Towards Learning to Reason: Comparing LLMs with Neuro-Symbolic on Arithmetic Relations in Abstract Reasoning

Michael Hersche [a,*], Giacomo Camposampiero [a,b], Roger Wattenhofer [b], Abu Sebastian [a] and Abbas Rahimi [a]
[a] *IBM Research – Zurich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland.*
[b] *ETH Zürich, Gloriastrasse 35, 8092 Zürich, Switzerland.*

Abstract. This work compares large language models (LLMs) and neuro-symbolic approaches in solving Raven's progressive matrices (RPM), a visual abstract reasoning test that involves the understanding of mathematical rules such as progression or arithmetic addition. Providing the visual attributes directly as textual prompts, which assumes an oracle visual perception module, allows us to measure the model's abstract reasoning capability in isolation. Despite providing such compositionally structured representations from the oracle visual perception and advanced prompting techniques, both GPT-4 and Llama-3 70B cannot achieve perfect accuracy on the `center` constellation of the I-RAVEN dataset. Our analysis reveals that the root cause lies in the LLM's weakness in understanding and executing arithmetic rules. As a potential remedy, we analyze the Abductive Rule Learner with Context-awareness (ARLC), a neuro-symbolic approach that *learns to reason* with vector-symbolic architectures (VSAs). Here, concepts are represented with distributed vectors s.t. dot products between encoded vectors define a similarity kernel, and simple element-wise operations on the vectors perform addition/subtraction on the encoded values. We find that ARLC achieves almost perfect accuracy on the `center` constellation of I-RAVEN, demonstrating a high fidelity in arithmetic rules. To stress the length generalization capabilities of the models, we extend the RPM tests to larger matrices (3×10 instead of typical 3×3) and larger dynamic ranges of the attribute values (from 10 up to 1000). We find that the LLM's accuracy of solving arithmetic rules drops to sub-10%, especially as the dynamic range expands, while ARLC can maintain a high accuracy due to emulating symbolic computations on top of properly distributed representations.[1]

Keywords: Analogical reasoning, large language models, vector-symbolic architectures, reasoning benchmarks

1. Introduction

Abstract reasoning is often regarded as a core feature of human intelligence. This cognitive process involves abstracting rules from observed patterns in a source domain, and applying them in an unseen target domain. With the ultimate aim of achieving human-level intelligence, abstract reasoning tasks have sparked the interest of many in machine learning research. Thanks to the availability of large datasets [1–3], various learning-based methods, ranging from pure connectionist [4, 5] to neuro-symbolic [6–11] approaches, achieved promising results in this domain.

More recently, the zero- and few-shot capabilities of LLMs and their multi-modal variants have been tested on various abstract reasoning tasks such as verbal [12–15] or visual [12, 15–24] analogies. One natural approach

*Corresponding author. E-mail: michael.hersche@ibm.com.
[1] Our code is available at https://github.com/IBM/raven-large-language-models.

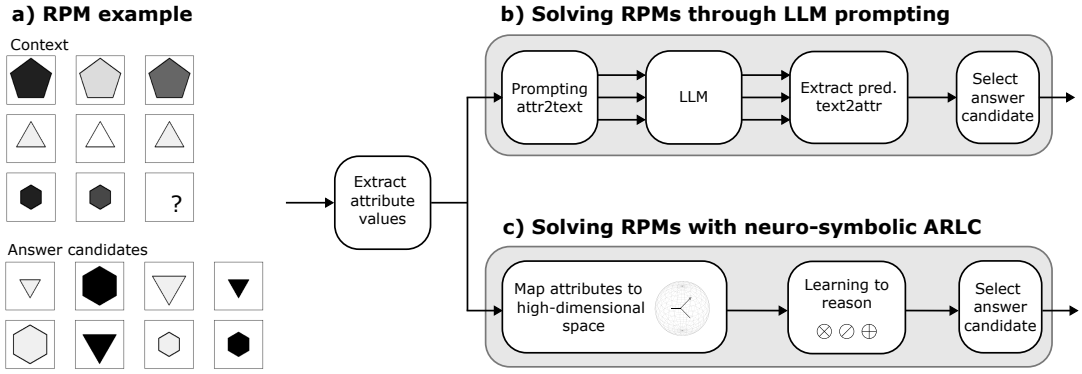


Fig. 1. This work compares the abstract reasoning capabilities of large language models (LLMs) and neuro-symbolic ARLC on Raven's progressive matrices (RPM) tests. **a)** An RPM example taken from the `center` constellation of I-RAVEN. The task is to find the empty panel at the bottom-right of the context matrix by selecting one of the answer candidates. **b)** Solving RPMs through LLM prompting. Visual attribute values are extracted from the I-RAVEN dataset and assembled to individual per-attribute text-only prompts. LLMs are prompted to predict the attribute of the empty panel. Finally, the attribute predictions are compared with the answer candidates, whereby the best-matching answer is selected as the final answer. **c)** Solving RPMs with neuro-symbolic ARLC that relies on distributed similarity-preserving representations and manipulates them via dimensionality-preserving operations; it learns rule-formulations as a differentiable assignment problem.

towards zero-shot visual abstract reasoning is to leverage multi-modal LLM's vision capabilities to solve the task end-to-end. However, these multi-modal models perform significantly worse than their text-only version [18], which might stem from a missing fine-grained compositional feature comprehension [16]. As an additional help, LLMs have been provided with text-only inputs by giving them access to an oracle perception, i.e., providing perfectly disentangled representations [12, 17]. While this generally improves their reasoning abilities, LLMs still fail to achieve perfect accuracy on many simple tasks. One example is represented by Raven's progressive matrices (RPMs) [25], a benchmark that tests visual abstract reasoning capabilities by measuring the fluid intelligence of humans. Here, the state-of-the-art (SOTA) LLM-based approach [17] achieves only 86.4% accuracy in the `center` constellation of I-RAVEN [3], which we observe to be a gate-keeper for this task (see Section 2.1).

In contrast, recent neuro-symbolic approaches showed not only almost perfect accuracy on the `center` constellation of I-RAVEN, but also demonstrated high fidelity in out-of-distribution (OOD) settings. For instance, the Abductive Rule Learner with Context-awareness (ARLC) represents attribute values with high-dimensional, distributed representations based on vector-symbolic architectures (VSAs) [26–29]. Learning the RPM rules boils down to a differentiable assignment problem of high-dimensional panel representations in a series of binding and unbinding operations, which can be solved with unconstrained optimization algorithms such as stochastic gradient descent (SGD). ARLC outperformed the SOTA LLM-based approach [17] both on in-distribution and OOD, thanks to relying on structured and similarity-preserving representations based on fractional power encoding (FPE) [27].

This paper extends on the initial work on ARLC [10], by comparing its abstract reasoning capability with two prominent LLMs (GPT-4 [30] and Llama-3 70B [31]) (see Fig. 1). Circumventing the perception by providing ground-truth attribute labels to the models allows us to measure their analogical and mathematical reasoning capabilities in isolation. Hence, we evaluate the reasoning capabilities of LLMs under conditions that play to their strengths, namely, language understanding, when such *compositionally structured* (i.e., disentangled) representations are provided. Our comprehensive prompting efforts lead to very high accuracy for Llama-3 70B (85.0%) and GPT-4 (93.2%), where the latter notably outperforms previous reports with GPT-3 [17] (86.4%) and GPT-4 o1-preview [24] (18.00%). This LLM's imperfect accuracy on the isolated task motivated us to further analyze their capability of detecting and executing different rules. In both GPT-4 and Llama-3 70B, we find a notable weakness in performing arithmetic rules that require row-wise additions or subtractions (e.g., see the last prompt in Fig. 2). To gain more insight about this behavior, we set up a new RPM dataset (I-RAVEN-X) that increases the grid size from 3×3 to 3×10, additionally allowing for a configurable dynamic range for the arithmetic computations. Also here, we observe a notable weakness in the arithmetic rule that gets even amplified by an increasing dynamic range.

On the other hand, ARLC demonstrates high accuracy on larger grid sizes and allows to increase the dynamic range without further retraining, thanks to the the capability of adjusting the underlying structured FPE representations.

2. Datasets

2.1. I-RAVEN

We test the models on the `center` constellation of I-RAVEN [3] (see Fig. 1). The test consists of a 3×3 context matrix, where the bottom-right panel is missing. The task is then to select the correct answer from eight candidate panels to complete the matrix. Independent of this setup, each panel contains a number of objects arranged according to a specific constellation. For instance, panels in the `center` constellation contain only one object, whereas panels in the `2x2` constellation contain at most four objects. The objects are characterized by different attributes (shape, size, and color). The relation between each attribute's value in different panels is governed by a well-defined set of rules:

- `constant`: This rule keeps the attribute value constant within the row.
- `progression`: The attribute value monotonically increases or decreases in a row by a value of 1 or 2.
- `arithmetic`: The attribute values of the first two panels are either added (`arithmetic plus`) or subtracted (`arithmetic minus`), yielding the attribute value of the third panel in the row.
- `distribute three`: This rule involves the fact that three different values of an attribute appear in the three panels of every row (with distinct permutations of the values in different rows). The same holds with respect to the columns.

The task is to infer the rule governing each attribute in the context matrix and use it to determine the content of the missing (bottom-right) panel, selecting it within the eight candidate answers. Compared to other RPM benchmarks that have been used to evaluate LLMs [12], I-RAVEN tests a more complex range of logical and arithmetic skills. While I-RAVEN provides tests in various constellations with more objects that may intuitively appear more arduous to solve, LLMs are more challenged with the seemingly simple constellations. For instance, GPT-3 achieved a higher accuracy on the `2x2` and `3x3` constellations (78.0% and 86.4%) than on `center` (77.2%) [17]. Moreover, high accuracy can be maintained on the `2x2` and `3x3` constellations while only looking at the last row of the context matrix [17], effectively showing that no analogical reasoning is required to solve the test in these constellations. Hence, we opted to focus our evaluation on the `center` constellation only, using 500 samples from I-RAVEN's test set.

Inspired by recent works [12, 17], we simplify RPM from a visual abstract reasoning test to a purely abstract reasoning test. Assuming a perfect perception, we extract the attribute values from I-RAVEN and use them to create the prompts for the model. This approach simplifies the RPM task as it not only provides the correct attributes but also filters irrelevant attributes. In a follow-up work [32], we tested the robustness of LLMs against uncertainties in the perception. As expected, adding confounding attributes as well as a smoothened non-one-hot distribution degraded the LLMs' performance considerably.

2.2. New I-RAVEN-X

To further evaluate the mathematical reasoning capabilities at scale, we introduce an extension of the I-RAVEN's `center` constellation, called I-RAVEN-X. Our new benchmark maintains I-RAVEN's four rules and three attributes but allows for a parameterizable number of columns (g) and a dynamic range of attribute values (m). When generating a new RPM example, we uniformly sample from one of the available rules (`constant`, `progression`, `arithmetic`, and `distribute three`). Note that the attribute `shape` does not incur the `arithmetic` rule.

In the following, we describe the generation process of the RPM context matrix of size $3 \times g$ for the individual rules. The overall goal is that the values stay in the range $[0, m-1]$.

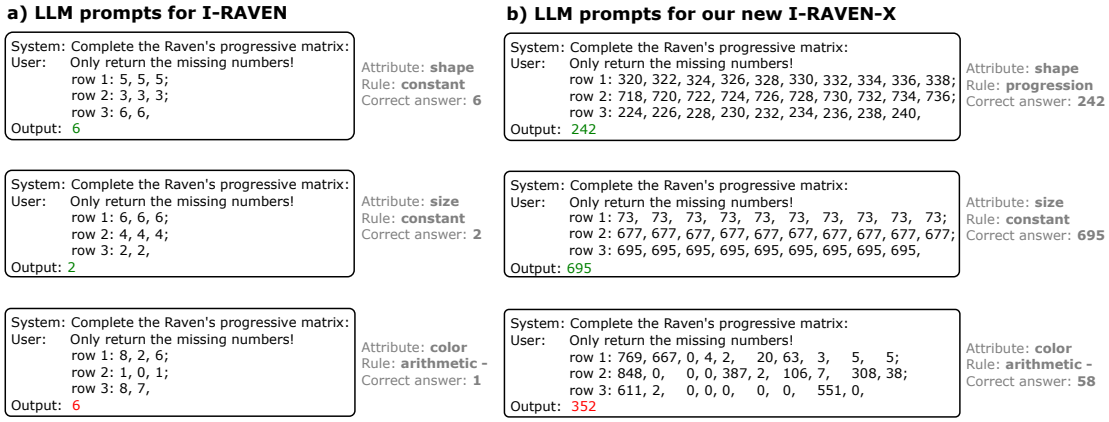


Fig. 2. **a)** Individual per-attribute text-only prompts to solve RPM tasks from I-RAVEN. **b)** Example prompts with of our novel configurable I-RAVEN-X dataset of size 3×10 with a value range of $m = 1000$. In both the I-RAVEN and I-RAVEN-X examples, the LLM (GPT-4) errs in the arithmetic rules.

- `constant`: For each row, we uniformly sample an integer from the set $\{0, 1, ..., m-1\}$, and duplicate along the row.
- `progression`: First, we uniformly sample the progressive increment/decrement (δ) from the set $\{-2, -1, +1, +2\}$. In case of a positive increment, we first define the values of the right-most columns, by uniformly sampling from the set $\{(g-1)\cdot\delta, ..., m-1\}$ for each row. Then, the rest of the matrix is completed by applying the progression rule. The sampling for a negative δ is done specularly from the first column.
- `arithmetic`: The attribute values of the first $g-1$ panels are either added (`arithmetic plus`) or subtracted (`arithmetic minus`), yielding the attribute value of the last panel in the row. In `arithmetic plus`, we sequentially sample the values from the first $g-1$ panels in the row. For each panel, we set the sampling range to $\{0, ..., m-s\}$, where s is the sum of the already sampled panels in the row. Afterward, the first $g-1$ panels are shuffled. Finally, the values of the last panels are the sum of the first $g-1$ ones, applied row-wise. For `arithmetic minus`, we apply the same sampling strategy but leave the first column empty. The value of the first column is then defined as the sum of the other columns.
- `distribute-n`: We uniformly sample distinct values for the first row from $\{0, ..., m-1\}$. The content of the remaining rows is defined by applying a circular shift per row (either right or left).

Finally, we generate the candidate answers using I-RAVEN's attribute bisection tree [3]. The original RAVEN dataset had a flaw in the generation of the answer set. Each distractor in the answer set (i.e., a wrong answer candidate) was generated by randomly altering one attribute of the correct answer. As a result, one could predict the correct answer by taking the mode of the answer candidates without looking at the context matrix, therefore bypassing the actual reasoning task. As a remedy, the attribute bisection tree generates unbiased answers that are well balanced. Fig. 2b shows example prompts generated from samples of our new dataset.

3. LLM-based RPM solving

3.1. Models

We focused our evaluations on text-only LLMs. There exist attempts [16, 18, 20–22] that leverage vision support of some multi-modal LLMs (e.g., GPT-4V) directly feeding the models with visual RPM data; however, they achieve consistently lower reasoning performance than with text-only prompting. The SOTA LLM-based abstract reasoning approach [17] relied on reading out GPT-3's (`text-davinci-002`) token probabilities. However, this model is no longer accessible to users and its successive iterations do not allow the retrieval of prediction logits. Hence, we considered discrete classification approaches that are based on output strings rather than distribution over tokens.

In particular, we investigated two SOTA LLMs: the proprietary GPT-4 [30][2] (`gpt-4-0613`) and the open-source Llama-3 70B [31][3]. More recent iterations of these models were not considered in our analysis for different reasons. Meta's attribution requirement in their updated terms regarding naming conventions prevented us from testing Llama-3.1 During initial tests, GPT-4o yielded worse results than GPT-4, hence we focused on GPT-4. Moreover, the evaluation of large reasoning models, such as DeepSeek's R1 [33] or OpenAI's o-series [34], is covered in our follow-up work [32].

3.2. Prompting and classification

Entangled and disentangled prompts. Following [17], we use numerical descriptions of the attribute values that has lead to better performance than textual descriptions [24]. Moreover, we evaluate two different prompting strategies, *entangled* and *disentangled* prompting. The entangled prompting provides all the attributes' values in a single prompt (see Appendix A.1). The disentangled prompting, on the other hand, is a compositionally structured approach that queries the LLM for individual attribute prediction. Disentangled prompting simplifies the task, but increases the number of queries by $3\times$.

Discriminative and predictive classification. Similarly to [14], we consider two approaches to solve RPM tests with LLMs. In the *discriminative* approach, we provide the attribute descriptions of both the context matrix and the answer candidates. The LLM is then asked to return the panel number of the predicted answer. Appendix A.2 provides an example prompt of the discriminative approach. In the *predictive* approach, we prompt the LLM only with the context matrix without the candidate answers. The LLM has to predict the value of the empty panel (see Fig. 2). For selecting the final answer, we compare the predicted values with the answer panels and pick the one with the highest number of overlapping values. While the predictive approach may appear more difficult, it implicitly biases the LLM to approach the task as humans usually do, i.e., first applying a generative process to abduce rules and execute them to synthesize a possible solution, and then discriminatively selecting the most similar answer from choices [35]. Moreover, the final answer selection is done without the intervention of the LLM, rendering phenomena like hallucinations less likely. Thus, the predictive classification can be seen as a more guided approach that helps LLM to solve the task.

Self-consistency. As an optional extension, we employ self-consistency [36, 37] by querying the model multiple times ($n = 7$ times), sampling the next token from the distribution with a non-zero soft-max temperature. We find the optimal soft-max temperature for GPT-4 ($T = 0.5$) and Llama-3 70 B ($T = 0.4$) via a grid search on a subset of 50 I-RAVEN problems. We did not explore the effect of other parameters, such as top-k or top-p, and set them to the default values. The final prediction is determined by a majority vote over the sampled outputs. The selection of an odd number of samples (i.e., $n = 7$) helps to prevent potential ties.

In-context learning For a better understanding of the RPM task, we optionally prefix 16 in-context examples to the prompt [38]. In the predictive classification approach (where no answer candidates are provided), we simply provide complete example RPM matrices. The in-context samples are randomly selected from I-RAVEN's training set. Examples that had the same context matrix as the actual task are discarded and re-sampled to prevent shortcut solutions.

4. ARLC: learning abductive reasoning using VSA distributed representations

This section presents the Abductive Rule Learner with Context-awareness (ARLC), which performs neuro-symbolic reasoning with distributed VSA representations (see Fig. 3). ARLC projects each panel's attribute value (or distributions of values) into a high-dimensional VSA space. The resulting VSA vectors preserve the semantic similarity between attribute values: the dot products between corresponding VSA encoded vectors define a similarity kernel [27, 39]. Moreover, simple component-wise operations on these vectors, binding and unbinding, perform

[2]GPT-4 was accessed between 07/03/2024–10/30/2024.
[3]The model weights were downloaded and evaluated locally. Unless stated otherwise, we use the base model without instruction tuning.

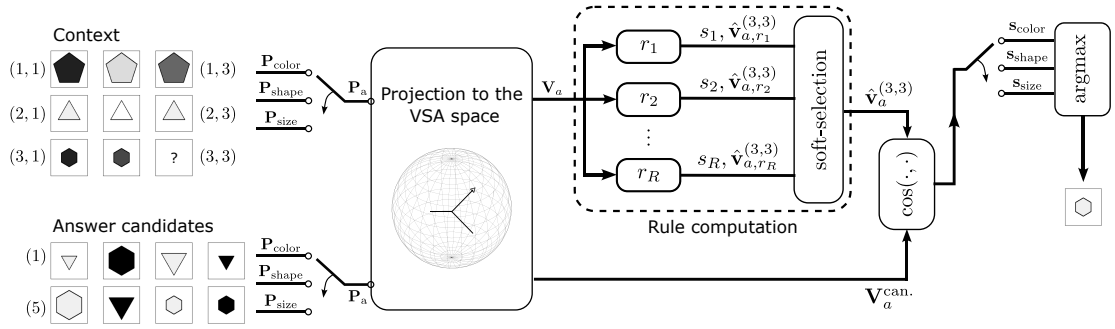


Fig. 3. ARLC architecture. ARLC maps attribute values, or distributions of values, to distributed VSA representations, where the semantic similarity between values is preserved via a notion of kernel. Learnable rules $(r_1, ..., r_R)$ predict the VSA representation of the empty panel $(\hat{\mathbf{v}}_{a,r}^{(3,3)})$ together with a confidence value (s_r). The closest answer to the predicted soft-selected prediction $(\hat{\mathbf{v}}_a^{(3,3)})$ is chosen as the final answer.

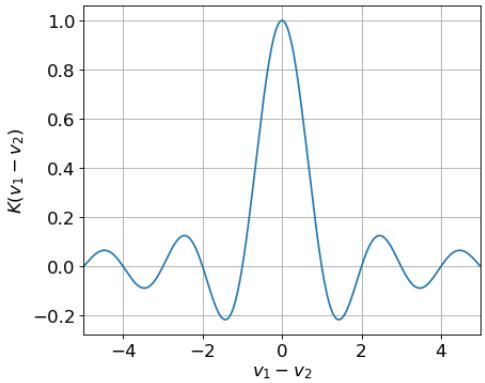


Fig. 4. Similarity kernel in VSA. Mapping two values (v_1 and v_2) to a VSA space (i.e., GSBC in ARLC) that uses fractional power encoding (FPE) and computing their similarity in the VSA space yields the shown similarity kernel $K(v_1 - v_2)$.

addition and subtraction respectively on the encoded values. For rule learning, ARLC introduces a generic rule template with several terms forming a series of binding and unbinding operations between vectors. The problem of learning the rules from data is reduced to a differentiable assignment problem between the terms of the general rule template and the VSA vectors encoding the contents of the panels, which can be learned with standard SGD. ARLC was initially presented in [10]; this work mainly compares it to the reasoning capabilities of LLMs on I-RAVEN, and demonstrates its extension to larger grid sizes and dynamic ranges on our novel I-RAVEN-X.

4.1. From visual attributes to distributed VSA representations

ARLC's key concept is to represent attribute values with high-dimensional, distributed VSA vectors that preserve the semantic similarity between the attribute values thanks to an introduced kernel notion. We start by defining a VSA that equips the space with dimensionality-preserving vector operations. Bundling ($\oplus$) is a similarity-preserving operation that creates a superposition of the operands, that is, the resulting vector will have a high similarity with the two operands. Binding ($\otimes$) associates two elements, effectively encoding a relationship between two vectors. For example, binding the attribute "color" with the value "red" produces a new vector that represents this pair. Importantly, this operation destroys similarity: the result is dissimilar to both operands, which helps prevent interference between different bindings. Unbinding ($\oslash$) is the inverse operation. Given a bound pair and one of its components (e.g., the attribute), unbinding retrieves the other (e.g., the value). This allows for structured information retrieval. The main difference between members of the VSA family is the specific realization of the bundling, binding, and vector space (see [40]).

Table 1
Supported VSA operations and their equivalent in $\mathbb{R}$.

Operation	Binary GSBCs with FPE	Equivalent in $\mathbb{R}$
Binding ($\otimes$)	Block-wise circular convolution	Addition $+$
Unbinding ($\oslash$)	Block-wise circular correlation	Subtraction $-$
Bundling ($\oplus$)	Sum & normalization	—
Similarity ($\odot$)	Cosine similarity ($\cos(\cdot,\cdot)$)	—

Specifically, ARLC uses binary generalized sparse block codes (GSBCs) [41] as a particular VSA instance. In binary GSBCs, the D-dimensional vectors are divided into B blocks of equal length, $L = D/B$, where only one (randomly selected) element per block is set to 1 ($D = 1024$ and $B = 4$). The algebraic operations of binary GSBCs are defined in Table 1. The choice of binary GSBCs is motivated by better retrieval accuracy when encoding probability mass functions (PMFs), compared to other alternatives such as Fourier holographic reduced representations (FHRR) [8]. See Appendix B for a detailed background on VSA.

Next, we define a mapping $z : \mathbb{Z}^+ \rightarrow \mathbb{R}^D$ that enables the projection of input RPM attributes into a corresponding high-dimensional, semantically-rich feature space. Note that this work focuses on mapping integer values as the attribute values in I-RAVEN are integer-valued too. However, generalizing this approach to real-valued domain mappings is possible using frequency holographic reduced representations (FHRR) [26]. Leveraging fractional power encoding (FPE) [27], a value $v \in \mathbb{Z}^+$ is encoded as

$$\mathbf{z}(v) = \mathbf{z}^v = \bigotimes_{n=1}^{v} \mathbf{z},$$

where $\mathbf{z} \in \mathbb{R}^D$ is a randomly drawn binary GSBC vector. This mapping yields a similarity kernel between neighboring vector representations [39], as shown in Fig. 4.

Let us assume two variables with values v_1 and v_2, which are represented with two VSA vectors ($\mathbf{z}(v_1) = \mathbf{z}^{v_1}$ and $\mathbf{z}(v_1) = \mathbf{z}^{v_2}$). Binding the two vectors yields $\mathbf{z}(v_1) \otimes \mathbf{z}(v_2) = \mathbf{z}^{v_1} \otimes \mathbf{z}^{v_2} = \mathbf{z}^{v_1+v_2}$. Hence, binding in the VSA space is equivalent to the addition in $\mathbb{R}$. In other words, the FPE initialization allows to establish a semantic equivalence between high-dimensional vectors and real numbers. This property is consistently exploited in ARLC's framework, as it allows to solve the analogies in the RPM puzzles as simple algebraic operations in the domain of real numbers. For example, by computing the similarity between the bound representation and a third projected variable ($\text{sim}(\mathbf{z}^{v_1+v_2}, \mathbf{z}^{v_3})$), we can evaluate whether $v_1 + v_2 \stackrel{?}{=} v_3$ representing the `arithmetic plus` rule in RPM.

One advantage of performing reasoning with distributed VSA representations is its capability to represent perceptual uncertainty in the variable values. Connecting to the previous example, let us assume that the first variable takes value v_1 with probability p and value v_1' with probability $p' = 1 - p$. The distribution can be encoded as the weighted superposition of the two corresponding codewords: $p \cdot \mathbf{z}^{v_1} + p' \cdot \mathbf{z}^{v_1'}$. The similarity computation between the bound representation and a third variable would then yield

$$\text{sim}((p \cdot \mathbf{z}^{v_1} + p' \cdot \mathbf{z}^{v_1'}) \otimes \mathbf{z}^{v_2}, \mathbf{z}^{v_3}) = \text{sim}(p \cdot \mathbf{z}^{v_1} \otimes \mathbf{z}^{v_2} + p' \cdot \mathbf{z}^{v_1'} \otimes \mathbf{z}^{v_2}, \mathbf{z}^{v_3}) \quad (1)$$

$$\approx p \cdot \text{sim}(\mathbf{z}^{v_1} \otimes \mathbf{z}^{v_2}, \mathbf{z}^{v_3}) + p' \cdot \text{sim}(\mathbf{z}^{v_1'} \otimes \mathbf{z}^{v_2}, \mathbf{z}^{v_3}), \quad (2)$$

where the first equality uses the linearity of the binding operation, and the second approximation requires linearity of the similarity metric[4]. Overall, this formulation allows the validation of multiple solutions (in this case two) using only a single binding and similarity computation.

[4]The GSBC uses the non-linear cosine similarity that impacts the approximation in Equation (2). Note, however, that ARLC's rule selection considers relative similarities between different rule probabilities, where the approximation is sufficient.

In the RPM application, each panel's label is translated to a PMF $\mathbf{p}_a^{(i,j)}$, where a is the attribute, i is the row index and j is the column index of the panel. The panel's PMF is then projected into the VSA space as

$$\mathbf{v}_a^{(i,j)} = \sum_{k=1}^{m} \mathbf{p}_a^{(i,j)}[k] \cdot \mathbf{z}^k,$$

where m is the number of possible values that the attribute a can assume. Overall, this yields eight VSA vectors for each attribute a (one for each panel of the input RPM matrix), represented by

$$\mathbf{V}_a := \left(\mathbf{v}_a^{(1,1)}, \mathbf{v}_a^{(1,2)}, \ldots, \mathbf{v}_a^{(3,2)}\right). \tag{3}$$

Note that the basis vectors are pre-computed and stored in a dictionary $\mathbb{C} = \{\mathbf{z}^k\}_{i=1}^{r}$ containing m elements.

4.2. Learning RPM rules as an assignment problem

The previous example demonstrates that executing the `arithmetic` rule requires addition computations, which can be efficiently performed in the VSA space using the binding operation. Indeed, we find that other RPM rules (`constant`, `progression`, `distribute three`) can be described with one or multiple additions and subtractions as well, which can be represented in the VSA space using binding and unbinding operations, respectively (see Appendix D). Hence, the rules used in RPM can be generally framed as a series of binding and unbinding operations:

$$r = (\mathbf{c}_1 \otimes \mathbf{c}_2 \otimes \mathbf{c}_3 \otimes \mathbf{c}_4 \otimes \mathbf{c}_5 \otimes \mathbf{c}_6) \oslash (\mathbf{c}_7 \otimes \mathbf{c}_8 \otimes \mathbf{c}_9 \otimes \mathbf{c}_{10} \otimes \mathbf{c}_{11} \otimes \mathbf{c}_{12}), \tag{4}$$

Here, each c_i can either assume the value of a context panel $v_a^{(i,j)}$ or the identity vector e. The assignments between each placeholder c and its value are learned during training (or programmed) and depend on the specific rule. For instance, during the inference of the arithmetic plus rule on the 3^{rd} row of the context matrix, the assignments would correspond to:

$$c_1 = v_a^{(3,1)}, \quad c_2 = v_a^{(3,2)}, \quad c_i = e \quad \forall i \in \{3, 4, ..., 12\}.$$

where $v_a^{(3,1)}$ and $v_a^{(3,2)}$ are the vector representations of the first and second panel of the row, respectively. In this setting, learning RPM rules can be interpreted as an assignment problem between VSA vectors and the terms in Equation (4).

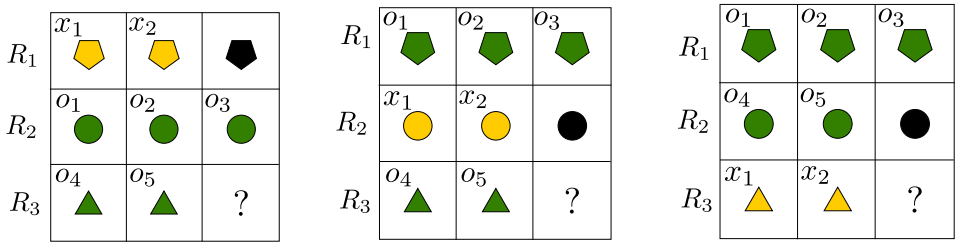
Motivated by works in cognitive sciences and psychology that argue for the importance of context in the solution of analogies for humans [42, 43], ARLC uses a general formulation of the soft-assignment problem which relies on the notion of *context*:

$$\mathbf{c}_k = \sum_{i=1}^{I} w_k^i \cdot \mathbf{x}_i + \sum_{j=1}^{J} u_k^j \cdot \mathbf{o}_j + v_k \cdot \mathbf{e}, \tag{5}$$

where $\mathbf{w}, \mathbf{u}, \mathbf{v}$ are the learned parameters and they are subject to the following constraints:

$$\sum_{i=0}^{I} w_k^i + \sum_{j=0}^{J} w_k^j + v_k = 1, \quad 0 \leqslant w_k^i \leqslant 1 \, \forall i, \quad 0 \leqslant u_k^j \leqslant 1 \, \forall j, \quad 0 \leqslant v_k \leqslant 1, \, \forall k.$$

Here, $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_I\}$ is the set of attributes that define the current sample, that is, the description of the problem for which we infer a solution. $\mathbf{O} = \{\mathbf{o}_1, \ldots, \mathbf{o}_J\}$ is the set of attributes that define the context for



(a) $X = \{R_1\}, O = \{R_2, R_3\}$ (b) $X = \{R_2\}, O = \{R_1, R_3\}$ (c) $X = \{R_3\}, O = \{R_1, R_2\}$

Fig. 5. Visualization of current samples ($\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2\}$, in yellow) and context ($\mathbf{O} = \{\mathbf{o}_1, \ldots, \mathbf{o}_5\}$, in green) panels when predicting the third panel for different rows, namely the first row (left), second row (center) and third row (right). Black objects represent panels that are not used for the computation, while the question mark represents the unknown test panel, which is unavailable during inference.

that sample, that could be interpreted as a working memory from which additional information to infer the answer can be retrieved. For predicting the empty panel in the last row, the context ($\mathbf{O}$) corresponds to the first two rows and the current samples ($\mathbf{X}$) to the last row (see Fig. 5c). Formally, we set $\mathbf{X} = \{\mathbf{v}_a^{(3,1)}, \mathbf{v}_a^{(3,2)}\}$ and $\mathbf{O} = \{\mathbf{v}_a^{(1,1)}, \mathbf{v}_a^{(1,2)}, \mathbf{v}_a^{(1,3)}, \mathbf{v}_a^{(2,1)}, \mathbf{v}_a^{(2,2)}\}$ for predicting the rightmost panel in the last row. We augment this standard prediction with two more permutations, which aim to predict the rightmost panel of the first and second row (see Fig. 5a and Fig. 5b). The knowledge of the rightmost panels in the first two rows allows us to compute a rule confidence by comparing the rule's prediction with the actual panel representation via the cosine similarity.

4.3. Executing and selecting the learned rules

ARLC learns a set of R different rules with rule-specific weights ($\mathbf{w}_r, \mathbf{u}_r, \mathbf{v}_r$). Inference with the learned rule set is a two-step process: an execution step (where all the rules are applied in parallel to the input) and a selection step (where a prediction for the missing panel is generated). The application of each rule r to an RPM example generates a tuple of three VSA vectors $(\hat{\mathbf{v}}_{a,r}^{(i,3)})_{i=1}^{3}$, which corresponds to the result of the rule execution on the three rows of the RPM matrix, together with a rule confidence value s_r. The confidence value is computed as the sum of the cosine similarities between the predicted VSA vectors and their respective ground-truth vector,

$$s_r = \sum_{i=1}^{3} \cos\left(\mathbf{v}_a^{(i,3)}, \hat{\mathbf{v}}_{a,r}^{(i,3)}\right). \tag{6}$$

Note that the ground-truth value for the last row ($\mathbf{v}_a^{(3,3)}$) is unknown during inference, since the RPM task is to predict this panel. Hence, we omit the last term of the sum (i = 3) in the inference. The answer is finally produced by taking a linear combination of the VSA vectors generated by executing all the rules, weighted by their respective confidence scores (normalized to a valid probability distribution using a softmax function). More formally, if we define $\mathbf{s} = [s_1, \ldots, s_R]$ to be the concatenation of all rules' confidence score and $\hat{\mathbf{V}}_a^{(3,3)} = [\hat{\mathbf{v}}_{a,1}^{(3,3)}, \ldots, \hat{\mathbf{v}}_{a,R}^{(3,3)}]$ to be the concatenation of all rules' predictions for the missing panel, the final VSA vector predicted by the model for the attribute a becomes

$$\hat{\mathbf{v}}_a^{(3,3)} = \text{softmax}\left(\mathbf{s}\right) \cdot \hat{\mathbf{V}}_a^{(3,3)}. \tag{7}$$

The use of the weighted combination can be understood as a *soft-selection* mechanism between rules and was found to be more effective compared to the *hard-selection* mechanism provided by sampling [9].

4.4. Training Loss and other Implementation Aspects

We follow the training recipe provided by Learn-VRF [9]. The model is trained using stochastic gradient descent (SGD) with a learning rate $\text{lr} = 0.01$ for 25 epochs. The training loss is defined as the inverse cosine similarity

Table 2

Task accuracy (%) on the `center` constellation of I-RAVEN. Among the baselines, we replicate Learn-VRF [9]; the other results are taken from [8]. The standard deviations are reported over 10 random seeds. Llama-3 and GPT-4 are queried with the corresponding best prompting technique (see Table 3). ARLC's weights are either manually programmed ($\text{ARLC}_{\text{progr}}$), learned from scratch ($\text{ARLC}_{\text{learn}}$), or learned after manual programming ($\text{ARLC}_{\text{p}\mapsto\text{l}}$). The number of parameters for GPT-4 is not publicly available. The reasoning backend of PrAE, NVSA, and our $\text{ARLC}_{\text{progr}}$ do not have trainable parameters.

Method	Parameters	Accuracy
MLP [9]	300 k	97.6
SCL [5]	961 k	$99.9^{\pm 0.0}$
PrAE [6]	n.a.	$83.8^{\pm 3.4}$
NVSA [8]	n.a.	$99.8^{\pm 0.2}$
Learn-VRF [9]	20 k	$97.7^{\pm 4.1}$
GPT-3 [17]	175 b	86.4
Llama-3	70 b	85.0
GPT-4	unk.	93.2
$\text{ARLC}_{\text{progr}}$	n.a.	$99.6^{\pm 0.0}$
$\text{ARLC}_{\text{p}\mapsto\text{l}}$	480	$99.6^{\pm 0.0}$
$\text{ARLC}_{\text{learn}}$	480	$98.4^{\pm 1.5}$

between the three predicted panels and their corresponding ground truth

$$\mathcal{L} = 1 - \sum_{i=1}^{3} \cos\left(\mathbf{v}_a^{(i,3)}, \hat{\mathbf{v}}_a^{(i,3)}\right). \tag{8}$$

As in Learn-VRF, we set the number of rules to $R = 5$. A single set of rules is instantiated and shared between all RPM attributes.

4.5. Applying ARLC on I-RAVEN-X

While ARLC was initially designed for I-RAVEN, it can be seamlessly extended to our I-RAVEN-X with minor modifications. First, the number of binding/unbinding terms in Equation (4) is increased, e.g., from 12 to 22 to support the larger grid size of $g = 10$. Moreover, we increase the number of entries in the dictionary ($\mathbb{C}$) to support the larger dynamic range (m). Notably, only varying the dynamic range at constant grid size does not require retraining: we can simply replace the dictionary in order to support OOD generalization. Indeed, we could demonstrate that ARLC trained on a dynamic range of $m = 45$ can favorably generalize to a dynamic range of $m = 1000$.

5. Results

5.1. Main results on I-RAVEN

Table 2 compares our LLM results with ARLC on the `center` constellation of I-RAVEN, considering also a range of neuro-symbolic and connectionist baselines. For the LLMs, we show the results with the corresponding best prompting techniques (see the ablation in Section 5.2). Moreover, we present results for three different versions of ARLC: $\text{ARLC}_{\text{progr}}$, where the model's weights are manually programmed with RPM rules ($R = 4$, since `constant` can be considered as a special case of `progression`), $\text{ARLC}_{\text{p}\mapsto\text{l}}$, where the model is initialized with the programmed rules and then trained with gradient descent, and $\text{ARLC}_{\text{learn}}$, where the rules are learned from scratch from data.

Among the LLM approaches, our GPT-4-based approach achieved the highest accuracy (93.2%) notably outperforming previous SOTA LLM-based abstract reasoning approaches on this benchmark (86.4%) [17]. Yet, all LLM

Table 3
Ablation study considering various LLM prompting techniques. We report the task accuracy (%) on the `center` constellation of I-RAVEN.

Predictive/ discriminative	Disentangled queries per attribute (3×queries)	Self-consistency (n=7)	In-context learning (s=16)	GPT-4	Llama-3 70B
Discriminative				46.8	22.8
Discriminative	✓			63.0	22.4
Predictive				74.8	79.0
Predictive	✓			91.4	83.2
Predictive	✓	✓		**93.2**	84.8
Predictive	✓		✓	85.4	84.8
Predictive	✓	✓	✓	86.4	**85.0**

Table 4
Accuracy (%) of predicting the correct attribute value. Self-consistency (n=7) is used. Results are averaged across all attributes.

Model	Disentangled queries per attribute (3×queries)	Constant	Progression	Distribute three	Arithmetic
GPT-4	No	100	98.0	91.6	27.1
	Yes	100	100	99.5	73.6
Llama-3 70B	No	100	97.2	99.3	31.0
	Yes	100	100	96.6	45.0

approaches fall behind the tailored connectionist and neuro-symbolic solutions. Notably, with only 480 learnable parameters, ARLC achieves a high accuracy of 98.4%. Moreover, we show that post-programming training allows for maintaining the knowledge of the model, rather than completely erasing it as shown in other settings [44].

5.2. *Ablation of LLM prompting techniques*

Table 3 shows the task accuracy on I-RAVEN using GPT-4 and Llama-3 70B in various prompting configurations. Overall, both models benefit from the additional guidance provided by our prompting techniques. Concretely, using a predictive approach and querying for individual disentangled attributes yielded already high accuracies (91.4% and 83.2% for GPT-4 and Llama-3 70B, respectively). Introducing self-consistency further improves the accuracy for both models. Llama-3 70B's performance can be further pushed (to 85.0%) by using self-consistency and in-context learning. On the contrary, GPT-4 cannot make use of the additional in-context samples, yielding a lower accuracy instead. Indeed, recent work on LLM reasoning models [33] made a similar observation, where "few-shot prompting consistently degrades its performance."

To test the potential impact of instruction-tuning, we conducted experiments with Llama 3 70B Instruct. We found that the instruction-tuned model generally performs worse, achieving 64.6% and 79.2% with and without in-context learning, respectively. We leave the exploration of finding an optimized set and sequence of in-context examples, which has been shown to improve the performance of instruction-tuned models [45], for future work.

5.3. *LLMs show weakness in arithmetic rule*

Even though both LLMs achieve a reasonable overall task accuracy, they fail in some instances. We shed more light on the reasoning capability of the two models by analyzing the accuracy of predicting the correct value for a given rule. As shown in Table 4, both models perform well on `constant`, `progression`, and `distribute three` rules, whereas the accuracy notably drops for the `arithmetic` rule. One explanation for the accuracy drop could be the LLM's tendency for (short-sighted) relational reasoning, instead of performing relational mapping that requires the understanding of the first two rows before applying a rule on the last row [13]. We analyze this

Table 5

Task accuracy (%) on I-RAVEN and our novel I-RAVEN-X. The LLMs use self-consistency (n=7). For $\text{ARLC}_{\text{learn}}$ we report max/mean evaluation accuracies over 5 different training seeds.

	I-RAVEN	I-RAVEN-X		
	3×3	3×10		
Dynamic range (m)	5–10	50	100	1000
Llama-3 70B	85.0	76.8	73.0	74.2
GPT-4	93.2	82.2	79.6	76.6
$\text{ARLC}_{\text{progr}}$	99.6	100.0	100.0	99.7
$\text{ARLC}_{\text{learn}}$	99.1/98.6	94.6/86.3	95.1/88.0	91.6/82.8

Table 6

Arithmetic accuracy (%) on I-RAVEN and our novel I-RAVEN-X. The LLMs use self-consistency (n=7). For $\text{ARLC}_{\text{learn}}$ we report max/mean evaluation accuracies over 5 different training seeds.

	I-RAVEN	I-RAVEN-X		
	3×3	3×10		
Dynamic range (m)	5–10	50	100	1000
Llama-3 70B	45.0	1.5	2.6	0.4
GPT-4	73.6	30.4	25.1	8.4
$\text{ARLC}_{\text{progr}}$	100.0	99.8	100.0	99.5
$\text{ARLC}_{\text{learn}}$	99.5/99.2	99.1/95.5	98.9/96.3	97.9/95.3

hypothesis in Appendix C, where we attempt to explain the LLM's wrong predictions by rules that may have been inferred from the last row. For GPT-4, 32 out of 68 errors can be explained by rules that might have been inferred from a partial context matrix, e.g., a `constant` or `progression` rule based on the last row.

5.4. *Results on our novel I-RAVEN-X*

Finally, we conduct experiments on our novel I-RAVEN-X test, which allows us to configure the matrix size and the dynamic range of the attribute values. We fix the grid size to 3×10 and vary the dynamic range between 50, 100, and 1000. As shown in Table 5, the LLM's drops not only due to the larger grid size but also generally degrades with an increasing dynamic range. At the same time, our ARLC maintains a high accuracy across the board, while only being trained at dynamic range of 50 and reconfigured for the higher ranges. Investigating the performance on the `arithmetic` rule in Table 6 explains the overall accuracy degradation: the arithmetic accuracy drops below 10% for both LLMs at the highest dynamic range (1000).

6. Conclusion

This work revealed LLM's limitations in recognizing and executing arithmetic rules in abstract reasoning tasks, despite being provided disentangled prompts with ground-truth visual attributes and using advanced prompting techniques. We further showed the serious limitation on a larger (3×10) RPM test. As a viable alternative, we presented a neuro-symbolic approach (ARLC) that achieves a high accuracy both on I-RAVEN and our I-RAVEN-X, thanks to learning to reason with distributed VSA representations and operators. Beyond accuracy, ARLC not only inherits advantages from symbolic methods (e.g., interpretability and programmability) but also advances efficiency and trainability. Yet, it is still tailored and trained to solve the given RPM task. In contrast, LLMs are more general but lack interpretability and require more computing resources. Combining the strengths of both methods, we see great potential in integrating ARLC into more general frameworks, e.g., within a neuro-symbolic system where

ARLC could execute [46] or validate [47] reasoning steps from neural models (e.g., LLMs). Moreover, it would be interesting to tighten the integration between the two systems at the embedding level [48].

References

[1] D.G.T. Barrett, F. Hill, A. Santoro, A.S. Morcos and T. Lillicrap, Measuring abstract reasoning in neural networks, in: *International Conference on Machine Learning (ICML)*, 2018, pp. 511–520.

[2] C. Zhang, F. Gao, B. Jia, Y. Zhu and S.-C. Zhu, RAVEN: A Dataset for Relational and Analogical Visual REasoNing, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[3] S. Hu, Y. Ma, X. Liu, Y. Wei and S. Bai, Stratified Rule-Aware Network for Abstract Visual Reasoning, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[4] Y. Benny, N. Pekar and L. Wolf, Scale-Localized Abstract Reasoning, in: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Nashville, TN, USA, 2021, pp. 12552–12560.

[5] Y. Wu, H. Dong, R. Grosse and J. Ba, The Scattering Compositional Learner: Discovering Objects, Attributes, Relationships in Analogical Reasoning, *arxiv preprint arXiv:2007.04212* (2020).

[6] C. Zhang, B. Jia, S.-C. Zhu and Y. Zhu, Abstract Spatial-Temporal Reasoning via Probabilistic Abduction and Execution, in: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Nashville, TN, USA, 2021, pp. 9731–9741.

[7] C. Zhang, S. Xie, B. Jia, Y.N. Wu, S.-C. Zhu and Y. Zhu, Learning Algebraic Representation for Systematic Generalization in Abstract Reasoning, in: *European Conference on Computer Vision (ECCV)*, Springer, 2022, pp. 692–709.

[8] M. Hersche, M. Zeqiri, L. Benini, A. Sebastian and A. Rahimi, A Neuro-vector-symbolic Architecture for Solving Raven's Progressive Matrices, *Nature Machine Intelligence* **5**(4) (2023), 363–375.

[9] M. Hersche, F. di Stefano, T. Hofmann, A. Sebastian and A. Rahimi, Probabilistic Abduction for Visual Abstract Reasoning via Learning Rules in Vector-symbolic Architectures, in: *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*, 2023.

[10] G. Camposampiero, M. Hersche, A. Terzic, R. Wattenhofer, A. Sebastian and A. Rahimi, Towards Learning Abductive Reasoning using VSA Distributed Representations, in: *International Conference on Neural-Symbolic Learning and Reasoning (NeSy)*, Springer, 2024, pp. 370–385.

[11] Z.-H. Sun, R.-Y. Zhang, Z. Zhen, D.-H. Wang, Y.-J. Li, X. Wan and H. You, Systematic Abductive Reasoning via Diverse Relation Representations in Vector-symbolic Architecture, *arXiv preprint arXiv:2501.11896* (2025).

[12] T. Webb, K.J. Holyoak and H. Lu, Emergent analogical reasoning in large language models, *Nature Human Behaviour* **7**(9) (2023), 1526–1541.

[13] C.E. Stevenson, M. ter Veen, R. Choenni, H.L.J. van der Maas and E. Shutova, Do large language models solve verbal analogies like children do?, *arxiv preprint arXiv:2310.20384* (2023).

[14] G. Gendron, Q. Bao, M. Witbrock and G. Dobbie, Large Language Models Are Not Strong Abstract Reasoners, in: *Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, Vol. 7, 2024, pp. 6270–6278, ISSN: 1045-0823.

[15] M. Lewis and M. Mitchell, Evaluating the Robustness of Analogical Reasoning in Large Language Models, *Transactions on Machine Learning Research* (2025).

[16] X. Cao, B. Lai, W. Ye, Y. Ma, J. Heintz, J. Chen, J. Cao and J.M. Rehg, What is the Visual Cognition Gap between Humans and Multimodal LLMs?, *arXiv preprint arXiv:2406.10424* (2024).

[17] X. Hu, S. Storks, R. Lewis and J. Chai, In-Context Analogical Reasoning with Pre-Trained Language Models, in: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Toronto, Canada, 2023, pp. 1953–1969.

[18] M. Mitchell, A.B. Palmarini and A. Moskvichev, Comparing Humans, GPT-4, and GPT-4V On Abstraction and Reasoning Tasks, in: *AAAI 2024 Workshop on "Are Large Language Models Simply Causal Parrots?"*, 2024.

[19] G. Camposampiero, L. Houmard, B. Estermann, J. Mathys and R. Wattenhofer, Abstract Visual Reasoning Enabled by Language, in: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, Vancouver, BC, Canada, 2023, pp. 2643–2647.

[20] Y. Jiang, J. Zhang, K. Sun, Z. Sourati, K. Ahrabian, K. Ma, F. Ilievski and J. Pujara, MARVEL: Multidimensional Abstraction and Reasoning through Visual Evaluation and Learning, in: *The Thirty-eight Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[21] K. Ahrabian, Z. Sourati, K. Sun, J. Zhang, Y. Jiang, F. Morstatter and J. Pujara, The Curious Case of Nonverbal Abstract Reasoning with Multi-Modal Large Language Models, in: *First Conference on Language Modeling (COLM)*, 2024.

[22] Y. Zhang, H. Bai, R. Zhang, J. Gu, S. Zhai, J.M. Susskind and N. Jaitly, How Far Are We from Intelligent Visual Deductive Reasoning?, in: *ICLR 2024 Workshop: How Far Are We From AGI*, 2024.

[23] A. Wüst, T. Tobiasch, L. Helff, D.S. Dhami, C.A. Rothkopf and K. Kersting, Bongard in Wonderland: Visual Puzzles that Still Make AI Go Mad?, in: *The First Workshop on System-2 Reasoning at Scale, NeurIPS'24*, 2024.

[24] E. Latif, Yifan Zhou, Shuchen Guo, Yizhu Gao, Lehong Shi, M. Nyaaba, Gyeonggeon Lee, L. Zhang et al., A Systematic Assessment of OpenAI o1-Preview for Higher Order Thinking in Education, *arXiv preprint arXiv:2410.21287* (2024), Publisher: Unpublished.

[25] J.C. Raven, J.H. Court and J. Raven, *Raven's progressive matrices*, Oxford Psychologists Press, 1938.

[26] T.A. Plate, Holographic Reduced Representations, *IEEE Transactions on Neural Networks and Learning Systems* **6**(3) (1995).

[27] T.A. Plate, *Holographic Reduced Representations: Distributed Representation for Cognitive Structures*, Center for the Study of Language and Information, Stanford, 2003.
[28] R.W. Gayler, Vector Symbolic Architectures answer Jackendoff's challenges for cognitive neuroscience, in: *Proceedings of the Joint International Conference on Cognitive Science. ICCS/ASCS*, 2003, pp. 133–138.
[29] P. Kanerva, Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors, *Cognitive Computation* **1**(2) (2009), 139–159.
[30] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F.L. Aleman, D. Almeida, J. Altenschmidt, S. Altman et al., GPT-4 Technical Report, *arXiv preprint arXiv:2303.08774* (2024).
[31] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten et al., The Llama 3 Herd of Models, *arxiv preprint arXiv:2407.21783* (2024).
[32] G. Camposampiero, M. Hersche, R. Wattenhofer, A. Sebastian and A. Rahimi, Can Large Reasoning Models do Analogical Reasoning under Perceptual Uncertainty?, in: *International Conference on Neural-Symbolic Learning and Reasoning (NeSy)*, 2025.
[33] DeepSeek-AI et al., DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, *arxiv preprint arXiv:2501.12948* (2025).
[34] OpenAI, Learning to reason with LLMs, 2024.
[35] K.J. Holyoak and R.G. Morrison, *The Oxford Handbook of Thinking and Reasoning*, OUP USA, 2013.
[36] X. Wang, J. Wei, D. Schuurmans, Q. Le, E.H. Chi, S. Narang, A. Chowdhery and D. Zhou, Self-Consistency Improves Chain of Thought Reasoning in Language Models, in: *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.
[37] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, Y. Wu, B. Neyshabur, G. Gur-Ari and V. Misra, Solving Quantitative Reasoning Problems With Language Models, in: *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 35, 2022, pp. 3843–3857.
[38] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, Language Models are Few-Shot Learners, in: *Advances in Neural Information Processing Systems*, Vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan and H. Lin, eds, Curran Associates, Inc., 2020, pp. 1877–1901.
[39] E.P. Frady, D. Kleyko, C.J. Kymn, B.A. Olshausen and F.T. Sommer, Computing on Functions Using Randomized Vector Representations (in brief), in: *Neuro-Inspired Computational Elements Conference*, 2022.
[40] D. Kleyko, D.A. Rachkovskij, E. Osipov and A. Rahimi, A Survey on Hyperdimensional Computing aka Vector Symbolic Architectures, Part I: Models and Data Transformations, *ACM Computing Surveys* **55**(6) (2023), 1–40.
[41] M. Hersche, A. Terzic, G. Karunaratne, J. Langenegger, A. Pouget, G. Cherubini, L. Benini, A. Sebastian and A. Rahimi, Factorizers for Distributed Sparse Block Codes, *Neurosymbolic Artificial Intelligence* (2024).
[42] D.J. Chalmers, R.M. French and D.R. Hofstadter, High-level perception, representation, and analogy: A critique of artificial intelligence methodology, *Journal of Experimental & Theoretical Artificial Intelligence* **4**(3) (1992), 185–211.
[43] Y. Cheng, Context-dependent similarity, in: *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, Elsevier Science Inc., 1990, pp. 41–50.
[44] X. Wu, X. Zhang and X. Shu, Cognitive Deficit of Deep Learning in Numerosity, *Proceedings of the AAAI Conference on Artificial Intelligence* **33**(01) (2019), 1303–1310.
[45] Y. Liu, J. Liu, X. Shi, Q. Cheng, Y. Huang and W. Lu, Let's Learn Step by Step: Enhancing In-Context Learning Ability with Curriculum Learning, *arXiv preprint arXiv:2402.10738* (2024).
[46] L. Pan, A. Albalak, X. Wang and W. Wang, Logic-LM: Empowering Large Language Models with Symbolic Solvers for Faithful Logical Reasoning, in: *Findings of the Association for Computational Linguistics: EMNLP 2023*, Association for Computational Linguistics, Singapore, 2023, pp. 3806–3824.
[47] S. Kambhampati, K. Valmeekam, L. Guan, M. Verma, K. Stechly, S. Bhambri, L.P. Saldyt and A.B. Murthy, Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks, in: *Forty-first International Conference on Machine Learning (ICML)*, 2024.
[48] W. Bounsi, B. Ibarz, A. Dudzik, J.B. Hamrick, L. Markeeva, A. Vitvitskyi, R. Pascanu and P. Veličković, Transformers meet Neural Algorithmic Reasoners, in: *CVPR 2024 Multimodal Algorithmic Reasoning (MAR) Workshop*, 2024.

Appendix A. Prompting details

This appendix provides more details on our prompting strategy. While the prompt design was mainly inspired by [17], we extended it with predictive and discriminative classification and fine-tuned it for the different models. For example, we found that adding a prefix ("Only return the missing number") helped to slightly improve GPT4's accuracy, whereas it reduced Llama-3 70B's performance. Thus, we used individual prompts for the different models.

A.1. Joint attribute querying

As an alternative to individually querying the LLM for predicting the separate attributes, we also devised a joint attribute prompting scheme, shown in Fig. A.6. The attributes of each panel are represented in brackets: (`shape`, `size`, `color`). In this setting, the LLM is required to predict all three attributes of the missing panel at once. For better distinguishing between the different attributes, they are scaled with individual factors ($1\times$, $0.1\times$, $10\times$).

```
System: Complete the Raven's progressive matrix:
User:   Only return the missing numbers!
        row 1: (3,0.5,50), (6,0.5,50), (4,0.5,50);
        row 2: (4,0.3,10), (3,0.3,10), (6,0.3,10);
        row 3: (6,0.1,70), (4,0.1,70), (
Out:    3,0.1,70)
```

Fig. A.6. Example prompt for joint prediction of all three attributes.

A.2. Discriminative classification approach

Fig. A.7 shows an example prompt for performing discriminative classification. As shown, the answers only contain two distinct values ("6" and "7"); finding the correct answer requires the consideration of all attributes. For choosing the final answer, we extract all attribute values that correspond to the predicted answer (e.g., value "7" for `shape`) and select the best matching answer candidate, i.e., the answer with the highest number of overlaps with the predicted attributes.

```
System: Complete the Raven's progressive matrix:
User:   row 1: 4, 4, 4;
        row 2: 6, 6, 6;
        row 3: 7, 7,

        Select the correct Answer from the following list
        Answer #0: 7
        Answer #1: 6
        Answer #2: 7
        Answer #3: 7
        Answer #4: 6
        Answer #5: 6
        Answer #6: 7
        Answer #7: 6

        Solution: The correct answer is Answer #
Output: 0: 7
```

Fig. A.7. Example prompt for discriminative classification approach, where the answer candidates are provided. The underlying attribute is `shape` and the rule is `constant`.

Appendix B. Vector-symbolic architectures

Vector-symbolic architectures (VSAs) [26–29] are a family of computational models that rely on the mathematical properties of high-dimensional vector spaces. VSAs make use of high-dimensional distributed representations for structured (symbolic) representation of data while maintaining the advantages of connectionist distributed vector representations (see [40] for a survey). Here is a formal definition of VSAs:

Definition 1 (VSA)**.** *A vector-symbolic architecture (VSA) consists of a 4-tuple* $\mathbb{V} = (\mathbb{C}, \oplus, \otimes, \odot)$*, where* $\mathbb{C}$ *is a set of high-dimensional distributed vectors equipped with two main operations,* $\oplus$ *(bundling) and* $\otimes$ *(binding), and on which it is possible to define a similarity measure* $\odot$*.*

Appendix C. Analysis of arithmetic errors

This appendix aims to find explanations for LLM's errors by analyzing the structure behind the predicted answers. A recent study [13] showed that LLMs tend to solve verbal analogy problems in an associative way instead of performing proper relational mapping. The associative reasoning can be explained as ignoring the source domain and solving the task directly at the target domain (e.g., only looking at the possible solutions without reading the questions). Interestingly, children tend to perform associative reasoning, whereas adults opt for relational mapping.

In RPMs, the source domain can be defined as the first two rows (with values $x_{1,1}, x_{1,2}, x_{1,3}$ and $x_{2,1}, x_{2,2}, x_{2,3}$), whereby the target domain is the last row ($x_{3,1}, x_{3,2}$). Therefore, an associative reasoner would only look at the last row to solve the task. In the following, we aim to find potential incorrect rules that the LLMs may have been inferred from the last row(s):

- `constant`: The values of the last row are identical ($x_{3,1} = x_{3,2}$), and the model predicts $\hat{x}_{3,3} = x_{3,2} = x_{3,1}$
- `progression`: The values of the last row differ by $\delta = x_{3,2} - x_{3,1}$, and the model predicts $\hat{x}_{3,3} = x_{3,2} + \delta$
- `short constant`: The model just copies the penultimate value: $\hat{x}_{3,3} = x_{3,2}$.
- `short distribute three`: Assuming a distribute three over the last two rows: $x_{3,1} \in \{x_{2,1}, x_{2,2}, x_{2,3}\}$, $x_{3,2} \in \{x_{2,1}, x_{2,2}, x_{2,3}\}$, and hence $\hat{x}_{3,3} \in \{x_{2,1}, x_{2,2}, x_{2,3}\}$.

Fig. C.8 shows the resulting confusion matrix summarizing all the attributes. The `arithmetic` rule has fewer occurrences as this rule is not integrated in the attribute `shape`. As already stated in the main text, the majority of wrong predictions are related to the `arithmetic` rule. For GPT-4, our new rule interpretations can explain 32 out of the 68 errors, while 36 errors remain unknown. Llama-3 70B showed many more errors in the arithmetic rule; here, we can explain 57 out of 142 errors with relational reasoning. In summary, some (40.1–47.1%) of the LLM's errors can be rooted in relational reasoning. Further understanding the behavior of the unknown rules is scope for future work.

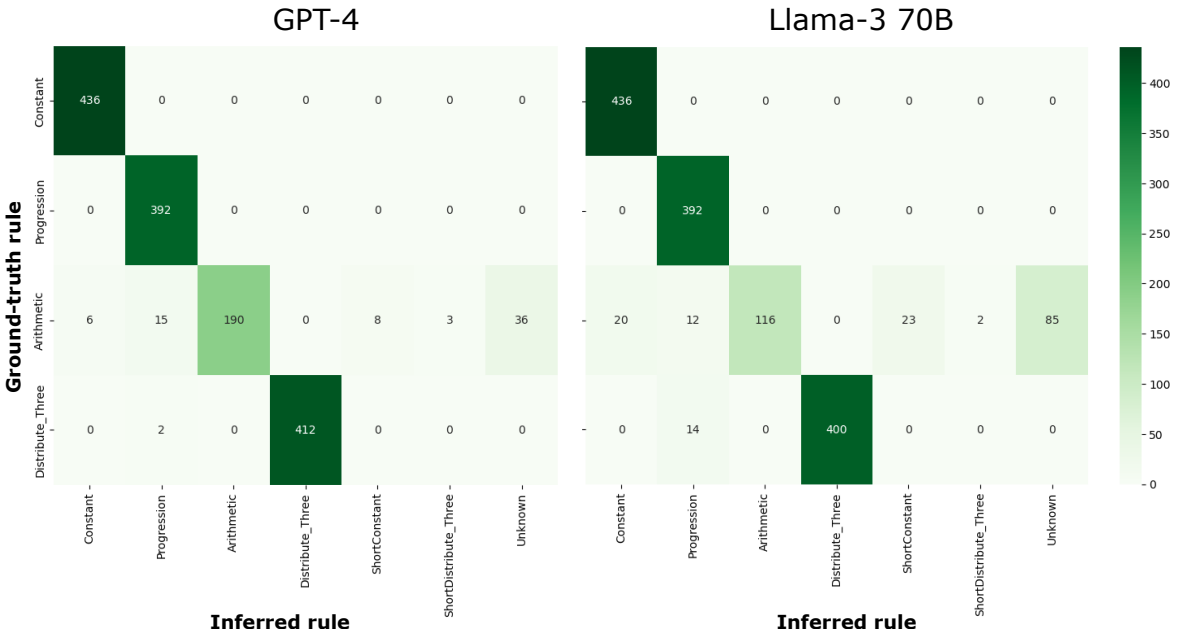


Fig. C.8. Rule confusion matrix of GPT-4 (left) and Llama-3 70B (right).

Appendix D. ARLC programmed weights

This appendix provides the rules programmed in $\text{ARLC}_{\text{progr}}$ for both I-RAVEN and I-RAVEN-X in Tables D.7 and D.8, respectively. All the terms of the general rule template in Equation (4) that are not explicitly filled in each rule are either set to $\mathbf{e}$, the identity element for the binding operation, or with terms that cancel out. In practice, the programming of the terms of each rule is performed by setting the weight in the corresponding position to a high positive value ($1e+5$) and all the other terms to 0.

Table D.7
I-RAVEN rules programmed in $\text{ARLC}_{\text{progr}}$.

RPM Rule	Programmed rule
`constant`	$(\mathbf{x}_a^1) \oslash (\mathbf{e})$
`progression`	$(\mathbf{x}_a^2 \otimes \mathbf{x}_a^2) \oslash (\mathbf{x}_a^1)$
`arithmetic plus`	$(\mathbf{x}_a^1 \otimes \mathbf{x}_a^2) \oslash (\mathbf{e})$
`arithmetic minus`	$(\mathbf{x}_a^1) \oslash (\mathbf{x}_a^2)$
`distribute three`	$(\mathbf{o}_a^1 \otimes \mathbf{o}_a^2 \otimes \mathbf{o}_a^3) \oslash (\mathbf{x}_a^1 \otimes \mathbf{x}_a^2)$

Table D.8
I-RAVEN-X rules programmed in $\text{ARLC}_{\text{progr}}$.

RPM Rule	Programmed rule
`constant`	$(\mathbf{x}_a^1) \oslash (\mathbf{e})$
`progression`	$(\mathbf{x}_a^9 \otimes \mathbf{x}_a^2) \oslash (\mathbf{x}_a^1)$
`arithmetic plus`	$(\mathbf{x}_a^1 \otimes \mathbf{x}_a^2 \otimes \mathbf{x}_a^3 \otimes \mathbf{x}_a^4 \otimes \mathbf{x}_a^5 \otimes \mathbf{x}_a^6 \otimes \mathbf{x}_a^7 \otimes \mathbf{x}_a^8 \otimes \mathbf{x}_a^9) \oslash (\mathbf{e})$
`arithmetic minus`	$(\mathbf{x}_a^1) \oslash (\mathbf{x}_a^2 \oslash \mathbf{x}_a^3 \oslash \mathbf{x}_a^4 \oslash \mathbf{x}_a^5 \oslash \mathbf{x}_a^6 \oslash \mathbf{x}_a^7 \oslash \mathbf{x}_a^8 \oslash \mathbf{x}_a^9)$
`distribute-n`	$(\mathbf{o}_a^1 \otimes \mathbf{o}_a^2 \otimes \mathbf{o}_a^3 \otimes \mathbf{o}_a^4 \otimes \mathbf{o}_a^5 \otimes \mathbf{o}_a^6 \otimes \mathbf{o}_a^7 \otimes \mathbf{o}_a^8 \otimes \mathbf{o}_a^9 \otimes \mathbf{o}_a^{10}) \oslash (\mathbf{x}_a^1 \otimes \mathbf{x}_a^2 \otimes \mathbf{x}_a^3 \otimes \mathbf{x}_a^4 \otimes \mathbf{x}_a^5 \otimes \mathbf{x}_a^6 \otimes \mathbf{x}_a^7 \otimes \mathbf{x}_a^8 \otimes \mathbf{x}_a^9)$