Abstract Visual Reasoning: An Algebraic Approach for Solving Raven's Progressive Matrices

Jingyi Xu[1*] Tushar Vaidya[2°*] Yufei Wu[2°*] Saket Chandra[1] Zhangsheng Lai[3°] Kai Fong Ernest Chong[1†]
[1]Singapore University of Technology and Design
[2]Nanyang Technological University [3]Singapore Polytechnic
jingyi_xu@mymail.sutd.edu.sg tushar.vaidya@ntu.edu.sg yufei002@e.ntu.edu.sg
lai_zhangsheng@sp.edu.sg {saket_chandra,ernest_chong}@sutd.edu.sg

Abstract

We introduce algebraic machine reasoning, a new reasoning framework that is well-suited for abstract reasoning. Effectively, algebraic machine reasoning reduces the difficult process of novel problem-solving to routine algebraic computation. The fundamental algebraic objects of interest are the ideals of some suitably initialized polynomial ring. We shall explain how solving Raven's Progressive Matrices (RPMs) can be realized as computational problems in algebra, which combine various well-known algebraic subroutines that include: Computing the Gröbner basis of an ideal, checking for ideal containment, etc. Crucially, the additional algebraic structure satisfied by ideals allows for more operations on ideals beyond set-theoretic operations.

Our algebraic machine reasoning framework is not only able to select the correct answer from a given answer set, but also able to generate the correct answer with only the question matrix given. Experiments on the I-RAVEN dataset yield an overall 93.2% *accuracy, which significantly outperforms the current state-of-the-art accuracy of* 77.0% *and exceeds human performance at* 84.4% *accuracy.*

1. Introduction

When we think of machine reasoning, nothing captures our imagination more than the possibility that machines would eventually surpass humans in intelligence tests and general reasoning tasks. Even for humans, to excel in IQ tests, such as the well-known Raven's progressive matrices (RPMs) [5], is already a non-trivial feat. A typical RPM instance is composed of a question matrix and an answer set; see Fig. 1. A question matrix is a 3×3 grid of panels

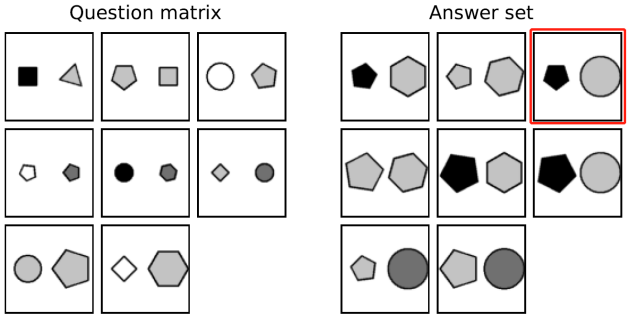

Figure 1. An example of RPM instance from the I-RAVEN dataset. The correct answer is marked with a red box.

that satisfy certain hidden rules, where the first 8 panels are filled with geometric entities, and the 9-th panel is "missing". The goal is to infer the correct answer for this last panel from among the 8 panels in the given answer set.

The ability to solve RPMs is the quintessential display of what cognitive scientists call fluid intelligence. The word "fluid" alludes to the mental agility of discovering new relations and abstractions [28], especially for solving *novel* problems not encountered before. Thus, it is not surprising that abstract reasoning on *novel* problems is widely hailed as the hallmark of human intelligence [6].

Although there has been much recent progress in machine reasoning [15, 17, 30–33, 37, 38, 46, 47], a common criticism [9, 25, 26] is that existing reasoning frameworks have focused on approaches involving extensive training, even when solving well-established reasoning tests such as RPMs. Perhaps most pertinently, as [9] argues, reasoning tasks such as RPMs should not need task-specific perfor-

*Equal contributions. † Corresponding author.
° This work was done when the author was previously at SUTD.
Code: https://github.com/Xu-Jingyi/AlgebraicMR

This work is supported by the National Research Foundation, Singapore under its AI Singapore Program (AISG Award No: AISG-RP-2019-015) and under its NRFF Program (NRFFAI1-2019-0005), and by Ministry of Education, Singapore, under its Tier 2 Research Fund (MOE-T2EP20221-0016).

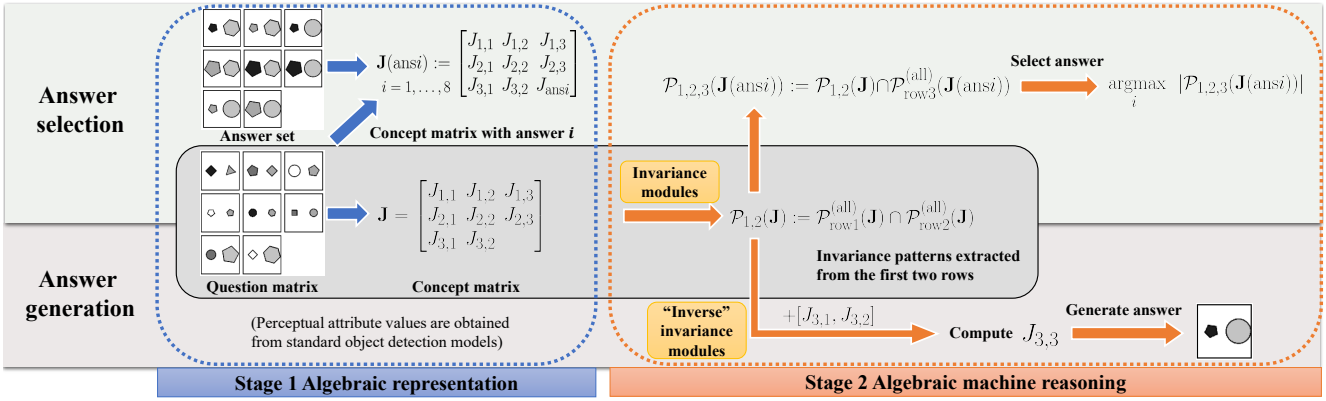

Figure 2. An overview of our algebraic machine reasoning framework, organized into 2 stages.

mance optimization. After all, if a machine optimizes performance by training on task-specific data, then that task cannot possibly be novel to the machine.

To better emulate human reasoning, we propose what we call "algebraic machine reasoning", a new reasoning framework that is well-suited for abstract reasoning. Our framework solves RPMs *without* needing to optimize for performance on task-specific data, analogous to how a gifted child solves RPMs without needing practice on RPMs. Our key starting point is to *define* concepts as ideals of some suitably initialized polynomial ring. These ideals are treated as the "actual objects of study" in algebraic machine reasoning, which do not require any numerical values to be assigned to them. We shall elucidate how the RPM task can be realized as a computational problem in algebra involving ideals.

Our reasoning framework can be broadly divided into two stages: (1) algebraic representation, and (2) algebraic machine reasoning; see Fig. 2. In the first stage, we represent RPM panels as ideals, based on perceptual attribute values extracted from object detection models. In the second stage, we propose 4 invariance modules to extract patterns from the RPM question matrix.

To summarize, our main contributions are as follows:

- We reduce "solving the RPM task" to "solving a computational problem in algebra". Specifically, we present how the discovery of abstract patterns can be realized very concretely as algebraic computations known as primary decompositions of ideals.
- In our algebraic machine reasoning framework, we introduce 4 invariance modules for extracting patterns that are meaningful to humans.
- Our framework is not only able to select the correct answer from a given answer set, but also able to generate answers *without needing any given answer set*.
- Experiments conducted on RAVEN and I-RAVEN datasets demonstrate that our reasoning framework significantly outperforms state-of-the-art methods.

2. Related Work

RPM solvers. There has been much recent interest in solving RPMs with deep-learning-based methods [15, 23, 32, 47, 48, 51–54]. Most methods extract features from raw RPM images using nueral networks, and select answers by measuring panel similarities. Several works instead focus on generating correct answers without needing the answer set [27, 34]. To evaluate the reasoning capabilities of these methods, RPM-like datasets such as PGM [32] and RAVEN [46] have been proposed. Subsequently, I-RAVEN [12] and RAVEN-FAIR [3] are introduced to overcome a shortcut flaw in the answer set generation of RAVEN.

Algebraic methods in AI. Using algebraic methods in AI is not new. Systems of polynomial equations are commonly seen in computer vision [29] and robotics [8], which are solved algebraically via Gröbner basis computations. In statistical learning theory, methods in algebraic geometry [41] and algebraic statistics [10] are used to study singularities in statistical models [22, 42, 43, 45], to analyze generalization error in hierarchical models [39, 40], to learn invariant subspaces of probability distributions [18, 20], and to model Bayesian networks [11, 36]. A common theme in these works is to study suitably defined algebraic varieties. In deep learning, algebraic methods are used to study the expressivity of neural nets [7, 16, 24, 50]. In automated theorem proving, Gröbner basis computations are used in proof-checking [35]. Recently, a matrix formulation of first-order logic was applied to the RPM task [49], where relations are approximated by matrices and reasoning is framed as a bilevel optimization task to find best-fit matrix operators. As far as we know, methods from commutative algebra have not been used in machine reasoning.

3. Proposed Algebraic Framework

In abstract reasoning, a key cognitive step is to "discover patterns from observations", which can be formulated con-

cretely as "finding invariances in observations". In this section, we describe how algebraic objects known as ideals are used to represent RPM instances, how patterns are extracted from such algebraic representations, and how RPMs can be solved, both for answer selection and answer generation, as computational problems in algebra.

3.1. Preliminaries

Throughout, let $R = \mathbb{R}[x_1, \ldots, x_n]$ be the ring of polynomials in variables $x_1, \ldots, x_n$, with real coefficients. In particular, R is closed under addition and multiplication of polynomials, i.e., for any $a, b \in R$, we have $a + b, ab \in R$.

3.1.1 Algebraic definitions

Ideals in polynomial rings. A subset $I \subseteq R$ is called an *ideal* if there exist polynomials $g_1, \ldots, g_k$ in R such that

$$I = \{f_1 g_1 + \cdots + f_k g_k | f_1, \ldots, f_k \in R\}$$

contains all polynomial combinations of $g_1, \ldots, g_k$. We say that $\mathcal{G} = \{g_1, \ldots, g_k\}$ is a *generating set* for I, we call $g_1, \ldots, g_k$ *generators*, and we write either $I = \langle g_1, \ldots, g_k \rangle$ or $I = \langle \mathcal{G} \rangle$. Note that generating sets of ideals are not unique. If I has a generating set consisting only of monomials, then we say that I is a *monomial ideal*. (Recall that a *monomial* is a polynomial with a single term.) Given ideals $J_1 = \langle g_1, \ldots, g_k \rangle$ and $J_2 = \langle h_1, \ldots, h_\ell \rangle$, there are three basic operations (sums, products, intersections):

$$J_1 + J_2 := \langle g_1, \ldots, g_k, h_1, \ldots, h_\ell \rangle;$$
$$J_1 J_2 := \langle \{g_i h_j | 1 \le i \le k, 1 \le j \le \ell\} \rangle;$$
$$J_1 \cap J_2 := \{r \in R : r \in J_1 \text{ and } r \in J_2\}.$$

Most algebraic computations involving ideals, especially "advanced" operations (e.g. primary decompositions), require computing their Gröbner bases as a key initial step. More generally, Gröbner basis computation forms the backbone of most algorithms in algebra; see Appendix A.2.

Primary decompositions. In commutative algebra, primary decompositions of ideals are a far-reaching generalization of the idea of prime factorization for integers. Its importance to algebraists cannot be overstated. Informally, every ideal J has a decomposition $J = J_1 \cap \cdots \cap J_s$ as an intersection of finitely many *primary* ideals. This intersection is called a *primary decomposition* of J, and each J_j is called a *primary component* of the decomposition. In the special case when J is a monomial ideal, there is an **unique** minimal primary decomposition with maximal monomial primary components [2]; We denote this unique set of primary components by $\text{pd}(J)$. See Appendix A.3 for details.

3.1.2 Concepts as monomial ideals

We define a *concept* to be a monomial ideal of R. In particular, the zero ideal $\langle 0 \rangle \subseteq R$ is the concept "null", and could be interpreted as "impossible" or "nothing", while the ideal $\langle 1 \rangle = R$ is the concept "conceivable", and could be interpreted as "possible" or "everything". Given a concept $J \subseteq R$, a monomial in J is called an *instance* of the concept. For example, $x_{\text{black}} x_{\text{square}}$ is an instance of $\langle x_{\text{square}} \rangle$ (the concept "square"). For each x_i, we say $\langle x_i \rangle \subseteq R$ is a *primitive* concept, and x_i is a *primitive* instance.

Theorem 3.1. *There are infinitely many concepts in R, even though there are finitely many primitive concepts in R. Furthermore, if $J \subseteq R$ is a concept, then the following hold:*

(i) J has infinitely many instances, unless $J = \langle 0 \rangle$.

(ii) J has a unique minimal generating set consisting of finitely many instances, which we denote by $\text{mingen}(J)$.

(iii) If $J \neq \langle 1 \rangle$, then J has a unique set of associated concepts $\{P_1, \ldots, P_k\}$, together with a unique minimal primary decomposition $J = J_1 \cap \cdots \cap J_k$, such that each J_i is a concept contained in P_i, that is maximal among all possible primary components contained in P_i that are concepts.

See Appendix A.4 for a proof of Theorem 3.1 and for more details on why defining concepts as monomial ideals captures the expressiveness of concepts in human reasoning.

3.2. Stage 1: Algebraic representation

We shall use the RPM instance depicted in Fig 1 as our running example, to show the entire algebraic reasoning process: (1) algebraic representation; and (2) algebraic machine reasoning. In this subsection, we focus on the first stage. Recall that every RPM instance is composed of 16 panels filled with geometric entities. For our running example, each entity can be described using 4 attributes: "color", "size", "type", and "position". We also need one additional attribute to represent the "number" of entities in the panel.

3.2.1 Attribute concepts

In human cognition, certain semantically similar concepts are naturally grouped to form a more general concept. For example, concepts such as "red", "green", "blue", "yellow", etc., can be grouped to form a new concept that represents "color". Intuitively, we can think of "color" as an attribute, and "red", "green", "blue", "yellow" as attribute values.

For our running example, the 5 attributes are represented by 5 concepts (monomial ideals). In general, all possible values for each attribute are encoded as generators for the concept representing that attribute. However, for ease of explanation, we shall consider only those attribute values that are involved in Fig. 1 to explain our example:

$$\mathcal{A}_{\text{num}} := \{x_{\text{one}}, x_{\text{two}}\},$$
$$\mathcal{A}_{\text{pos}} := \{x_{\text{left}}, x_{\text{right}}\},$$
$$\mathcal{A}_{\text{type}} := \{x_{\text{triangle}}, x_{\text{square}}, x_{\text{pentagon}}, x_{\text{hexagon}}, x_{\text{circle}}\},$$
$$\mathcal{A}_{\text{color}} := \{x_{\text{white}}, x_{\text{gray}}, x_{\text{dgray}}, x_{\text{black}}\},$$
$$\mathcal{A}_{\text{size}} := \{x_{\text{small}}, x_{\text{avg}}, x_{\text{large}}\}.$$

Let $\mathcal{L} := \{\text{num}, \text{pos}, \text{type}, \text{color}, \text{size}\}$ be the set of attribute labels, and let $\mathcal{A}_{\text{all}} := \bigcup_{\ell \in \mathcal{L}} \mathcal{A}_\ell$. Initialize the ring

$R := \mathbb{R}[\mathcal{A}_{\text{all}}]$ of all polynomials on the variables in $\mathcal{A}_{\text{all}}$ with real coefficients. For each $\ell \in \mathcal{L}$, let J_ℓ be the concept $\langle \mathcal{A}_\ell \rangle \subseteq R$. These concepts, which we call *attribute concepts*, are task-specific. We assume humans tend to discover and organize complex patterns in terms of attributes. Thus for pattern extraction, we shall use the inductive bias that a concept representing a pattern is deemed meaningful if it is in some attribute concept.

3.2.2 Representation of RPM panels

In order to encode the RPM images algebraically, we first need to train perception modules to extract attribute information directly from raw images. One possible approach for perception, as used in our experiments, is to train 4 RetinaNet models (each with a ResNet-50 backbone) separately for all 4 attributes except "number", which can be directly inferred by counting the number of bounding boxes.

After extracting attribute values for entities, we can represent each panel as a concept. For example, the top-left panel of the RPM in Fig. 1 can be encoded as the concept $J_{1,1} = \langle x_{\text{two}} x_{\text{left}} x_{\text{square}} x_{\text{black}} x_{\text{avg}}, x_{\text{two}} x_{\text{right}} x_{\text{triangle}} x_{\text{gray}} x_{\text{avg}} \rangle$ in the polynomial ring R. Here, $J_{1,1}$ represents a panel with two entities, a black square of average size on the left, and a gray triangle of average size on the right. The indices in $J_{1,1}$ tell us that the panel is in row 1, column 1. Similarly, we can encode the remaining 7 panels of the question matrix as concepts $J_{1,2}, J_{1,3}, \ldots, J_{3,2}$ and encode the 8 answer options as concepts $J_{\text{ans}1}, \ldots, J_{\text{ans}8}$. In general, every monomial generator of each concept describes an entity in the associated panel.

The list of 8 concepts $\mathbf{J} = [J_{1,1}, \ldots, J_{3,2}]$ shall be called a *concept matrix*; this represents the RPM question matrix with a missing 9-th panel. Let $\mathbf{J}_i := [J_{i,1}, J_{i,2}, J_{i,3}]$ (for $i = 1, 2$) represent the i-th row in the question matrix.

3.3. Stage 2: algebraic machine reasoning

Previously in Section 3.2, we have already encoded the question matrix in an RPM instance as a concept matrix $\mathbf{J} = [J_{1,1}, \ldots, J_{3,2}]$. In this subsection, we will introduce the reasoning process of our algebraic framework.

Our goal of extracting patterns for a single row of $\mathbf{J}$ can be mathematically formulated as "finding invariance" across the concepts that represent the panels in this row. (The same process can be applied to columns.) This seemingly imprecise idea of "finding invariance" can be realized very concretely via the computation of primary decompositions. Ideally, we want to extract patterns that are meaningful to humans. Hence we have designed 4 invariance modules to mimic human cognition in pattern recognition.

3.3.1 Prior knowledge

To use algebraic machine reasoning, we adopt:

- Inductive bias of attribute concepts (see Section 3.2.1);
- Useful binary operations on numerical values;
- Functions that map concepts to concepts.

There are numerous binary operations, such as $+, -, \times, \div, \min, \max$, etc., that can be applied to numerical values extracted from concepts. For the RPM task, we use $+, -$.

In algebra, the study of functions between algebraic objects is a productive strategy for understanding the underlying algebraic structure. Analogously, we shall use maps on concepts to extract complex patterns. For the RPM task, we need to cyclically order the values in $\mathcal{A}_\ell$ for each attribute $\ell \in \mathcal{L}$ before we can extract sequential information. To encode the idea of "next", we introduce the function $f_{\text{next}}(J|\Delta)$ defined on concepts J, where Δ represents the step-size. Each variable $x \in \mathcal{A}_\ell$ that appears in a generator of J is mapped to the Δ-th variable after x, w.r.t. the cyclic order on $\mathcal{A}_\ell$. For example, $f_{\text{next}}(\langle x_{\text{square}} x_{\text{gray}} x_{\text{avg}} \rangle | 1) = \langle x_{\text{pentagon}} x_{\text{dgray}} x_{\text{large}} \rangle$, and $f_{\text{next}}(\langle x_{\text{square}} \rangle | -2) = \langle x_{\text{circle}} \rangle$.

3.3.2 Reasoning via primary decompositions

Given concepts $J_1, \ldots, J_k$ that share a common "pattern", how do we extract this pattern? Abstractly, a common pattern can be treated as a concept K that contains all of these concepts $J_1, \ldots, J_k$. If there are several common patterns $K_1, \ldots, K_r$, then each concept J_i can be "decomposed" as $J_i = K_1 \cap \cdots \cap K_r \cap K_i'$ for some ideal K_i'. Thus, we have the following algebraic problem: Given $J_1, \ldots, J_k$, compute their common components $K_1, \ldots, K_r$.

Recall that a concept J has a unique minimal primary decomposition, since concepts are monomial ideals. Thus, to extract the common patterns of concepts $J_1, \ldots, J_k$, we first have to compute $\text{pd}(J_1), \ldots, \text{pd}(J_k)$, then extract the common primary components. The intersection of (any subset of) these common components would yield a new concept, which can be interpreted as a common pattern of the concepts $J_1, \ldots, J_k$. As part of our inductive bias, we are only interested in those primary components that are contained in attribute concepts. See Appendix A.3 for further details.

3.3.3 Proposed invariance modules

Our 4 proposed invariance modules are: (1) intra-invariance module, (2) inter-invariance module, (3) compositional invariance module, and (4) binary-operator invariance module. Intuitively, they check for 4 general types of invariances across a sequence of concepts $J_1, \ldots, J_k$ (e.g. a row $\mathbf{J}_i = [J_{i,1}, J_{i,2}, J_{i,3}]$ for the RPM task). Such invariances apply not just to the RPM task, but could be applied to other RPM-like tasks, e.g. based on different prior knowledge, different grid layouts, etc. Full computational details for our running example can be found in Appendix B.3.

1. Intra-invariance module extracts patterns where the set of values for some attribute within concept J_i remains invariant over all i. First, we define $J_+ := J_1 + \cdots + J_k$ and $J_\cap := J_1 \cap \cdots \cap J_k$; see Section 3.1.1. Intuitively, J_+ and

$J_\cap$ are concepts that capture information about the entire sequence $J_1, \ldots, J_k$ in two different ways. Next, we compute the common primary components of J_+ and $J_\cap$ that are contained in attribute concepts. Finally, we return the attributes associated to these common primary components: $\mathcal{P}_{\text{intra}}([J_1 \ldots J_k]) := \{\text{attr} \in \mathcal{L} \mid \exists I \in \text{pd}(J_+) \cap \text{pd}(J_\cap), I \subseteq \langle \mathcal{A}_{\text{attr}} \rangle\}$.

2. Inter-invariance module extracts patterns arising from the set difference between $\text{pd}(J_\cap)$ and $\text{pd}(J_+)$. Thereafter, we check for the invariance of these extracted patterns across multiple sequences. The extracted set of patterns is:

$$\mathcal{P}_{\text{inter}}([J_1, \ldots, J_k]) := \left\{ (\text{attr}, \mathcal{I}) \,\middle|\, \begin{array}{c} \mathcal{I} \subseteq \text{pd}(J_\cap) - \text{pd}(J_+), \\ \text{attr} \in \mathcal{L}, I \subseteq \langle \mathcal{A}_{\text{attr}} \rangle \; \forall I \in \mathcal{I} \end{array} \right\},$$

where $\mathcal{I}$ is a set of concepts, and "$-$" refers to set difference. We omit $\text{pd}(J_+)$ so that we do not overcount the patterns already extracted in the previous module. Informally, for each pair $(\text{attr}, \mathcal{I})$, the concepts in $\mathcal{I}$ can be interpreted as those "primary" concepts that correspond to at least one of $J_1, \ldots, J_k$, that do not correspond to all of $J_1, \ldots, J_k$, and that are contained in $\langle \mathcal{A}_{\text{attr}} \rangle$.

3. Compositional invariance module extracts patterns arising from invariant attribute values in the following new sequence of concepts:

$$[J'_1, \ldots, J'_k] = [f^{k-1}(J_1), f^{k-2}(J_2), \ldots, f(J_{k-1}), J_k],$$

where f is some given function. Intuitively, for such patterns, there are some attributes whose values are invariant in $[f(J_i), J_{i+1}]$ for all $i = 1, \ldots, k-1$. By checking the intersection of primary components of the concepts in the new sequence, the extracted set of patterns is given by:

$$\mathcal{P}_{\text{comp}}([J_1, \ldots, J_k]) := \left\{ (\text{attr}, f) \,\middle|\, \begin{array}{c} \exists I \in \bigcap_{i=1}^{k} \text{pd}(f^{k-i}(J_i)), \\ \text{attr} \in \mathcal{L}, I \subseteq \langle \mathcal{A}_{\text{attr}} \rangle \end{array} \right\}.$$

The given function used for the RPM task is $f_{\text{next}}(\cdot|\Delta)$, where Δ represents the number of steps; see Section 3.3.1.

4. Binary-operator module extracts numerical patterns, based on a given real-valued function g on concepts, and a given set Λ of binary operators. The extracted patterns are:

$$\mathcal{P}_{\text{binary}}(\mathbf{J}_i) := \left\{ \overline{\oslash} \,\middle|\, \begin{array}{c} \overline{\oslash} = [\oslash_1, \ldots, \oslash_{k-2}], \; \oslash_i \in \Lambda, \\ g(J_1) \oslash_1 \cdots \oslash_{k-2} g(J_{k-1}) = g(J_k) \end{array} \right\}.$$

3.3.4 Extracting row-wise patterns

Given a concept matrix $\mathbf{J} = [J_{1,1}, \ldots, J_{3,2}]$, how do we extract the patterns from its i-th row? We first begin by extracting the common position values among all 8 panels:

$$\text{comPos}(\mathbf{J}) := \{p \in \mathcal{A}_{\text{pos}} \mid \exists I \in \textstyle\bigcap_{J \in \mathbf{J}} \text{pd}(J), p \in I\}$$

For each common position $p \in \text{comPos}(\mathbf{J})$, we generate two new concept matrices $\bar{\mathbf{J}}^{(p)}$ and $\hat{\mathbf{J}}^{(p)}$, such that:

- Each concept $\bar{J}^{(p)}_{i,j}$ in $\bar{\mathbf{J}}^{(p)}$ is generated by the unique generator in $J_{i,j}$ that is divisible by p;
- Each concept $\hat{J}^{(p)}_{i,j}$ in $\hat{\mathbf{J}}^{(p)}$ is generated by all generators in $J_{i,j}$ that are not divisible by p.

(Recall that generators of a concept are polynomials.)

Informally, we are splitting each panel in the RPM image into 2 panels, one that contains only the entity in the common position p, and the other that contains all remaining entities not in position p. This step allows us to reason about rules that involve only a portion of the panels.

Consequently, if $\text{comPos}(\mathbf{J}) = \{p_1, \ldots, p_k\}$, then we can extend the single concept matrix into a list of concept matrices $[\mathbf{J}, \bar{\mathbf{J}}^{(p_1)}, \hat{\mathbf{J}}^{(p_1)}, \ldots, \bar{\mathbf{J}}^{(p_k)}, \hat{\mathbf{J}}^{(p_k)}]$.

For each concept matrix $\check{\mathbf{J}}$ from the extended list, we consider its i-th row $\check{\mathbf{J}}_i = [\check{J}_{i,1}, \check{J}_{i,2}, \check{J}_{i,3}]$ (left-to-right) and extract patterns from $\check{\mathbf{J}}_i$ via the 4 modules from Section 3.3.3. Let $\mathcal{P}(\check{\mathbf{J}}_i)$ be the set of all such patterns, i.e., $\mathcal{P}(\check{\mathbf{J}}_i) := \mathcal{P}_{\text{intra}}(\check{\mathbf{J}}_i) \cup \mathcal{P}_{\text{inter}}(\check{\mathbf{J}}_i) \cup \mathcal{P}_{\text{comp}}(\check{\mathbf{J}}_i) \cup \mathcal{P}_{\text{binary}}(\check{\mathbf{J}}_i)$.

Finally, for row $i = 1, 2$, we define

$$\mathcal{P}^{(\text{all})}_i(\mathbf{J}) := \textstyle\bigcup_{\check{\mathbf{J}}} \{(K, \check{\mathbf{J}}) \mid K \in \mathcal{P}(\check{\mathbf{J}}_i)\}, \qquad (1)$$

where the union ranges over all concept matrices $\check{\mathbf{J}}$ in the extended list, i.e. $\check{\mathbf{J}} \in [\mathbf{J}, \bar{\mathbf{J}}^{(p_1)}, \hat{\mathbf{J}}^{(p_1)}, \ldots, \bar{\mathbf{J}}^{(p_k)}, \hat{\mathbf{J}}^{(p_k)}]$. Note that $\mathcal{P}^{(\text{all})}_i(\mathbf{J})$ can be regarded as all the patterns extracted from the i-th row of the original concept matrix $\mathbf{J}$. If instead $\mathbf{J} = [J_{1,1}, \ldots, J_{3,3}]$ is a list containing 9 concepts, then we can define $\mathcal{P}^{(\text{all})}_3(\mathbf{J})$ analogously.

Algorithm 1 Answer selection.

Inputs: Concept matrix $\mathbf{J} = [J_{1,1} \ldots J_{3,2}]$, and associated answer set $[J_{\text{ans}1}, \ldots, J_{\text{ans}8}]$.

```
Initialize comPattern = [0, ..., 0]_{1×8}.
Compute P_{1,2}(J) := P_1^(all)(J) ∩ P_2^(all)(J).      // see (1)
for i from 1 to 8 do
    J ← [J_{1,1}, ..., J_{3,2}, J_{ans i}]
    Compute P_3^(all)(J).
    comPattern[i] ← |P_{1,2}(J) ∩ P_3^(all)(J)|
return answer index i = argmax_{i'} comPattern[i'].
```

3.4. Solving RPMs

3.4.1 Answer selection

In Section 3.3.4, we described how row-wise patterns can be extracted using the 4 invariance modules. Thus, a natural approach for answer selection is to determine which answer option, when inserted in place of the missing panel, would maximize the number of patterns that are common to all three rows. Consequently, answer selection is reduced to a simple optimization problem; see Algorithm 1.

3.4.2 Answer generation

Since our algebraic machine reasoning framework is able to extract common patterns that are meaningful to humans, hidden in the raw RPM images, it provides a new way to generate answers without needing a given answer set. This is similar to a gifted human who is able to solve the RPM task, by first recognizing the patterns in the first two rows, then inferring what the missing panel should be. Intuitively, we are applying "inverse" operations of the 4 invariance modules to generate the concept representing the missing panel; see Algorithm 2 for an overview.

Briefly speaking, for a given RPM concept matrix $\mathbf{J}$, we first compute the common patterns among the first two rows via $\mathcal{P}_{1,2}(\mathbf{J}) := \mathcal{P}_1^{(\text{all})}(\mathbf{J}) \cap \mathcal{P}_2^{(\text{all})}(\mathbf{J})$; see (1). Each element in $\mathcal{P}_{1,2}(\mathbf{J})$ is a pair $(K, \check{\mathbf{J}})$, where K is a common pattern (for rows 1 and 2) specific to one attribute, and $\check{\mathbf{J}}$ is the corresponding concept matrix. (This represents the *difficult* step of pattern discovery by a gifted human.) Then, we go through all common patterns to compute the attribute values for the missing 9th panel. (This represents a *routine* consistency check of the discovered patterns; see Appendix B.2 for full algorithmic details, and B.3 for an example.)

In general, when integrating all the attribute values for $J_{3,3}$ derived from the patterns in $\mathcal{P}_{1,2}(\mathbf{J})$, it is possible that entities (i) have multiple possible values for a single attribute; or (ii) have missing attribute values. Case (i) occurs when there are multiple patterns extracted for a single attribute, while case (ii) occurs when there are no non-conflicting patterns for this attribute. For either case, we randomly select an attribute value from the possible values.

Algorithm 2 Answer generation.

Inputs: Concept matrix $\mathbf{J} = [J_{1,1} \dots J_{3,2}]$.

for $(K, \check{\mathbf{J}}) \in \mathcal{P}_1^{(\text{all})}(\mathbf{J}) \cap \mathcal{P}_2^{(\text{all})}(\mathbf{J})$ **do** // *see* (1)
  if $[\check{J}_{3,1}, \check{J}_{3,2}]$ does not conflict with pattern K **then**
    Compute attribute value for $\check{J}_{3,3}$ using pattern K.
Collect all the above attribute values for $J_{3,3}$.
while $\nexists$ unique value for some attribute of an entity **do**
  Randomly choose one valid attribute value.
Generate ideal $J_{3,3} \subseteq R$.
return $J_{3,3}$ and the corresponding image.

4. Discussion

Algebraic machine reasoning provides a fundamentally new paradigm for machine reasoning beyond numerical computation. Abstract notions in reasoning tasks are encoded very concretely as ideals, which are computable algebraic objects. We treat ideals as "actual objects of study", and we do not require numerical values to be assigned to them. This means our framework is capable of reasoning on more qualitative or abstract notions that do not naturally have associated numerical values. Novel problem-solving, such as the discovery of new abstract patterns from observations, is realized concretely as computations on ideals (e.g. computing the primary decompositions of ideals). In particular, we are *not* solving a system of polynomial equations, in contrast to existing applications of algebra in AI (cf. Section 2). Variables (or primitive instances) are *not* assigned values. We do *not* evaluate polynomials at input values.

Theory-wise, our proposed approach breaks new ground. We established a new connection between machine reasoning and commutative algebra, two areas that were completely unrelated previously. There is over a century's worth of very deep results in commutative algebra that have not been tapped. Could algebraic methods be the key to tackling the long-standing fundamental questions in machine reasoning? It was only much more recently in 2014 that Léon Bottou [4] suggested that humans should "build reasoning capabilities from the ground up", and he speculated that the missing ingredient could be an algebraic approach.

Why use ideals to represent concepts? Why not use sets? Why not use symbolic expressions, e.g. polynomials? Intuitively, we think of a concept as an "umbrella term" consisting of multiple (potentially infinitely many) instances of the concept. Treating concepts as merely sets of instances is inadequate in capturing the expressiveness of human reasoning. A set-theoretic representation system with finitely many "primitive sets" can only have finitely many possible sets in total. In contrast, we proved that we can construct infinitely many concepts from only *finitely many* primitive concepts (Theorem 3.1). This agrees with our intuition that humans are able to express infinitely many concepts from only finitely many primitive concepts. The main reason is that the "richer" algebraic structure of ideals allows for significantly more operations on ideals, beyond set-theoretic operations. See Appendix A.4 for further discussion.

Why is our algebraic method fundamentally different from logic-based methods, e.g. those based on logic programming? At the heart of logic-based reasoning is the idea that reasoning can be realized concretely as the resolution (or inverse resolution) of logical expressions. Inherent in this idea is the notion of *satisfiability*; cf. [14]. Intuitively, we have a logical expression, usually expressed in a canonical normal form, and we want to assign truth values (true or false) to literals in the logical expression, so that the entire expression is satisfied (i.e. truth value is "true"); see Appendix C.1 for more discussion. In fact, much of the exciting progress in automated theorem proving [1, 13, 19, 21, 44, 55] is based on logic-based reasoning.

In contrast, algebraic machine reasoning builds upon computational algebra and computer algebra systems. At the heart of our algebraic approach is the idea that reasoning can be realized concretely as solving computational problems in algebra. Crucially, there is no notion of satisfiability. We do not assign truth values (or numerical values) to concepts in $R = \Bbbk[x_1, \dots, x_n]$. In particular, although primitive concepts $\langle x_1 \rangle, \dots, \langle x_n \rangle$ in R correspond to the variables $x_1, \dots, x_n$, we do not assign values to primitive concepts. Instead, ideals are treated as the "actual objects of study", and we reduce "solving a reasoning task" to "solving non-numerical computational problems involving ideals". Moreoever, our framework can discover new patterns beyond the actual rules of the RPM task; see Section 5.2.

In the RPM task, we have attribute concepts representing "position", "number", "type", "size", and "color"; these are concepts that categorize the primitive instances according

	Method	Avg. Acc.	Center	2×2G	3×3G	O-IC	O-IG	L-R	U-D
1	LSTM [46]	18.9 / 13.1	26.2 / 13.2	16.7 / 14.1	15.1 / 13.7	21.9 / 12.2	21.1 / 13.0	14.6 / 12.8	16.5 / 12.4
2	WReN [32]	23.8 / 34.0	29.4 / 58.4	26.8 / 38.9	23.5 / 37.7	22.5 / 38.8	21.5 / 22.6	21.9 / 21.6	21.4 / 19.7
3	ResNet [46]	40.3 / 53.4	44.7 / 52.8	29.3 / 41.9	27.9 / 44.3	46.2 / 63.2	35.8 / 53.1	51.2 / 58.8	47.4 / 60.2
4	ResNet+DRT [46]	40.4 / 59.6	46.5 / 58.1	28.8 / 46.5	27.3 / 50.4	46.0 / 69.1	34.2 / 60.1	50.1 / 65.8	49.8 / 67.1
5	LEN [51]	41.4 / 72.9	56.4 / 80.2	31.7 / 57.5	29.7 / 62.1	52.1 / 84.4	31.7 / 71.5	44.2 / 73.5	44.2 / 81.2
6	CoPINet [47]	46.1 / 91.4	54.4 / 95.1	36.8 / 77.5	31.9 / 78.9	52.2 / 98.5	42.8 / 91.4	51.9 / 99.1	52.5 / 99.7
7	DCNet [54]	49.4 / **93.6**	57.8 / 97.8	34.1 / 81.7	35.5 / 86.7	57.0 / **99.0**	42.9 / **91.5**	58.5 / **99.8**	60.0 / **99.8**
8	NCD [52]	48.2 / 37.0	60.0 / 45.5	31.2 / 35.5	30.0 / 39.5	62.4 / 40.3	39.0 / 30.0	58.9 / 34.9	57.2 / 33.4
9	SRAN [12]	60.8 / -	78.2 / -	50.1 / -	42.4 / -	68.2 / -	46.3 / -	70.1 / -	70.3 / -
10	PrAE [48]	77.0 / 65.0	90.5 / 76.5	85.4 / 78.6	45.6 / 28.6	63.5 / 48.1	60.7 / 42.6	96.3 / 90.1	97.4 / 90.9
11	Our Method	**93.2** / 92.9	**99.5** / **98.8**	**89.6** / **91.9**	**89.7** / **93.1**	**99.6** / 98.2	**74.7** / 70.1	**99.7** / 99.2	**99.5** / 99.1
	Human [46]	- / 84.4	- / 95.5	- / 81.8	- / 79.6	- / 86.4	- / 81.8	- / 86.4	- / 81.8

Table 1. Performance on I-RAVEN/RAVEN. We report mean accuracy, and the accuracies for all configurations: `Center`, `2x2Grid`, `3x3Grid`, `Out-InCenter`, `Out-InGrid`, `Left-Right`, and `Up-Down`.

to their semantics, into what humans would call attributes. Intuitively, an attribute concept combines certain primitive concepts together in a manner that is "meaningful" to the task. For example, $\langle x_{\text{white}}, x_{\text{gray}}, x_{\text{black}} \rangle$ is "more meaningful" than $\langle x_{\text{white}}, x_{\text{circle}}, x_{\text{large}} \rangle$ as a "simpler" or "generalized" concept, since we would treat $x_{\text{white}}, x_{\text{gray}}, x_{\text{black}}$ as instances of a single broader "color" concept.

Notice that the primitive concepts correspond precisely to the prediction classes of our object detection models. Such prediction classes are already implicitly identified by the available data. Consequently, our method is limited by what our perception modules can perceive. For other tasks, e.g. where text data is available, entity extraction methods can be used to identify primitive concepts. Note also that our method requires prior knowledge, since there is no training step for the reasoning module. This limitation can be mitigated if we replace user-defined functions on concepts with trainable functions optimized via deep learning.

In general, the identification of attribute concepts is task-specific, and the resulting reasoning performance would depend heavily on these identified attribute concepts. Effectively, our choice of attribute concepts would determine the *inductive bias* of our reasoning framework: As we decompose a concept J into "simpler" concepts (i.e. primary components in $\text{pd}(J)$), only those "simpler" concepts contained in attribute concepts are deemed "meaningful". Concretely, let $J, J' \subsetneq R$ be concepts such that $\text{pd}(J) = \{J_1, \ldots, J_k\}$ and $\text{pd}(J') = \{J'_1, \ldots, J'_\ell\}$, i.e. J, J' have minimal primary decompositions $J = J_1 \cap \cdots \cap J_k$ and $J' = J'_1 \cap \cdots \cap J'_\ell$, respectively. We can examine their primary components and extract out those primary components (between the two primary decompositions) that are contained in some common attribute concept. For example, if A is an attribute concept of R such that $J_1 \subseteq A$ and $J'_1 \subseteq A$, then J and J' share a "common pattern", represented by the attribute concept A.

5. Experiment results

To show the effectiveness of our framework, we conducted experiments on the RAVEN [46] and I-RAVEN datasets. In both datasets, RPMs are generated according to 7 configurations. We trained our perception modules on 4200 images from I-RAVEN [12] (600 from each configuration), and used them to predict attribute values of entities. The average accuracy of our perception modules is 96.24%. For both datasets, we tested on 2000 instances for each configuration. Overall, our reasoning framework is fast (7 hours for 14000 instances on a 16-core Gen11 Intel i7 CPU processor). See Appendix B for full experiment details.

5.1. Comparison with other baselines

Table 1 compares the performance of our method with 10 other baseline methods. We use the accuracies on I-RAVEN reported in [12, 52] for methods 1-7, and the accuracies on RAVEN reported in [46, 52] for methods 1-5. All the other accuracies are obtained from the original papers. As a reference, we also include the human performance on the RAVEN dataset (i.e. *not* I-RAVEN) as reported in [46].

5.2. Ambiguous instances and new patterns

Although our method outperforms all baselines, some instances have multiple answer options that are assigned equal top scores by our framework. Most of these cases occur due to the discovery of (i) "accidental" unintended rules (e.g. Fig. 3); or (ii) new patterns beyond the actual rules in the dataset (e.g. Fig. 4). Case (i) occurs because in the design of I-RAVEN, at most one rule is assigned to each attribute.

Interestingly, case (ii) reveals that our framework is able to discover completely new patterns that are not originally designed as rules for I-RAVEN. In Fig. 4, the new pattern discovered is arguably very natural to humans.

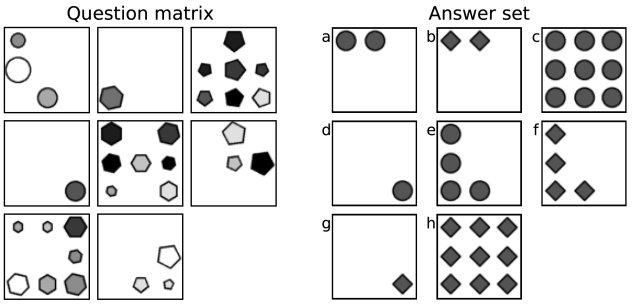

Figure 3. An example of an ambiguous RPM instance. The given answer is option **g**. For I-RAVEN, the type sequence ("circle", "hexagon", "pentagon") in the first two rows follows a Progression rule with consecutively decreasing type indices, so **g** could be a correct answer. (Remaining attribute values are determined by other patterns.) However, our framework assigns equal top scores to both options **d** and **g**, as a result of another inter-invariance pattern for type (the type set {"circle", "hexagon", "pentagon"} is invariant across the rows). Thus, option **d** could also be correct.

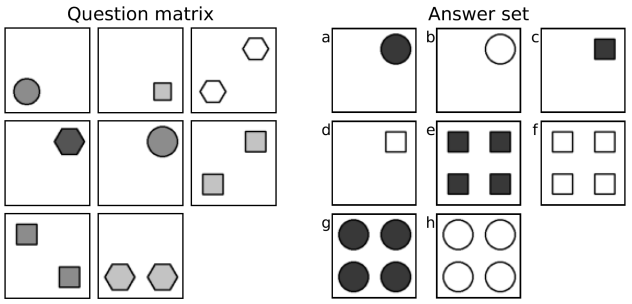

Figure 4. An example of an RPM instance with an unexpected new pattern. The given answer is option **h**. In each row, the number of entities in the first 2 panels sum up to the number of entities in the 3rd panel, so **h** could be correct. However, our framework assigns equal top scores to both options **b** and **h**, as a result of a new inter-invariance pattern for number (informally, every panel has either 1 or 2 entities). Thus option **b** could also be correct.

5.3. Evaluation of answer generation

Every RPM instance is assumed to have a single correct answer from the given answer set. However, there are multiple other possible images that are also acceptable as correct answers. For example, images modified from the given correct answer, via random perturbations of those attributes that are not involved in any of the rules (e.g. entity angles in the I-RAVEN dataset), are also correct. All these distinct correct answers (images) can be encoded algebraically as the same concept, based on prior knowledge of which raw perceptual attributes are relevant for the RPM task. Hence, to evaluate the answer generation process proposed in Section 3.4.2, we will directly evaluate the generated concepts.

Let $J = \langle e_1, \ldots, e_k \rangle$ and $J' = \langle e'_1, \ldots, e'_\ell \rangle$ be concepts representing the ground truth answer and our generated answer, respectively. Here, each e_i (or e'_i) is a monomial of the form $x_i^{(\text{pos})} x_i^{(\text{type})} x_i^{(\text{color})} x_i^{(\text{size})}$, and represents an entity described by 4 attributes. Motivated by the well-known idea of Intersection over Union (IoU), we propose a new similarity measure between J and J'. In order to define analogous notions of "intersection" and "union", we first pair e_i with e'_j if $x_i^{(\text{pos})} = x_j'^{(\text{pos})}$ (i.e. same "position" values). This pairing is well-defined, since the "position" values of the entities in any panel are uniquely determined. Hence we can group all entities in J and J' into 3 sets:

$$S_1 := \{(e_i, e'_j) \mid e_i \in J, e'_j \in J', x_i^{(\text{pos})} = x_j'^{(\text{pos})}\};$$
$$S_2 := \{e_i \in J \mid \nexists e'_j \in J' \text{ such that } (e_i, e'_j) \in S_1\};$$
$$S_3 := \{e'_j \in J' \mid \nexists e_i \in J \text{ such that } (e_i, e'_j) \in S_1\}.$$

We can interpret S_1 and $S_1 \cup S_2 \cup S_3$ as analogous notions of the "intersection" and "union" of J and J', respectively. Thus, we define our similarity measure as follows:

$$\varphi(J, J') := \frac{\sum_{(e_i, e'_j) \in S_1} \phi(e_i, e'_j)}{|S_1| + |S_2| + |S_3|}; \tag{2}$$

$$\phi(e_i, e'_j) := \frac{1}{4} \sum_a \mathbb{1}(x_i^{(\text{a})} = x_j'^{(\text{a})}); \tag{3}$$

where in (3), a ranges over the 4 attributes in {pos, type, color, size}. Here, $\phi(e_i, e'_j)$ is the similarity score between e_i and e'_j, measured by the proportion of common variables.

The overall average similarity score of the generated answers is 67.7%. Note that within a panel, some attribute values such as "size", "color" and "position", may be totally random for `2x2Grid`, `3x3Grid`, `Out-InGrid` (e.g. as shown in Fig. 3). Hence, achieving high similarity scores for such cases would inherently require task-specific optimization and knowledge of how the data is generated. We assume neither. This could explain why our overall similarity score is lower than our answer selection accuracy.

For examples of generated images, see Appendix B.5.

6. Conclusion

Algebraic machine reasoning is a reasoning framework that is well-suited for abstract reasoning. In its current form, we have used primary decompositions as a key algebraic operation to discover abstract patterns in the RPM task, via the invariance modules that we have specially designed to mimic human reasoning. The idea that "discovering common patterns" can be realized concretely as "computing primary decompositions" is rather broad, and could potentially be applied to other inferential reasoning tasks.

More generally, our algebraic approach opens up new possibilities of tapping into the vast literature of commutative algebra and computational algebra. There are numerous algebraic operations on ideals (ideal quotients, radicals, saturation, etc.) and algebraic invariants (depth, height, etc.) that have not been explored in machine reasoning (or even in AI). Can we use them to tackle other reasoning tasks?

References

[1] Ibrahim Abdelaziz, Maxwell Crouse, Bassem Makni, Vernon Austel, Cristina Cornelio, Shajith Ikbal, Pavan Kapanipathi, Ndivhuwo Makondo, Kavitha Srinivas, Michael Witbrock, et al. Learning to guide a saturation-based theorem prover. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 6

[2] Dave Bayer, André Galligo, and Mike Stillman. Gröbner bases and extension of scalars. In *Computational algebraic geometry and commutative algebra (Cortona, 1991)*, Sympos. Math., XXXIV, pages 198–215. Cambridge Univ. Press, Cambridge, 1993. 3

[3] Yaniv Benny, Niv Pekar, and Lior Wolf. Scale-localized abstract reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12557–12565, 2021. 2

[4] Léon Bottou. From machine learning to machine reasoning. *Machine learning*, 94(2):133–149, 2014. 6

[5] Patricia A Carpenter, Marcel A Just, and Peter Shell. What one intelligence test measures: a theoretical account of the processing in the Raven progressive matrices test. *Psychological review*, 97(3):404, 1990. 1

[6] John B Carroll et al. *Human cognitive abilities: A survey of factor-analytic studies*. Number 1. Cambridge University Press, 1993. 1

[7] Kai Fong Ernest Chong. A closer look at the approximation capabilities of neural networks. In *International Conference on Learning Representations*, 2020. 2

[8] David A Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer, 2015. 2

[9] Ian Davidson and Peter B. Walker. Towards fluid machine intelligence: Can we make a gifted AI? *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9760–9764, 2019. 1

[10] Mathias Drton, Bernd Sturmfels, and Seth Sullivant. *Lectures on algebraic statistics*, volume 39. Springer Science & Business Media, 2008. 2

[11] Eliana Duarte, Orlando Marigliano, and Bernd Sturmfels. Discrete statistical models with rational maximum likelihood estimator. *Bernoulli*, 27(1):135–154, 2021. 2

[12] Sheng Hu, Yuqing Ma, Xianglong Liu, Yanlu Wei, and Shihao Bai. Stratified rule-aware network for abstract visual reasoning. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2021. 2, 7

[13] Geoffrey Irving, Christian Szegedy, Alexander A Alemi, Niklas Eén, François Chollet, and Josef Urban. Deepmath-deep sequence models for premise selection. In *Advances in Neural Information Processing Systems*, pages 2235–2243, 2016. 6

[14] Joxan Jaffar and J-L Lassez. Constraint logic programming. In *Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 111–119, 1987. 6

[15] Marius Jahrens and Thomas Martinetz. Solving Raven's progressive matrices with multi-layer relation networks. *arXiv preprint arXiv:2003.11608*, 2020. 1, 2

[16] Joa Kileel, Matthew Trager, and Joan Bruna. On the expressive power of deep polynomial neural networks. *Advances in neural information processing systems*, 2019. 2

[17] Youngsung Kim, Jinwoo Shin, Eunho Yang, and Sung Ju Hwang. Few-shot visual reasoning with meta-analogical contrastive learning. *Neurips 2020*, 2020. 1

[18] Franz J Király, Paul Von Bünau, Frank C Meinecke, Duncan AJ Blythe, Klaus-Robert Müller, and Kenji Fukumizu. Algebraic geometric comparison of probability distributions. *Journal of Machine Learning Research*, 13(3), 2012. 2

[19] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*, 2020. 6

[20] Paul Larsen and Franz Király. Fano schemes of generic intersections and machine learning. *International Journal of Algebra and Computation*, 24(07):923–933, 2014. 2

[21] Wenda Li, Lei Yu, Yuhuai Wu, and Lawrence C Paulson. Isarstep: a benchmark for high-level mathematical reasoning. In *International Conference on Learning Representations*, 2021. 6

[22] Shaowei Lin. *Algebraic methods for evaluating integrals in Bayesian statistics*. PhD thesis, UC Berkeley, 2011. 2

[23] Muyang Lyu, Ruixuan Liu, and Junyi Wang. Solving raven's progressive matrices using rnn reasoning network. In *2022 7th International Conference on Computational Intelligence and Applications (ICCIA)*, pages 32–37. IEEE, 2022. 2

[24] Petros Maragos, Vasileios Charisopoulos, and Emmanouil Theodosis. Tropical geometry and machine learning. *Proceedings of the IEEE*, 2021. 2

[25] Melanie Mitchell. Abstraction and analogy-making in artificial intelligence. *arXiv preprint arXiv:2102.10717*, 2021. To appear in Annals of the New York Academy of Sciences. 1

[26] Melanie Mitchell. Why AI is harder than we think. *arXiv preprint arXiv:2104.12871*, 2021. 1

[27] Niv Pekar, Yaniv Benny, and Lior Wolf. Generating correct answers for progressive matrices intelligence tests. *Advances in Neural Information Processing Systems*, 33:7390–7400, 2020. 2

[28] Patrick Perret. Children's inductive reasoning: Developmental and educational perspectives. *Journal of Cognitive Education and Psychology*, 14(3):389–408, 2015. 1

[29] Sylvain Petitjean. Algebraic geometry and object representation in computer vision. In Martial Hebert, Jean Ponce, Terry Boult, and Ari Gross, editors, *Object Representation in Computer Vision*, pages 155–165, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. 2

[30] Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. RNNLogic: Learning logic rules for reasoning on knowledge graphs. In *International Conference on Learning Representations*, 2021. 1

[31] Markus Norman Rabe, Dennis Lee, Kshitij Bansal, and Christian Szegedy. Mathematical reasoning via self-supervised skip-tree training. In *International Conference on Learning Representations*, 2021. 1

[32] Adam Santoro, Felix Hill, David Barrett, Ari Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In *International Conference on Machine Learning*, pages 4477–4486, 2018. 1, 2, 7

[33] Snejana Shegheva and Ashok Goel. The structural affinity method for solving the Raven's Progressive Matrices test for intelligence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 1

[34] Fan Shi, Bin Li, and Xiangyang Xue. Raven's progressive matrices completion with latent gaussian process priors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9612–9620, 2021. 2

[35] Jürgen Stuber. Superposition theorem proving for commutative rings. In *Automated Deduction—A Basis for Applications*, pages 31–55. Springer, 1998. 2

[36] Seth Sullivant. Algebraic geometry of gaussian bayesian networks. *Advances in Applied Mathematics*, 40(4):482–513, 2008. 2

[37] Sjoerd Van Steenkiste, Francesco Locatello, Jürgen Schmidhuber, and Olivier Bachem. Are disentangled representations helpful for abstract visual reasoning? In *Advances in Neural Information Processing Systems*, pages 14245–14258, 2019. 1

[38] Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pages 6545–6554. PMLR, 2019. 1

[39] Sumio Watanabe. Algebraic analysis for nonidentifiable learning machines. *Neural Comput.*, 13(4):899–933, apr 2001. 2

[40] Sumio Watanabe. Algebraic geometrical methods for hierarchical learning machines. *Neural Netw.*, 14(8):1049–1060, oct 2001. 2

[41] Sumio Watanabe. *Algebraic geometry and statistical learning theory*. Number 25. Cambridge university press, 2009. 2

[42] Sumio Watanabe. A widely applicable bayesian information criterion. *Journal of Machine Learning Research*, 14(3):867–897, 2013. 2

[43] Sumio Watanabe and Manfred Opper. Asymptotic equivalence of bayes cross validation and widely applicable information criterion in singular learning theory. *Journal of machine learning research*, 11(12), 2010. 2

[44] Yuhuai Wu, Albert Jiang, Jimmy Ba, and Roger Baker Grosse. Int: An inequality benchmark for evaluating generalization in theorem proving. In *International Conference on Learning Representations*, 2021. 6

[45] Keisuke Yamazaki and Sumio Watanabe. Singularities in mixture models and upper bounds of stochastic complexity. *Neural Networks*, 16(7):1029–1038, 2003. 2

[46] Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. RAVEN: A dataset for relational and analogical visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5317–5327, 2019. 1, 2, 7

[47] Chi Zhang, Baoxiong Jia, Feng Gao, Yixin Zhu, Hongjing Lu, and Song-Chun Zhu. Learning perceptual inference by contrasting. In *Advances in Neural Information Processing Systems*, pages 1075–1087, 2019. 1, 2, 7

[48] Chi Zhang, Baoxiong Jia, Song-Chun Zhu, and Yixin Zhu. Abstract spatial-temporal reasoning via probabilistic abduction and execution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 7

[49] Chi Zhang, Sirui Xie, Baoxiong Jia, Ying Nian Wu, Song-Chun Zhu, and Yixin Zhu. Learning algebraic representation for systematic generalization in abstract reasoning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIX*, pages 692–709. Springer, 2022. 2

[50] Liwen Zhang, Gregory Naitzat, and Lek-Heng Lim. Tropical geometry of deep neural networks. In *International Conference on Machine Learning*, pages 5824–5832. PMLR, 2018. 2

[51] Kecheng Zheng, Zheng-Jun Zha, and Wei Wei. Abstract reasoning with distracting features. In *Advances in Neural Information Processing Systems*, volume 32, 2019. 2, 7

[52] Tao Zhuo, Qiang Huang, and Mohan Kankanhalli. Unsupervised abstract reasoning for raven's problem matrices. *IEEE Transactions on Image Processing*, 30:8332–8341, 2021. 2, 7

[53] Tao Zhuo and Mohan Kankanhalli. Solving raven's progressive matrices with neural networks. *arXiv preprint arXiv:2002.01646*, 2020. 2

[54] Tao Zhuo and Mohan Kankanhalli. Effective abstract reasoning with dual-contrast network. In *International Conference on Learning Representations*, 2021. 2, 7

[55] Zsolt Zombori, Josef Urban, and Chad E Brown. Prolog technology reinforcement learning prover. In *International Joint Conference on Automated Reasoning*, pages 489–507. Springer, 2020. 6