w/ CoT	
	GPT-3.5	GPT-4	GPT-3.5	GPT-4
Object Descriptors	9	16	5	20
Object Descriptors with Edge	5	18	4	12
Object JSON	8	21	6	**23**
Object JSON with Edge	5	19	4	16

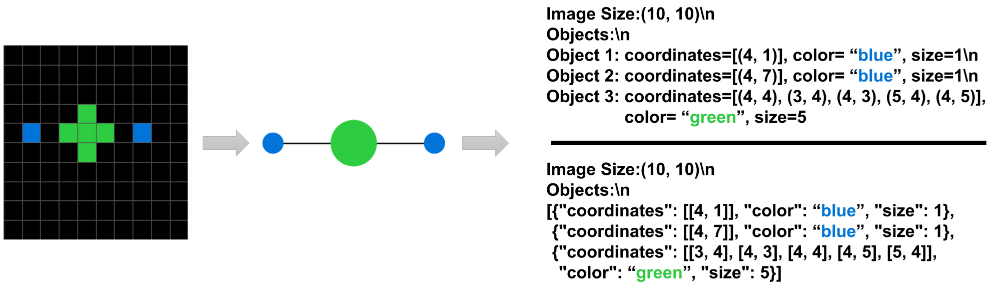
Object Descriptors: This encoding technique presents a list of objects, each corresponding to a node in the graph and its associated attributes; see Figure 8 (Top). It offers a clear and intuitive representation of the image as a set of distinct objects, each carrying its own properties.

Object JSON: On the other hand, the Object JSON encoding method provides a more structured representation of the graph; see Figure 8 (Bottom). This approach involves constructing a JSON list that encapsulates nodes and their corresponding attributes from the graph. The inherent organization of this format simplifies parsing and processing for the LLM, facilitating efficient extraction of pertinent information and relationships between the nodes.

Each encoding approach is further explored with an additional variant that includes edge information from the graph. In ARGA, edge information is utilized to identify relations between objects, as certain transformations applied to objects depend on other objects. For instance, an operation might involve recoloring an object to match the color of its neighbor. In the context of our study, our aim is to investigate whether the inclusion of edge information in the textual representation augments GPT's ability to solve ARC tasks.

4.2 Results

We leveraged the prompting methods outlined in Section 2.2 in combination with our proposed object-based textual representations, replacing the direct-grid encoding. The results, presented in Table 3, show a marked improvement, with the success rate increasing from 13/50 tasks to 23/50 tasks on the ARC subset. Table 4 also shows that the previously observed performance gap between horizontal and vertical tasks in Table 2 is eliminated with the object abstraction, confirming our hypothesis that GPT's challenges with object cohesion in non-sequential text were the root cause. The orientation of objects becomes inconsequential, as desired. An even bigger performance boost is observed for the 1D-ARC, where GPT-4 achieves 50/50 on some task types. Comparatively, on the complete ARC training set, the new method allowed GPT-4 to solve 97 out of the 400 tasks, up from 81 previously. Likewise, on the Mini-ARC dataset, performance improved to 45 out of 149 tasks, up from 35. These results underscore the value of augmenting the LLM with an external tool that provides an appropriate representation, particularly when it comes to ARC tasks.



```
Image Size:(10, 10)\n
Objects:\n
Object 1: coordinates=[(4, 1)], color= “blue”, size=1\n
Object 2: coordinates=[(4, 7)], color= “blue”, size=1\n
Object 3: coordinates=[(4, 4), (3, 4), (4, 3), (5, 4), (4, 5)],
          color= “green”, size=5
```

```
Image Size:(10, 10)\n
Objects:\n
[{"coordinates": [[4, 1]], "color": “blue”, "size": 1},
 {"coordinates": [[4, 7]], "color": “blue”, "size": 1},
 {"coordinates": [[3, 4], [4, 3], [4, 4], [4, 5], [5, 4]],
  "color": “green”, "size": 5}]
```

Figure 8: **Visualization of object-based textual encodings.** The 2D grid image is first transformed into a graph representation using ARGA. Then, the graph is encoded using the object descriptors representation (Top) or the object JSON representation (Bottom).

Table 4: **Results for object-based approach.** In addition to the caption of Table 2, the numbers in parentheses are the ratios of number of tasks solved with the object-based approach to the number of tasks solved by the direct-grid approach (Table 2); values larger than 1× indicate an increase of that factor in tasks solved with the object-based approach.

LLM	ARC Subset	Move 1 Pixel	Move 3 Pixels	Move Dynamic	Recolor by Size	Denoise	Fill		Move		Pile	
							H	V	H	V	H	V
GPT-3.5	6 (3×)	39 (3.9×)	14 (2×)	7 (1.16×)	21 (10.5×)	36 (2.76×)	17 (8.5×)	15 (∞)	1 (∞)	0 (-)	7 (3.5×)	10 (∞)
GPT-4	23 (1.76×)	50 (1.51×)	49 (4.08×)	37 (3.36×)	40 (2.85×)	50 (1.66×)	48 (1.04×)	49 (49×)	21 (1.75×)	20 (∞)	42 (1.31×)	37 (∞)

4.3 Analysis

We conducted the same solvability regression analysis from Section 2.4, observing the same correlations between task complexity attributes and solvability. Intriguingly, the models' performance was observed to decline when edge information was integrated into the representation. This unexpected result suggests that the influx of excessive information might overwhelm GPT, resulting in diminished performance. This discovery underscores the need for future research to find an optimal balance between supplying adequate contextual information and avoiding information overload.

Furthering our analysis, we performed a similar examination of the tasks correctly solved by GPT under the object-based approach. Out of the 23 tasks that produced the correct output, an impressive 20 tasks exhibited correct reasoning (See Appedix D). This significantly improved reasoning performance underscores the impact of effective object abstraction on GPT's reasoning abilities. Figure 5 additionally showcases GPT's reasoning when prompted using the object-based approach for a task where the direct-grid approach initially fell short in providing accurate reasoning.

5 Related work

Prompting methods for LLMs is a very active area of development (Qiao et al., 2022; Huang and Chang, 2022). CoT prompting is introduced in (Wei et al., 2022), providing LLMs with intermediate reasoning steps leading to improved performance on some complex reasoning tasks. Extending this, (Kojima et al., 2022) demonstrated LLMs' potential as zero-shot reasoners by incorporating a "Let's think step by step" phrase in the prompt. This approach notably enhanced accuracy on various reasoning tasks, thus hinting at untapped zero-shot capabilities within LLMs that can be leveraged through simple prompting techniques.

Augmented LLMs as surveyed in (Mialon et al., 2023) emphasizes their potential in overcoming limitations of a pure LLM approach. "Toolformer" self-learns to use external tools via APIs, significantly improving zero-shot performance across various tasks (Schick et al., 2023). Program-Aided Language models (PAL) (Gao et al., 2022) combines the strengths of LLMs with a Python interpreter to accurately solve some reasoning tasks.

Solvers for the ARC Since the introduction of the ARC in 2019, various methods have been proposed to address it. A powerful DSL coupled with an efficient program synthesis algorithm has the potential to solve the ARC, as initially proposed in (Chollet, 2019). Notable examples include the Kaggle challenge's (Kaggle, 2020) winning solution, which utilized a manually-created DSL and DAG-based search for program synthesis (top quarks, 2020). Other high-ranking Kaggle participants followed similar strategies (de Miquel Bleier, 2020; Golubev, 2020; Liukis, 2020; Penrose, 2020). (Fischer et al., 2020) employed a Grammatical Evolution algorithm within their chosen DSL, while (Alford et al., 2021) utilized the DreamCoder program synthesis system (Ellis et al., 2020) to derive abstractions from a basic DSL and compose solutions for new tasks through neural-guided synthesis. More recently, ARGA(Xu et al., 2023) was proposed as an object-centric framework that represents images using graphs and tree search for a correct program in a DSL based on the abstracted graph space. Alternative approaches for the ARC challenge have also been explored. The Neural Abstract Reasoner, a deep learning method, achieved success on a subset of ARC tasks (Kolev et al., 2020). (Assouel et al., 2022) devised a compositional imagination technique to generate unseen tasks for enhanced generalization. (Ferré, 2021) focused on an approach based on descriptive grids. However, these alternatives have not yet surpassed state-of-the-art results.

ARC-like datasets have been introduced to tackle the ARC's complexity. The Mini-ARC (Kim et al., 2022), a 5×5 compact version of the ARC, was generated manually to maintains the original's level of difficulty. The Sort-of-ARC (Assouel et al., 2022),shares ARC's input space but presents simpler problems with 20×20 images containing three distinct 3×3 objects. The ConceptARC dataset presents a set of manually crafted tasks, grouped and categorized by 16 distinct core concepts (Moskvichev et al., 2023). The PCFG benchmark (Mirchandani et al., 2023) is similar to our 1D-ARC and was generated using the probabilistic context-free grammar

LLM for the ARC In a recent study (Moskvichev et al., 2023), the capabilities of both automated methods and human cognition were explored with respect to the ARC. Their research employed state-of-the-art ARC solvers (top quarks, 2020; de Miquel Bleier, 2020) and GPT-4 to tackle tasks originating from the ConceptARC dataset, comparing these solutions with those produced by humans. The approach to prompt GPT-4 was comparable to our few-shot direct-grid encoding method outlined in Section 2.2. This study revealed that GPT-4 lags significantly behind both the leading ARC solver and human performance, a finding that aligns with our own. Other recent studies also echo this finding (Mirchandani et al., 2023; Camposampiero et al., 2023). However, it is critical to highlight that, based on our investigations, the proficiency of GPT-4 on the ConceptARC dataset could potentially be enhanced by adopting an object-based representation in the prompting process.

While our paper focused on the effects of representation, another recent study has shown improvements in performance by incorporating hypothesis search with LLMs (Wang et al., 2023).

6 Conclusion

We have explored the capabilities and limitations of the GPT LLM in solving ARC tasks seen as representatives of a certain kind of human-like intelligence. Our exploration started with a straightforward, grid-based textual encoding approach, which revealed that GPT struggles due to the non-sequential representation of complex objects in text. We then introduced the 1D-ARC, a simplified, single-dimensional version of the ARC. By reducing the task complexity and dimensionality, we aimed to make ARC tasks more approachable for LLMs. Our evaluations on the 1D-ARC indicated improvements in performance but also highlighted that simplification alone could not bridge all the gaps in GPT's reasoning processes.

In the third phase of our exploration, we adopted an object-based approach, integrating an external tool, the ARGA framework, to assist in object abstraction. This led to significant improvements in GPT's problem-solving abilities, reaffirming the importance of structured, object-based representations in complex reasoning tasks.

Future work Our research also uncovers potential avenues for future exploration. For instance, edge information was not fully utilized by the LLM, suggesting that GPT-4 may not be capable of dealing with graphs when represented in text form. As we delve deeper into the possibilities of structured representations, we might consider introducing a "language" of transformations for LLMs to use in solving ARC tasks. Recently, LLMs have been introduced that can process image inputs, albeit in early stages. We have conducted preliminary experiments on GPT-4V, a vision model by GPT (OpenAI, 2023b), using the subset of 50 tasks we introduced in this paper. We found that at the current stage, GPT-4V is only able to solve 2 out of 50 tasks. More details of the results are provided in Appendix G. These findings indicate a promising yet challenging pathway towards merging vision models with LLMs to enhance representation and improve performance in solving ARC tasks. However, the current stage of research underscores the necessity for substantial advancements to enable reliable reasoning by LLMs.

References

Sam Acquaviva, Yewen Pu, Marta Kryven, Theodoros Sechopoulos, Catherine Wong, Gabrielle Ecanow, Maxwell Nye, Michael Tessler, and Josh Tenenbaum. Communicating natural programs to humans and machines. *Advances in Neural Information Processing Systems*, 35:3731–3743, 2022.

Simon Alford, Anshula Gandhi, Akshay Rangamani, Andrzej Banburski, Tony Wang, Sylee Dandekar, John Chin, Tomaso Poggio, and Peter Chin. Neural-guided, bidirectional program search for abstraction and reasoning. In *International Conference on Complex Networks and Their Applications*, pages 657–668. Springer, 2021.

Rim Assouel, Pau Rodriguez, Perouz Taslakian, David Vazquez, and Yoshua Bengio. Object-centric compositional imagination for visual abstract reasoning. In *ICLR2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality*, 2022. URL `https://openreview.net/forum?id=rCzfIruU5x5`.

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

Giacomo Camposampiero, Loïc Houmard, Benjamin Estermann, Joël Mathys, and Roger Wattenhofer. Abstract visual reasoning enabled by language. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2642–2646, 2023.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

Alejandro de Miquel Bleier. Arc_kaggle. `https://github.com/alejandrodemiquel/ARC_Kaggle`, 2020. Accessed: 2022-08-01.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lucas Morales, Luke Hewitt, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning. *arXiv preprint arXiv:2006.08381*, 2020.

Sébastien Ferré. First steps of an approach to the arc challenge based on descriptive grid models and the minimum description length principle. *arXiv preprint arXiv:2112.00848*, 2021.

Raphael Fischer, Matthias Jakobs, Sascha Mücke, and Katharina Morik. Solving abstract reasoning tasks with grammatical evolution. In *LWDA*, pages 6–10, 2020.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*, 2022.

Vlad Golubev. Arc-kaggle-3rd-place. `https://www.kaggle.com/c/abstraction-and-reasoning-challenge/discussion/154305`, 2020. Accessed: 2022-08-01.

Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*, 2022.

Kaggle. Arc-kaggle-main. `https://www.kaggle.com/c/abstraction-and-reasoning-challenge`, 2020. Accessed: 2022-08-01.

Subin Kim, Prin Phunyaphibarn, Donghyun Ahn, and Sundong Kim. Playgrounds for abstraction and reasoning. In *NeurIPS 2022 Workshop on Neuro Causal and Symbolic AI (nCSI)*, 2022. URL `https://openreview.net/forum?id=F4RNpByoqP`.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=e2TBb5y0yFf`.

Victor Kolev, Bogdan Georgiev, and Svetlin Penkov. Neural abstract reasoner. *arXiv preprint arXiv:2011.09860*, 2020.

Agnis Liukis. Arc-kaggle-5th-place. `https://www.kaggle.com/c/abstraction-and-reasoning-challenge/discussion/154377`, 2020. Accessed: 2022-08-01.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*, 2023.

Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*, 2023.

Arseny Moskvichev, Victor Vikram Odouard, and Melanie Mitchell. The conceptarc benchmark: Evaluating understanding and generalization in the arc domain. *arXiv preprint arXiv:2305.07141*, 2023.

Victor Vikram Odouard and Melanie Mitchell. Evaluating understanding on conceptual abstraction benchmarks. *arXiv preprint arXiv:2206.14187*, 2022.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023a.

OpenAI. Gpt-4v(ision) system card. 2023b. URL `https://openai.com/research/gpt-4v-system-card`.

Andy Penrose. Arc-kaggle-8th-place. `https://www.kaggle.com/c/abstraction-and-reasoning-challenge/discussion/154384`, 2020. Accessed: 2022-08-01.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. Reasoning with language model prompting: A survey. *arXiv preprint arXiv:2212.09597*, 2022.

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

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Elizabeth S Spelke and Katherine D Kinzler. Core knowledge. *Developmental science*, 10(1):89–96, 2007.

top quarks. Arc-solution. `https://github.com/top-quarks/ARC-solution`, 2020. Accessed: 2022-08-01.

Ruocheng Wang, Eric Zelikman, Gabriel Poesia, Yewen Pu, Nick Haber, and Noah D Goodman. Hypothesis search: Inductive reasoning with language models. *arXiv preprint arXiv:2309.05660*, 2023.

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

Yudong Xu, Elias B. Khalil, and Scott Sanner. Graphs, constraints, and search for the abstraction and reasoning corpus. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI-23)*, Washington D.C., USA, 2023.