# Constructive solving of Raven's IQ tests with analogical proportions

William Correa Beltran, Henri Prade, Gilles Richard

# Constructive Solving of Raven's IQ Tests with Analogical Proportions

William Correa Beltran,[1,*] Henri Prade,[2,†] Gilles Richard[2,‡]
[1] *IRISA ENSSAT, Lannion France*
[2] *IRIT, University of Toulouse, Toulouse, France*

The paper shows that a Boolean logic modeling of analogical proportions can serve as a basis for solving quizzes as well as a common and popular type of IQ tests, namely Raven's progressive matrices. They are nonverbal tests supposedly measuring general intelligence. A $3 \times 3$ Raven matrix exhibits eight geometric pictures displayed as its eight first cells: the remaining ninth cell is empty. In these tests, a set of candidate pictures is also given among which the subject is asked to identify the solution. In this paper, we investigate a general approach allowing to automatically solve Raven's progressive matrices tests. The approach is based on a logical view of analogical proportions, i.e., statements of the form "*A* is to *B* as *C* is to *D*." We assume that analogical proportions hold between the rows and between the columns of the Raven's matrix. This view can be applied to a feature-based description of the pictures but also, in a number of cases, to a very low level representation, i.e., the pixel level. It appears that the analogical proportion reading just amounts here to a recopy of patterns of feature values that already appear in the data, after checking that there is no conflicting patterns. Implementing this principle, our algorithm builds up the ninth picture, without the help of any set of candidate solutions, and only on the basis of the eight known cells of the Raven matrices. A comparison with other approaches is provided. The ability to construct the missing picture without relying on candidate solutions is a distinctive feature of our work. Moreover, we emphasize the general principle underlying the approach that offers a simple and uniform mechanism applicable to the tests. At this step, the paper makes no claim about the cognitive validity of the approach with respect to the way humans solve such tests.

## 1. INTRODUCTION

The human ability "*to see a particular object or situation in one context as being the same as another object or situation in another context*"[1] is the main process at work when making analogy, and this ability is one of the main features of human intelligence. One of the classical approaches to intelligence evaluation is

*Author to whom all correspondence should be addressed: e-mail: william.correa_beltran@irisa.fr.

†e-mail: prade@irit.fr.
‡e-mail: richard@irit.fr.

DOI 10.1002/int.21817

to apply psychometric tests supposed to measure the intelligence quotient (namely IQ tests). Very early, these IQ tests have been considered as offering challenging problems for AI systems. These IQ tests often involve analogical reasoning tasks.

In the last past years, a logical view of analogical proportions has been proposed,[2–5] leading to infer plausible conclusions on the basis of existing analogical proportions. In this process, items are described as vectors of Boolean features and we look for proportions between items that hold componentwise. This view allows to design analogy-based classifiers where each example is described as a Boolean vector and associated with a label. A new vector is then classified on the basis of triples of already classified examples such that a proportion holds on a maximum number of features. Then the class of the new vector is determined by solving an analogical proportion equation. These analogical classifiers have shown promising results[6,7] and are able to often outperform the results obtained with the $k$-nearest neighbors method.

Besides, the potential of this logical view of proportions for solving IQ quizzes including an instance of Raven's tests,[8] has been first advocated by Prade and Richard.[9] Raven tests, which are widely used in practice, take the form of a series of instances having the format of a $n \times n$ matrix (where $n$ is 2, 3, or 4) whose cells contain diverse geometric figures, except the last cell which is empty and has to be completed by selecting a solution among eight candidates (when $n = 3$). The series of matrices in a test are of increasing difficulty, hence the name "Raven's progressive matrices," abbreviated as RPM. In that case, the application of analogical proportions is not as straightforward as it is with simple quizzes where a sequence of three figures has to be completed by a fourth one. Until now, two computerized approaches[10,11] have been proposed that are able to solve a large number of Raven tests. An approach proposed by Carpenter et al.[10] exploit human-originated rules that describe the different types of situations encountered (without reference to analogy). Lovett et al.[11] described an analogy-based method which leads to an abstract categorization of the possible instances of Raven tests, reducing the problem at hand to a small number of classes of situations.

Our aim in this paper is to show that it is possible to solve almost all Raven tests with a uniform approach, based on the analogical proportion view. The approaches of Carpenter et al.[10] and Lovett et al.[11] estimate the plausibility of each candidate solution and choose the most plausible one. In that respect, this is similar to the first AI program by Th. Evans[12] in the 1960s for solving simple geometric quizzes, which also considers the different candidate solutions. In our method, the candidate solutions are not supposed to be given, which departs from the previous works.

Our method is motivated by the results obtained by analogical proportion-based classifiers. The approaches like those of Carpenter et al.[10] and Lovett et al.[11] are based on models whose cognitive validity has been recognized. Despite its good performance with respect to Raven's tests, we do not make any claim regarding the cognitive value of our model. Moreover, the feature extraction step is supposed to be done by an external system. Even if the extraction of simple features from pictures may often be easy and could be realized by a computer program, identifying all the relevant ones (in particular, higher order features which may be required in difficult tests) remains a challenging part of the test.

The paper is organized in the following way. First, a brief background on the logical view of analogical proportions and its origins is provided emphasizing their predictive power. Then the proposed approach is detailed and illustrated on different Raven matrix problems. It is shown that some instances of Raven problems can be solved with a low-level representation in terms of pixels, whereas others can be only solved on the basis of a feature-based representation. It is worth pointing out that the approach computes the content of the last cell in Raven matrices, rather than selecting it among a set of candidate solutions. Interestingly enough, it is in fact, a particular view of analogical proportions, presented in this paper, which is at work here. It mainly amounts to *recopying* observed changes or absence of changes in a controlled manner. The paper ends with a state of the art and a comparative discussion of the related works, after having analyzed the few cases where the approach (partially) fails. The paper is a thoroughly revised and substantially expanded version of two conference papers.[13,14]

## 2. ANALOGICAL PROPORTIONS: A REVIEW

To investigate analogical proportions, a relevant starting point is to consider numerical proportions linking four numbers $a, b, c, d$. We have two kinds of numerical proportion: the geometric one asserting an equality of ratios as $\frac{a}{b} = \frac{c}{d}$, and the arithmetic one asserting an equality of differences as $a - b = c - d$. In the case of geometric proportion, the ratio $\frac{a}{b}$ could be considered as a compact representation of the dissimilarities between $a$ and $b$. In the case of arithmetic proportion, the representation is more straightforward. If we interpret a ratio $\frac{a}{b}$ (resp. a difference $a - b$) as the way $a$ differs from $b$, since $\frac{a}{b} = \frac{c}{d}$ (resp. $a - b = c - d$) is equivalent to $\frac{b}{a} = \frac{d}{c}$ (resp. $b - a = d - c$); in both cases, these numerical proportions tell us that "$a$ differs from $b$ as $c$ differs from $d$" and conversely.

### 2.1. Basic Definitions

When we consider Boolean values instead of numbers, a simple way to abstract these notions has been given by Prade and Richard,[4] where similarity and dissimilarity indicators are defined for a Boolean pair $(a, b)$ as

- $a \wedge b$ and $\overline{a} \wedge \overline{b}$ are the *similarity indicators*.
- $a \wedge \overline{b}$ and $\overline{a} \wedge b$ are the *dissimilarity indicators*.

Indeed, $a$ and $b$ are supposed to be the values of a given binary feature for two distinct items. Then, $a \wedge b$ holds if the two items have this feature in common, $\overline{a} \wedge \overline{b}$ holds if none of the items has this feature, $a \wedge \overline{b}$ holds if the first item has the feature alone, and vice versa for $\overline{a} \wedge b$.

Generally speaking, a *logical proportion*[15] is a conjunction of two equivalences between such indicators. Among the 120 distinct logical proportions that exist, the

| $a$ | $b$ | $c$ | $d$ | Truth value |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |

**Figure 1.** Analogical proportion truth table.

analogical proportion, denoted $a : b :: c : d$, is defined with

$$a : b :: c : d =_{def} (a \wedge \overline{b} \equiv c \wedge \overline{d}) \wedge (\overline{a} \wedge b \equiv \overline{c} \wedge d) \quad (1)$$

(i.e., the proportion holds when this Boolean formula holds). Thus, a logical proportion is modeled by means of a quaternary connective. This definition can be seen as the logical counterpart of "$a$ differs from $b$ as $c$ differs from $d$," and vice versa.[2,3] This is the expected meaning of "$a$ is to $b$ as $c$ is to $d$". In the Boolean setting, we need two equivalences instead of one equality in the case of numerical proportions: This is due to the fact that $(a \wedge \overline{b} \equiv c \wedge \overline{d})$ does not imply $(\overline{a} \wedge b \equiv \overline{c} \wedge d)$. Using two equivalences ensures a proper "Boolean translation" of numerical proportions in the following sense: $a : b :: c : d$ implies $a : c :: b : d$, which would not be the case with only one equivalence. Other noticeable properties of numerical proportions have their Boolean counterpart still valid, as we are going to see. As a Boolean formula, analogical proportions can be viewed through its truth table that we display in Figure 1 (where the lines leading to false are discarded). As we can see, 0110 and 1001 do not satisfy an analogical proportion. It can be easily checked from definition (1) that the basic properties of a numerical proportion still hold in the Boolean setting:

- symmetry: $a : b :: c : d \Rightarrow c : d :: a : b$,
- central permutation: $a : b :: c : d \Rightarrow a : c :: b : d$, and
- transitivity: $(a : b :: c : d) \wedge (c : d :: e : f) \Rightarrow (a : b :: e : f)$.

From symmetry and central permutation, we get $a : b :: c : d \Rightarrow b : a :: d : c$, which is still a property of numerical proportions (see the work by Prade and Richard[15] for other properties).

One of the side product of the numerical proportion is the well known "rule of three," where it is stated that, when one number $x$ is missing for a numerical proportion $a : b :: c : x$ to hold, there is a way to compute/predict this missing number $x$ as being $\frac{c \times b}{a}$. In the Boolean context, depending on $a$, $b$, $c$, the analogical equation $a : b :: c : x = 1$ is not always solvable. Indeed, $0 : 1 :: 1 : x$ or $1 : 0 :: 0 : x$ have no solution. A direct examination of the truth table gives the answer to this equation and can be formalized as follows: The analogical equation $a : b :: c : x$ is solvable iff $((a \equiv b) \vee (a \equiv c))$. In that case, the unique solution $x$ is $a \equiv (b \equiv c)$.[2]

In light of the equation $a : b :: c : x = 1$, we have to observe that the above truth table can be split into two parts:

– a part with the four lines 0101, 1010, 1111, and 0000, where it is clear that the first pair $(a, b)$ and the second one $(c, d)$ are identical, then the third item is equal to the first one (i.e., $a \equiv c$). Then, the equation-solving process reduces to a (re)copy of the second item into the fourth position. The logical formula corresponding to a truth table with these four lines is $(a \equiv c) \wedge (b \equiv d)$. As we shall see, these four lines constitute the effective patterns at work for solving Raven's tests using only a recopy scheme.

– and a part with the remaining lines 0011 and 1100 : In this case, the proportion encodes that the two pairs $(a, b)$ and $(c, d)$ (which are distinct) refer to different contexts, and solving the equation $0 : 0 :: 1 : x$ does not correspond to having $(c, d)$ as a copy of $(a, b)$. Note however that the patterns 0011 and 1100 should belong to the truth table of the analogical proportion as being derivable from 0101 and 1010 by central permutation.

### 2.2. Equivalent Definitions for an Analogical Proportion

The initial definition of analogical proportion as

$$(a \wedge \overline{b} \equiv c \wedge \overline{d}) \wedge (\overline{a} \wedge b \equiv \overline{c} \wedge d)$$

is not the only one which could be considered. In fact there are other logically equivalent expressions,[3,4] and one of them is particularly worth mentioning. In a Boolean setting, it can be checked that the previous definition is strictly equivalent to the following one which does not involve any negation operator:

$$(a \wedge d \equiv b \wedge c) \wedge (a \vee d \equiv b \vee c) \tag{2}$$

This definition of $a : b : c : d$ asserts equivalences between a combination of the mean terms $b$ and $c$ and the same combination of the extreme terms $a$ and $d$. Interestingly enough, such an expression was considered and named "logical proportion" by Piaget,[16] but without, apparently, any reference to the idea of analogy. The truth table of $a : b :: c : d$ is then easily deducible from this definition: for instance, the fact that $1 : 0 :: 0 : 1$ is not a valid analogical proportion is obvious with respect to definition (2). When thinking in terms of Boolean lattice with $\wedge$ being the least upper bound $lub$, $\vee$ the greatest lower bound $glb$, and $\equiv$ the equality, the formula asserts that the pairs $(a, d)$ and $(b, c)$ should have the same $lub$ and $glb$.

In some sense, this formula establishes a strong link with the usual numerical proportion $\frac{a}{b} = \frac{c}{d}$ which is simply defined as $a \times d = b \times c$.[a] Defining analogical proportions on general lattices (not only Boolean lattices) has been done by Hesse,[17]

[a] When considering the set of natural numbers $\mathbb{N}$, partially ordered by divisibility, where the $lub$ is the least common multiple and the $glb$ is the greatest common divisor, the definition of numerical proportion $\frac{a}{b} = \frac{c}{d}$ as $a \times d = b \times c$ is more general than having $lub(a, d) = lub(b, c) \wedge glb(a, d) = glb(b, c)$: for instance $\frac{5}{6} = \frac{10}{12}$ but $glb(5, 12) = 1 \neq glb(6, 10) = 2$.

| | | | | | |
|---|---|---|---|---|---|
| $\overrightarrow{a}$ | 1 | 0 | 1 | 0 | 1 |
| $\overrightarrow{b}$ | 0 | 0 | 1 | 0 | 1 |
| $\overrightarrow{c}$ | 1 | 1 | 0 | 0 | 1 |
| $\overrightarrow{d}$ | 0 | 1 | 0 | 0 | 1 |

**Figure 2.** An example of proportion with Boolean vectors in $\mathbb{B}$.[5]

Stroppa and Yvon,[18,19] but it is out of the scope of this paper to investigate this view here.

### 2.3. Extension to Boolean Vectors

In practice, items are often described in terms of *several* Boolean features, and the idea of analogical proportion has then to be extended to Boolean vectors: We just consider that a proportion holds on Boolean vectors of dimension $n$ (i.e., belonging to $\mathbb{B}^n$), namely, $\overrightarrow{a}$, $\overrightarrow{b}$, $\overrightarrow{c}$, $\overrightarrow{d}$ iff it holds componentwise. This can be formally stated as

$$\overrightarrow{a} : \overrightarrow{b} :: \overrightarrow{c} : \overrightarrow{d} \text{ iff } \forall i \in [1, n], a_i : b_i :: c_i : d_i$$

For instance, when $\overrightarrow{a} = (1, 0, 1, 0, 1)$, $\overrightarrow{b} = (0, 0, 1, 0, 1)$, $\overrightarrow{c} = (1, 1, 0, 0, 1)$, $\overrightarrow{d} = (0, 1, 0, 0, 1)$, they build an analogical proportion, which is better viewed as the matrix in Figure 2.

An important point to be noticed is that we can now build proportions where all the involved items are different: This was not the case with the Boolean definition (i.e., with $\mathbb{B}$) just because $\mathbb{B}$ has exactly two elements. It is immediate to check that this definition on $\mathbb{B}^n$ still satisfies the requirement of an analogical proportion, namely, symmetry and central permutation. This is a simple way to consider a proportion as establishing a link between objects described by multiple features, but being still of the same type. When dealing with Raven's matrices, it appears that we need more than that. That is why, in the next subsection, we investigate a way to define analogical proportions between objects of different types.

### 2.4. Cross-Domain Analogical Proportions

In the above definition of an analogical proportion, $a$, $b$, $c$, and $d$ belong to the same domain, namely, the set of truth values $\{0, 1\}$. We may need a view of analogy where $a$, $b$, $c$, and $d$ do not belong to a unique universe. An easy way to do this is to state that an analogical proportion holds between $a$, $b$, $c$, and $d$ as soon as there exists a relation $R$ such that

$$R(a, b) = R(c, d)$$

Under this view, $R$ should practically be nontrivial (i.e., not every pair satisfies $R$), but it should also be as simple as possible. Moreover, to have a useful analogical proportion, $R$ needs to be relevant for the user's tasks. Indeed, defining $R$ just as the two pairs $R = \{(a, b), (c, d)\}$ is certainly not enough, while recognizing that $a$ and $b$ on the one hand and $c$ and $d$ on another hand are linked by the same structural equation may be useful (Hesse[20] provided us with an interesting discussion).

We see that the condition about the type of $a, b, c, d$ is relaxed as this definition only forces $a$ and $c$, respectively, $b$ and $d$, to belong to the same domain, respectively. Thus we may have two distinct domains.

A particular instance of the above definition is when $R$ is based on a function $f$ which leads to the definition:

$$a : b :: c : d \text{ iff } \exists f \text{ such that } b = f(a) \text{ and } d = f(c) \tag{3}$$

which leads to the generic pattern:

$$x : f(x) :: y : f(y)$$

Solving $a : b :: c : x$ assuming that $b = f(a)$ amounts to induce $f$ to compute $f(c)$. Again, the more relevant $f$, the more useful the proportion.

Somehow these patterns fit with the intuition and satisfy the symmetry property, i.e. if $a : b :: c : d$ holds then $c : d :: a : b$ holds. Unfortunately, they are in trouble with central permutation since there is no obvious generic way to link $a$ and $c$ starting from the link between $a$ and $b$. More precisely, in terms of the general definition with $R$, the idea of central permutation would lead to state that $a : b :: c : d$ holds, we should have $S(a, c) = S'(b, d)$ for some $S$ and $S'$. We have then two relations involved due to the existence of distinct domains, and there is still no clear generic way to build $S$ and $S'$ from $R$.

However, there is a particular instance of $x : f(x) :: y : f(y)$ which easily fits the analogical scheme, namely,

$$x : f(x) :: x : f(x)$$

Note that this pattern excludes patterns of the form $x : y :: x : y'$ with $y \neq y'$ where the underlying relation linking $x$ and $y$ is not functional. Moreover, $x : f(x) :: x :?$ is an equation solvable in a unique way without the need to guess $f$: we just have to copy $f(x)$.

Moreover, in this case, we may consider that the central permutation property holds since $S$ and $S'$ will just be the equality relations over the two respective domains. A particular example of such a proportion is when $f$ is a function whose domain is a particular Cartesian product (for instance, a function with two components), leading to the pattern:

$$(x_1, x_2) : f(x_1, x_2) :: (y_1, y_2) : f(y_1, y_2) \tag{4}$$

| | isA | isB | isReversed |
|---|---|---|---|
| A | 1 | 0 | 0 |
| ∀ | 1 | 0 | 1 |
| B | 0 | 1 | 0 |
| ? | ? | ? | ? |

**Figure 3.** Boolean coding for the quizz with reversed letter.

| | |
|---|---|
| $a : b :: a : b$ or $a : a :: b : b$ | basic patterns |
| $a : f(a) :: b : f(b)$ | functional pattern |
| $a : f(a) :: a : f(a)$ | basic functional pattern |

**Figure 4.** Generic analogical proportion patterns.

This approach is still suitable for solving $A : \forall :: B :?$ if we are able to guess that the function $f$ applied to $A$ results in a 180 degree rotation of $A$. In that case, a simple Boolean coding with three attributes: $isA$, $isB$, $isReversed$ leads to the representation in Figure 3.

For the three last columns in Figure 3, the equation-solving process leads to the vector (0, 1, 1), which is the exact representation of the solution, a reversed B: Ɐ. The attributes, especially the one of the third column, acknowledge the observable effects at work when browsing the first three items of the sequence. Obviously, in the case where the function $f$, which changes the first item into the second one (namely, $a$ into $f(a)$), is explicitly identified, we are in a position to choose the appropriate attributes for encoding the items.

In Figure 4, we gather generic analogical proportion patterns. The patterns $a : b :: a : b$ (which includes the particular case $a = b$) and $a : f(a) :: a : f(a)$ (which is a special case of the functional pattern) are of particular interest. Both of them embed the idea of recopying, whereas $f(a)$ (in place of $b$) opens the possibility that $a$ and $f(a)$ belong to different sets. For instance, in the following, $a$ is a pair of Boolean variables, whereas $f(a)$ is a Boolean variable. They have straightforward vector extensions, e.g., $\overrightarrow{a} : \overrightarrow{b} :: \overrightarrow{a} : \overrightarrow{b}$ for the first one. Extensions of analogical proportion patterns to numerical features[21] or to other non-Boolean structures[22] are out of the scope of this paper, where features are Boolean (or nominal in a few cases). Let us now investigate the use of these patterns to solve picture-based quizz problems.

### 2.5. Basic Examples

It is clear that any sequence of three items $a$, $b$, $c$, represented as Boolean vectors, which has to be completed with an item $d$ to build an analogical proportion, is

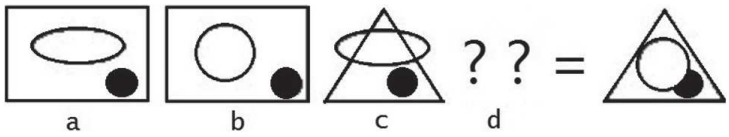



**Figure 5.** A simple analogical sequence of pictures.

| | $hS$ | $hBD$ | $hT$ | $hC$ | $hE$ |
|---|---|---|---|---|---|
| a | 1 | 1 | 0 | 0 | 1 |
| b | 1 | 1 | 0 | 1 | 0 |
| c | 0 | 1 | 1 | 0 | 1 |
| d | ? | ? | ? | ? | ? |

**Figure 6.** A Boolean coding for Figure 5.

just the vectorial extension of the equation-solving problem. Let us consider the example of Figure 5. A basic way to code the problem is to consider five Boolean predicates $hasSquare(hS)$, $hasBlackDot(hBD)$, $hasTriangle(hT)$, $hasCircle(hC)$, $hasEllipse(hE)$ in that order.

This leads to the code of Figure 6. Applying componentwise the solving process, we get $d = (0, 1, 1, 1, 0)$ which is the code of the expected solution. All the lines of the analogy truth table are used (except the line 0000 but it could be easy to have it by adding a missing feature like $hasWhiteDot$ for instance). Obviously, there are many examples of this type where only a subpart of the analogical proportion truth table is involved, or where the same pattern may occur several times.

Let us remark that this approach is constructive in the sense that the missing picture $d$ is obtained by computation from $a, b, c$. The method provides a picture which is the unique solution satisfying symmetry and central permutation, which are characteristic properties of analogical proportion. Clearly, we may imagine some sequence of pictures $a, b, c$ which cannot be completed by a fourth picture $d$ in the sense of analogical proportion since the equation $a : b :: c : x = 1$ is not always solvable. As in the case of Prade and Richard,[9] when there is no analogical solution, we may try to use another type of logical proportions among the so-called homogeneous proportions,[23] which include the analogical proportion. This may be a way to increase our ability to solve other tests beyond pure analogy, but this is out of the scope of this paper.

This contrasts with the classical approaches to this problem, pioneered by Th. Evans,[12] where $d$ is rather to be chosen among a set of candidate pictures which contains a picture considered as being the right answer.

It should also be clear that the approach may not be suitable for solving quizzes obeying to a functional pattern of the form $a : f(a) :: b : f(b)$, where the function $f$ has to be guessed on the basis of some simplicity principle.

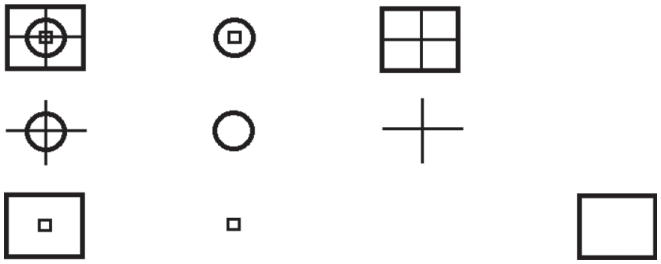


**Figure 7.** Modified Raven test 12 and its solution.

## 3. PROPOSED APPROACH FOR SOLVING RAVEN'S TESTS

Originally developed by John C. Raven in 1936, Raven's progressive matrices (RPM)[8] constitute a set of nonverbal tests where the participant is presented with a matrix ($4 \times 4$, $3 \times 3$ or $2 \times 2$) of images in which the bottom right image is missing and is asked to pick the answer that best completes the matrix, among a set of candidate answers. The resulting performance is considered as a measure of the reasoning ability of the participant, and ultimately of his/her general intelligence. An example[b] is given with its solution (a simple big square) in Figure 7. Solving an RPM is primarily a visual exercise and heavily relies on the representation of the space and objects at hand. Let us focus here on the $3 \times 3$ Raven's matrices. We denote the Raven matrix as $pic$: This is a $3 \times 3$ matrix where $pic[i, j]$ ($i, j \in \{1, 2, 3\} \times \{1, 2, 3\}$) denotes the picture at row $i$ and column $j$. $pic[3, 3]$ is unknown and has to be predicted. The other ones $pic[1, 2]$, $pic[1, 3]$, $pic[2, 1]$, $pic[2, 2]$, $pic[2, 3]$, $pic[3, 1]$, and $pic[3, 2]$ are completely known. The last picture $pic[3, 3]$ is missing and has to be chosen among a panel of eight candidate pictures (see example above). We assume that the Raven matrices can be understood in the following way, with respect to rows and columns:

$$\forall i \in [1, 2], \exists f \text{ such that } pic[i, 3] = f(pic[i, 1], pic[i, 2])$$

$$\forall j \in [1, 2], \exists g \text{ such that } pic[3, j] = g(pic[1, j], pic[2, j])$$

The two complete rows (resp. columns) are supposed to help to discover $f$ (resp. $g$) and to predict the missing picture $pic([3, 3])$ as $f(pic[3, 1], pic[3, 2])$ (resp. $g(pic[1, 3], pic[2, 3])$). This representation is summarized in Figure 8.

First of all, the extended analogical scheme has to be applied for telling us $(a, b) : f(a, b) :: (c, d) : f(c, d)$ holds for rows and $(a, b) : g(a, b) :: (c, d) :$

[b]For copyright reasons and to protect the security of the test problems, all original Raven tests have been replaced by specifically designed examples (still isomorphic in terms of logical encoding to the original ones).

| | | |
|---|---|---|
| pic[1,1] | pic[1,2] | f(pic[1,1],pic[1,2]) |
| pic[2,1] | pic[2,2] | f(pic[2,1],pic[2,2]) |
| g(pic[1,1],pic[2,1]) | g(pic[1,2],pic[2,2]) | |

**Figure 8.** Raven matrix representation.

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 0 | **?** |

**Figure 9.** A monofeature matrix with distinct column function $f$ and row function $g$.

$g(c, d)$ for columns, which translates into

$$(pic[1, 1], pic[1, 2]) : pic[1, 3] :: (pic[2, 1], pic[2, 2]) : pic[2, 3]$$

$$(pic[1, 1], pic[2, 1]) : pic[3, 1] :: (pic[1, 2], pic[2, 2]) : pic[3, 2]$$

Similar analogical proportions are supposed to relate row 1 to row 3 and row 2 to row 3, and the same for the columns. Thus, in that case, we have to consider a pair of cells $(pic[i, 1], pic[i, 2])$ as the first element $a$ of an analogical proportion, whereas the second one $b$ is the singleton $pic[i, 3]$. The pictures are assumed to be described in terms of Boolean features. Given such a Boolean feature, the values of this feature of the two first cells of a row (resp. a column) functionally determines the value of this feature in the third cell in the row (resp. the column). This view calls for the use of the fourth pattern $a : f(a) :: a : f(a)$ of Figure 4, linking different types of items (here pairs of values and single values) and where having the three first elements $a : f(a) :: a :?$, there is no need to guess $f$ to complete the proportion.

We also assume that $f$ and $g$ are defined componentwise with respect to the $n$ components of the vectors encoding the pictures, i.e., $f = (f_1, \ldots, f_n)$ and $g = (g_1, \ldots, g_n)$. In the following, these vectors are just the encoding of Boolean features.

Note that nothing forbids $f$ and $g$ to be distinct functions as illustrated by the example in Figure 9, where $? = 1$. Indeed, in that case, $f(1, 0) = 1$ (see row 2) and $g(0, 1) = 1$ (see column 1). Observe also that $g(1, 0) = 0$ (column 2) while $f(1, 0) = 1$, thus $f \neq g$.

| | $column1$ | $column2$ | $column3$ |
|---|---|---|---|
| $row1$ | 11 | 01 | 00 |
| $row2$ | 11 | 10 | 11 |
| $row3$ | 00 | 01 | ?? |

**Figure 10.** An example where a pattern to be completed cannot be found for the same feature.

### 3.1. Feature-Based Representation

In the following, we assume that the pictures are represented as vectors of high level Boolean features. This means that these features are provided by an external system or manually. In the case of Figure 7, each picture $pic[i, j]$ is represented as a Boolean vector of dimension 4, the four binary features denoting the presence of a Big Square, a Small Square, a Circle, and a Cross. In the next section, we investigate an automatic way to proceed using a pixel-based encoding. Then we can apply the basic functional pattern of Figure. 4, vertically and horizontally, to try to build up the missing picture. Let us first illustrate the idea on examples, before summarizing the whole procedure.

*Situation* 1: On the toy example in Figure 9, we can see that the horizontal pattern (1, 0) is completed in row 2 with $f(1, 0) = 1$. This suggests that the missing bit is one since we check that there is no contradiction of this row completion with respect to the other remaining row. But we have to do the same job with the columns. The corresponding vertical pattern (0, 1) is then completed in column 1 by $g(0, 1) = 1$. Again there is no contradiction of this column completion with respect to the other remaining column. Since the vertical analysis and the horizontal analysis lead to the same value 1, 1 is then acceptable as a solution for the missing value.

However there are two other situations missing in this example of Figure 9 that could be encountered.

*Situation* 2: This case is faced when the above process leads to a contradiction either internally within the rows, or internally within the columns, or between the rows and the columns candidate solutions. In those three cases, we cannot suggest any solution and the algorithm stops by giving a "no solution found" message.

*Situation* 3: The third case is when the horizontal (resp. vertical) pattern cannot be found in the two other rows (resp. columns). We illustrate this situation on another toy example in Figure 10.

Considering the first missing feature, we are in the context of situation 1 again: the horizontal pattern (0, 0) to be completed in row 3 does not appear neither in line 1 nor in line 2, so we have to move to a vertical analysis and to look for the vertical pattern (0, 1) to be completed in column 3, that we find in column 2 with solution 0. Since there is no contradiction, we can suggest 0 as a solution for this missing feature.

Unfortunately, when looking for a pattern for solving the second feature, we do not find the pattern (0, 1) neither horizontally (in rows 1 or 2) nor vertically (in columns 1 or 2) for this feature. We are exactly in the context of situation 3 described above.

We have to consider that a Raven matrix expresses a set of analogical proportions without any consideration of a particular feature. For this reason, to find the solution for feature $i$, we may take lesson from other features by looking for a solution in a row or a column coming from another feature $j \neq i$. If, *whatever* the considered feature in row or column, we always find the same solution, then we allocate its value to the missing feature. More precisely, to complete an horizontal pattern in row 3 pertaining to feature $i$, we have to check that this pattern, when appearing in another row for any other feature, is always completed in the same way. Moreover, we have also to check that the vertical pattern in column 3 for feature $i$, when appearing in another column for any other feature, is again completed in the same way. Finally, we have to check that the solution coming from the horizontal analysis is identical to the solution coming from the vertical one. In that case, this solution can be proposed, otherwise the algorithm stops by giving a "no solution found" message as in situation 2.

In the case of Figure 10, regarding the second missing feature, we observe that the pattern (0, 1) (here both found in row and column) to be completed in row 3, does not appear in rows 1 and 2 (resp. columns 1 and 2). But we find this pattern (0, 1) vertically in the second column for the first feature: This leads to solution 0. Since there is no other pattern (0, 1) leading to another solution, we allocate the value 0 to the second feature in $pic[3, 3]$. And we have completed the matrix.

Let us now summarize the procedure that is implemented in algorithm ANARAVEN. For finding the value of feature $i$ in the ninth-cell of the Raven matrix, we first consider the horizontal pattern to be completed in row 3 and the vertical pattern to be completed in column 3 for this feature.

1. If the horizontal (resp. vertical) pattern is found horizontally (resp. vertically), always with the same completion and the horizontal and vertical completions coincide, a solution has been found for feature $i$. If only either the horizontal or the vertical pattern is found (always with the same completion), still the completion found is considered as the solution for feature $i$.
2. If there is an horizontal contradiction, or a vertical one, or the horizontal and the vertical solutions do not coincide, we are unable to propose a value for feature $i$.
3. If both the horizontal and the vertical patterns cannot be found in the rest of the matrix for feature $i$, we have to apply the same procedure feature by feature, for all the other features $j \neq i$. Then a solution is obtained if and only if the solutions that may be found for some feature(s) coincide. Otherwise the algorithm stops by giving a "no solution found" message as in situation 2. This may be due to the fact that for some features $j$ we are in some of the three contradictory situations as explained in the previous item, or that there are two features $j$ and $k$ leading to distinct solutions.

ANARAVEN is applicable to any Raven test represented as assumed above (i.e., a Boolean vector in $\{0, 1\}^n$). Given an index $i$ corresponding to a feature, ANARAVEN first searches for an existing pattern leading to an answer by means of the function *existAnswer*, which calls for a consistency checking via the *existContradiction* function. If successful, the function *getAnswer* returns the unique solution for feature $i$.

**1 Algorithm ANARAVEN**

**Require:** picture array: p[3,3][n] ▷ n is the number of features, p[3,3] is unknown

1: **for** $k = 1$ to $n$ **do**
2: **if** $existAnswer(k)$ **then**
3: **if** $existContrad\ (k, getAnswer(k)) == false$ **then**
4: $p[3,3][k] \leftarrow getAnswer(k)$
5: **else**
6: $write\ (\ "NO\ SOLUTION\ FOUND")$
7: **end if**
8: **else**
9: $exAnswer \leftarrow false$
10: $error \leftarrow false$
11: $j \leftarrow 0$
12: **while** $j < n\ \&\ error = false$ **do**
13: **if** $j! = k$ **then**
14: **if** $existAnswer(j)$ **then**
15: **if** $existContrad\ (j, getAnswer(j)) = false$ **then**
16: **if** $exAnswer\ \&\ existContrad\ (j, p[3][3][k])$ **then**
17: $write\ (\ "NO\ SOLUTION\ FOUND")$
18: $error \leftarrow true$
19: **else**
20: $exAnswer \leftarrow true$
21: $matrix[3][3][k] \leftarrow getAnswer(j)$
22: **end if**
23: **end if**
24: **end if**
25: **end if**
26: $j \leftarrow j + 1$
27: **end while**
28: **end if**
29: **end for**

30: **function** $existAnswer(index\ f)$ ▷ returns a Boolean value
31: $exists \leftarrow false$
32: **for** $k = 1$ to 2 **do**
33: $okl \leftarrow p[k,1][f] = p[3,1][f] \& p[k,2][f] = p[3,2][f]$
34: $okr \leftarrow p[1,k][f] = p[1,3][f] \& p[2,k][f] = p[2,3][f]$
35: **if** $okl\ OR\ okr$ **then** $exists \leftarrow true$
36: **end if**
37: **end for**
**return** exists
38: **end function**

```
39: function existContrad(index f, value)                    ▷ returns a Boolean value
40:     exists ← false
41:     for k = 1 to 2 do
42:         okl ← p[k,1][f] = p[3,1][f]&p[k,2][f] = p[3,2][f]
43:         okr ← p[1,k][f] = p[1,3][f]&p[2,k][f] = p[2,3][f]
44:         if (okl AND (value != p[k,3][f])) OR (okr AND (value != p[3,k][f]))
    then
45:             exists ← true
46:         end if
47:     end for
        return exists
48: end function
```

```
49: function getAnswer(index f)                              ▷ returns an index
50:     exists ← false
51:     for k = 1 to 2 do
52:         okl ← p[k,1][f] = p[3,1][f]&p[k,2][f] = p[3,2][f]
53:         okr ← p[1,k][f] = p[1,3][f]&p[2,k][f] = p[2,3][f]
54:         if (okl) then answer ← p[k,3][f]
55:         end if
56:         if (okr) then
57:             answer ← p[3,k][f]
58:         end if
59:     end for
        return answer
60: end function
```

### 3.2. Pixel-Based Representation

Instead of dealing with a small number of high level features for describing the pictures, we may consider an image as a matrix of pixels of dimension $n \times m$ (considering only noncompressed format as BMP (bitmap), for instance), this is again a Boolean (or similar) coding automatically performed by the picture processing device (e.g., the camera or the scanner), where *each pixel*, black or white, plays the role of *a feature*. In that case, instead of dealing with eight pictures hand-coded as Boolean vectors, we deal with eight BMP files. Apart from the fact that we have to take care of the headers of the files (which do not obey any proportion pattern), there is no reason to change our method and our basic algorithm still applies without any change to the data part of the files. Clearly, we have to implicitly assume that the eight files have exactly the same dimensions in order for our algorithm to work: The RPM tests are amenable to such representation since they are entirely picture based.

In the following section, we experiment with both representations and we analyze the results.

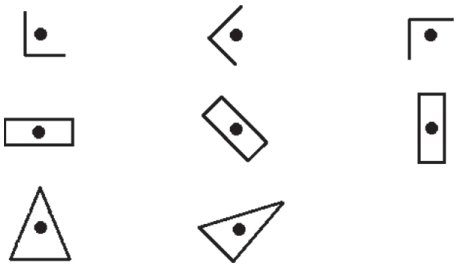


**Figure 11.** Modified Raven test 10.



**Figure 12.** Modified Raven test 10: What the algorithm finds.

**Figure 13.** Modified Raven test 10: Actual solution.

## 4. RESULTS AND EXAMPLES

There are diverse sets of RPM available in the literature. In this paper, we consider the advanced scheme, which is divided into a set of 12 tests and a set of 36 tests. Dedicated to adults with above-average IQ, the second batch is supposed to be the most difficult one. We have applied our algorithm to these 36 tests, both at a pixel level and with high level Boolean description. We now present examples of success and then discuss the failures. We recall that we do not provide the set of candidate solutions, normally given with a complete RPM test.

### 4.1. Pixel-Based Approach

Let us start with the pixel approach where we rely on the picture device to provide the coding of the pictures. The only input of our algorithm is a list of eight BMP files of the same size, corresponding to the sequence of eight pictures of a given Raven test. When we apply our algorithm to the 36 tests of our chosen test set, we get 16 successful results, leaving 20 unsolved. Among the tests that we solve with this approach, there is test 12 (in Figure 7) already mentioned. In such a case, it is possible to build the solution point by point (i.e., pixel by pixel) using the analogical proportion–based mechanism described in Section 3. Note that in Figure 7, there is no rotation or modification of shapes, this is just a matter of presence or absence (in each cell) of a big square, of a small square, of a circle, and of a cross, and this amounts, pixel by pixel, to the presence (or absence) of collections of black points at definite places that altogether figure the different shapes.

An example of a failure is test 10 in Figure 11. The solution that the algorithm provides starting from a pixel representation is shown Figure 12, and the actual solution is shown Figure 13, showing that the obtained solution just retains the

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 100100 | 100010 | 100001 |
| 2 | 010100 | 010010 | 010001 |
| 3 | 001100 | 001010 | ?????? |

**Figure 14.** A Boolean encoding of the modified Raven test 10.

constant feature appearing in all pictures. The algorithm is unable to provide the external lines needed to get the proper solution. This should not come as a surprise, and we cannot expect from such a low level (pixel) representation to capture the whole information coming from the eight pictures. Generally speaking, the proposed method is not successful when the variation of an object between two successive images is a matter of modification or rotation of a shape. It is nevertheless remarkable that this simple view is sufficient for solving 16 tests. Let us move to the higher level approach where we still consider a Boolean encoding, but which has now been manually done.

### 4.2. Feature-Based Approach

When using the high level feature-based approach, we can solve, on top of the previous list, 16 new tests, leading to a total of 32 test problems among 36. Let us consider here two examples, which are considered as relatively difficult in the RPM scale. First of all, consider test 10 that we previously failed to solve with the pixel-based approach. We can use the following encoding which provides the exact solution. Since the central dot is a constant in the eight pictures, we know that this dot will appear in the picture we want to build (and this is exactly what happens with the pixel-based approach). Then, we do not consider it in the following. Then, we have six features covering the problem leading to a picture being represented as a Boolean vector of dimension 6: (i) three features describing the figures: $corner$, $rectangle$, and $triangle$, (ii) three features describing the position of the figures $position1$, $position2$, and $position3$ where $position2$ denotes a 45-degree clockwise rotation from $position1$ and $position3$ denotes a similar rotation from $position2$.

We show the complete encoding in Figure 16. The features are coded in this order: $corner$, $rectangle$, $triangle$, $position1$, $position2$, $position3$. With a vertical analysis, we get the following code 001001 for the last picture. This means that the answer must include a triangle in $position3$. We have to note that we cannot vertically complete the pair 11 for the last feature: We have then to look for another target feature for finding a similar context in the first column (for instance) and subcolumn 4, which gives us the final value: 1.

Let us now come to a second example, namely, test 27, which we provide with its solution in Figure 15. A Boolean coding of this test uses the features (the ordering below is the one used in Figure 16): (i) for the presence (or not) of lines in $outsideSquare$, $centralSquare$, $insideSquare$; (ii) for the direction of the lines: $vertical$, $diagonal$, and $horizontal$, leading to Boolean vectors of dimension 6. We

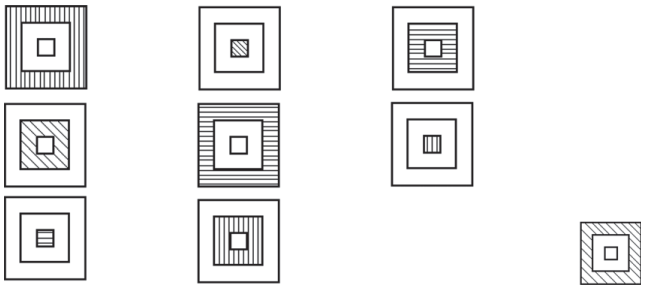


**Figure 15.** Modified Raven test 27 and its solution.

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 100100 | 001010 | 010001 |
| 2 | 010010 | 100001 | 001100 |
| 3 | 001001 | 010100 | ?????? |

**Figure 16.** A Boolean encoding of the modified Raven test 27.

| | pixel approach | feature-based approach |
|---|---|---|
| solvable | 16 | 16+16 |
| not solvable | 20 | 4 |
| **total** | 36 | 36 |

**Figure 17.** Results for 36 Raven tests.

| | | | | | | |
|---|---|---|---|---|---|---|
| R | V | B | | 1000 | 0100 | 0010 |
| V | B | J | –> | 0100 | 0010 | 0001 |
| B | J | ? | | 0010 | 0001 | ? |

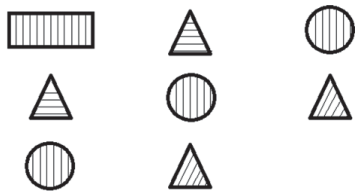
**Figure 18.** A typical failure case with the Boolean encoding.

show the complete encoding in Figure 16. By solving this matrix with our algorithm, we get the encoding 100010 which is the exact code of the solution picture.

To conclude this section, we summarize our results in Figure 17. It appears that four Raven's tests are not solvable with the proposed approach. In the following, we propose a careful examination of these tests, either suggesting that an advanced representation may be useful in some cases, or that in other cases, a constraint satisfaction mechanism may be needed.

### 4.3. Using Post Algebra

Among the cases where the procedure fails, some are due to the fact that a non-Boolean attribute is encoded through a finite set of Boolean features. For example, in Figure 18, where we have an attribute ranging over the four values $R$, $V$, $B$, and $J$ and where the solution is $R$, the Boolean encoding leads to inconsistencies when applying the approach described in subsection 3.1. Indeed, for the first feature, we



$a_1$ : rectangle $a_2$ : triangle $a_3$ : circle
$b_1$ : horizontal $b_2$ : vertical $b_3$ : diagonal

**Figure 19.** Example with an encoding "à la Post".

| | | |
|---|---|---|
| $a_1b_2$ | $a_2b_1$ | $a_3b_2$ |
| $a_2b_1$ | $a_3b_2$ | $a_2b_3$ |
| $a_3b_2$ | $a_2b_3$ | x y |

**Figure 20.** Encoding with Post algebra.

**Figure 21.** Solution of "Post example."

have to complete the pattern 00 and there is only one option which is 0 as revealed by looking at feature 1, in the second column. But for the second feature, we have again to complete the pattern 00, but we do not have any solution by looking horizontally or vertically at the second feature. So, we look at other features and we find, from feature 1, we get 0 whereas from feature 3 (from row 1) we get 1. We get a contradiction.

In such a case, one may use an encoding based on Post algebra,[24] as briefly described by Prade and Richard.[25] In this approach, an operator $\sigma$ enables us to define a cyclic negation, while preserving the definition of the analogical proportion in an extended way. Namely, given a total ordering over a set of items $\{a_1, ..., a_n\}$ (the ordering may be chosen arbitrarily), one defines $\sigma(a_i) = a_{i+1}$, for $i \in [1, n-1]$, and $\sigma(a_n)$=$a_1$. The negation of an element $a$ is simply $\sigma(a)$.

The analogical proportion schema becomes $a : \sigma^k(a) :: b : \sigma^k(b)$, and we can notice that the example in Figure 18 can now be solved with this encoding and the order $R$, $V$, $B$, $J$.

Let us consider a complete example (a RPM test), the one in Figure 19, where we use two attributes, a shape attribute with values *rectangle*, *circle*, *triangle*, and a second attribute with values $vertical$, $horizontal$, and $diagonal$ for describing the lines with the following ordering (as suggested by the two first rows).

This leads to the encoding shown in Figure 20. To take advantage of the cyclic encoding, we also need to apply analogy with the form of continuous proportions, which corresponds to the pattern $A$:$B$::$B$:$C$, for expressing (in row or in column) that if the cell $C$ comes after cell $B$ that comes after $A$, then in some sense "$A$ is to $B$ as $B$ is to $C$." Then, for solving with respect to attribute $a$, we have to

**Figure 22.** Example of a failure.

evaluate the pattern $a_3 : a_2 :: a_2 :?$ according to Post logic. Since $a_2 = \sigma^2(a_3)$, we get $x = \sigma^2(a_2) = a_1$. The same reasoning applies for attribute $b$ with the pattern $b_2 : b_3 :: b_3 :?$ where $b_3 = \sigma(b_2)$ then we get $y = \sigma(b_3) = b_1$. It turns out that $a_1 b_1$ is exactly the encoding of the expected solution (Figure 21). This enables us to solve this test and another one where we previously failed. This second test is shown in Appendix A, section A.1 with its encoding and its solution.

As can be seen, the solving of these two additional tests is done by means of an analogical proportion pattern, extended to non-Boolean domains, and also by means of a new analogical reading of the Raven matrix where a continuous analogical proportion is assumed to hold inside each row (and column), while the approach developed in this paper is based on analogical proportion between two rows or two columns. The full investigation of the alternative analogical reading introduced above is left for further research.

### 4.4. The Two Remaining Failure Cases

The analogical approach seems unable to solve the two remaining Raven tests (among the 36 tests). In one of the cases, when using cyclic or Boolean encodings, it leads to inconsistencies, i.e. we get two distinct answers for the same feature. However, in this case, the Boolean encoding yields the correct unique answers for all features but one, and for the feature where we get contradictory answers, one of the options is the right one, whereas the other leads to a feasible solution which is not among the eight candidate solutions proposed in the test. This would enable to conclude by elimination, but then the approach is no longer entirely constructive. Details about this test are given in Appendix A, section A.3.

Figure 22 exhibits the last case where the proposed approach fails. In this example, we have to predict the three figures that altogether make the answer, as well as their relative position. With a Boolean encoding, using the names of the figures as attributes (*circle*, *triangle*, *square*, *star*, *hexagon*) without taking into account either their position or their number of occurrences in each image, we conclude that the image of the answer should include a circle, a triangle and a

**Figure 23.** The solution for the test given in Figure 22.

square (see the encoding in Appendix A, section A.2). The approach cannot succeed in finding the relative positions of these figures (see Appendix A, section A.2).

It can be noticed in Figure 22 that there are situations that always hold both in rows or in columns: (a) There are two stars in each row and each column; (b) there are two circles in each row and each column, and moreover in columns they are in the same position; (c) there are two triangles in each row and each column, and moreover in rows, they are in the same position; (d) there is one hexagon in each row and each column. This suggests the need to evaluate the rows and the columns in terms of the above-mentioned constraints.

Constraint (b) ensures that a circle maintains its position in each column, and since in image [3, 1] the circle is on the right, it should have the same position in the solution. By using constraint (c), one concludes that the triangle should be on the left of the image and this leads to deduce that the square should be in the middle of the image. The final answer is thus "triangle square circle" as shown in Figure 23 (see Appendix A, section A.1. for details).

This shows that the complete solving of this example, beyond copying, involves constraints between characteristics for being able to conclude here that the central remaining position can be only occupied by the remaining element (the square): This does not seem to be easy to handle with a pure analogical approach since the analogical readings (between or inside rows and columns) fail to provide a solution for this test. One may also notice on Figure 22 that the solution corresponds to the unique permutation of the three elements (triangle, square, circle) that is missing, the three other cells corresponding to all the possible permutations of a hexagon and two stars. This seems beyond a standard definition of analogical proportion.

## 5. RELATED WORK

There is a huge literature on computational models of analogical reasoning with applications in different areas. For an introductory overview of computational trends in analogical reasoning, the reader is referred to Prade and Richard' bibliographical study.[26] The use of analogical proportions to solve geometric quizzes is just a particular instance of analogical reasoning. We can consider Th. Evans[12,27] as a pioneer in this domain. He designed a program able to select an answer to complete a sequence of three geometric figures $A$, $B$, and $C$. The problem was then to find, among a given set of candidate solutions, the figure $X$ that gives the best fit to the analogical proportion "$A$ is to $B$ as $C$ is to $X$." The program exploited appropriate representations of the geometrical figures in terms of how subfigures relate in a given figure and tried to match rules describing changes from $A$ to $B$ with rules describing changes from $C$ to $X$, for the different $X$s taken in the set of candidate solutions. This approach did not explicitly rely on a formal framework to analogical reasoning and was specifically tailored to solve analogies between geometric patterns.

When it comes to provide a formal framework for analogical reasoning, we can roughly distinguish two kinds of approaches: the symbolic ones and those incorporating connectionist features.[1] Among the first type of approaches, we can distinguish:

- the ones based on a purely logical approach, either first-order logic[28] or second-order logic[29,30]
- the ones proposed by Gentner[31] and Winston,[32] more model oriented, which use appropriate symbolic representations (for instance graphs) of the involved items and consider the analogy-making process as a matching process allowing to reveal some hidden analogical relations.

More problem-driven, the connectionist view[33–35] makes use of constraint satisfaction networks to determine, through an optimization process, the best solution among competing hypotheses.

RPM tests are a particularly challenging case of geometric quizz. Automatically solving RPM has been attempted with diverse computer models. In a comparative overview of computer models to solve intelligence test problems,[36] survey about 30 systems, among which half a dozen deal with RPM tests. We review these works below.

### 5.1. The First Model

As far as we know, the first effective computational model targeting RPM was developed by Carpenter et al.[10] Based on psychological studies, it has been found that most humans solve these tests in the following way:

- by observing the top row of the matrix, starting to compare pairs of consecutive pictures, to generate hypotheses about how the pictures vary along in the first row and to guess transformations rules from their observations,
- then by testing those hypotheses by looking at the middle row,
- finally by mapping the similarities and dissimilarities between those first two rows and the bottom one to know how to apply the same rules to this row.

Starting from these observations, Carpenter et al.[10] have implemented two programs, namely FAIRAVEN and BETTERAVEN, both of them using hand-coded input representations, and a set of six human-inspired rules implemented in the system. Given a Raven matrix, the two systems proceed as follows: (1) identifying which rules among the six rules apply to the first two rows and (2) computing a mapping between those two rows and the bottom row to determine how to apply the same rules in that row.

Whereas FAIRAVEN could perform at the level of an average participant, BETTERAVEN was able to match the performance of the best participants to RPM tests. It is not clear how the information coming from the columns is used in the global process. The authors do not make any reference to analogical reasoning.

Lovett et al.[11] suggested for the first time that beyond spatial reasoning, RPM appears to make use of another important cognitive ability, namely, the ability to perform analogy and more generally comparison. Starting from this assumption, the authors combine qualitative spatial representations with analogical comparison

via structure mapping, leading to a more powerful model than the previous one. The approach relies on a very successful model of analogy, the so-called structure mapping theory (SMT),[37] and on its computational counterpart, namely, the structure mapping engine (SME)[38] that we review in the next subsection.

### 5.2. Structure Mapping Theory and Structure Mapping Engine

SMT and its implementation SME are assumed to emulate the way human beings buildup analogies by mapping knowledge from one source domain to a target domain. The input of SME is the encoding of two contexts via entities, predicates, and relations linking predicates and entities. Acyclic graphs whose nodes are entities and the root is a higher order relation linking the elements of the graph, are thus obtained. Matching hypotheses are created between two context descriptions, modulo identities of predicates or entities. They are combined to build maximal set of coherent hypotheses. A conflict is encountered when an expression cannot match two distinct expressions.

As some expressions may appear in one graph but not in the other one, a maximal set of matching hypotheses allows the handling of candidate expressions possibly valid in the graph where they do not appear. Each maximal set is associated with a score taking into account the underlying graph structure.

The output of SME is made of the maximal sets with their corresponding score and their candidate inferred expressions. When it comes to solve RPM tests, CogSketch, a sketch understanding system created by Forbus et al.,[39] takes as input a sketch drawn by the user, which has to be segmented into objects and generates a qualitative representation of those objects (or their edges and groups of objects), and their relations (relative position, topology, etc.). For instance, CogSketch can tell which objects are placed side by side, whether two objects intersect, or whether one is located inside another. At the end of the process, each picture is represented as an entity with attributes and relations with other entities. At this stage, we have a representation of the relative position of the objects.

CogSketch uses this edge level representation (which identifies the corresponding edges in two distinct objects) to compare two objects in a sketch, with the aim of determining whether there is a transformation (rotation, size modification) or even a deformation (total shape change) between these two objects. With this information, the objects with equivalent or strict shapes in common, are grouped together. At this stage, we have a representation of the modification between objects.

To select the correct answer for the target test, the system described by Lovett et al.[11] proceeds as follows:

1. The first two rows of the current matrix are evaluated via SME to generate some rules for both of them, which are called *pattern of variance* and are a representation of how the objects change across the row of images. There are four different strategies available to build up these patterns of variances (Lovett et al.[11] provided us with more details).
2. SME is then used again, but now for comparing the two patterns of variance previously found for the top two rows and obtaining a *similarity score*. This comparison is called *second-order* comparison as it operates on patterns instead of object representations.
3. This similarity score is compared to a threshold to determine its validity.

4. If the patterns of variance are considered similar enough, an *analogical generalization* (which is a new pattern) is built describing what is common to both rows.
5. Each one of the eight candidate answers is scored by inserting that answer into the bottom row, computing a pattern of variance, and then using SME to compare this pattern to the generalization pattern for the top two rows. The final answer is the one with the highest score.
6. In the case where the two patterns of variance corresponding to the top rows are not similar enough, another strategy is applied.

That approach overcomes the limitations of the work by Carpenter et al.[10] because

- the spatial representations of the pictures are automatically generated using CogSketch (instead of being hand-coded), and
- there is no set of built-in rules since SMT is used for generating the rules that have to be applied to the bottom row.

Finally, this model was evaluated on the 48 problems of the sections B– E of the Standard Progressive Matrices tests, solving 44 of these problems. Lovett et al.[11] argued that the four remaining problems without solution are among the six hardest ones for human participants, being the ones with the lowest resolution scores. Again, it is not clear how the information coming from the columns is used in the system.

At this stage, four main differences can be observed between the SMT approach and the one presented in this paper:

1. First, the feature encoding of the pictures is not automatically handled, whereas Lovett et al.[11] use CogSketch for getting the encoding. Still, when images are entered in CogSketch, the type (e.g., rectangles, crosses) of each figure in each image has to be given to the program, which is then able to generate the representation of the relation between the figures. The choice of specific features to focus on is certainly a part of the cognitive difficulty of the tests for humans. For representing each problem (especially when tests cannot be solved by working at the pixel level), we use a Boolean description of all the features that are present or absent. These basic features often pertain to subparts easy to identify (e.g., "is there a circle?," "is it black?"), but may also pertain to the evolving position of a component, and in a few cases to shapes that are transformed in some way (e.g., "is the circle elongated?"). In any case for getting the encoding, there is no need to know the solution.
2. Once we have a feature-based description of the matrix, the remaining problem to be solved is not fairly straightforward. Indeed, Lovett et al.[11] have to build rules for describing changes in the first two rows of the matrix combined with four different strategies to be applied to solve a test. In particular, the change patterns identified in the first two rows are then compared to the change patterns that would be at work in the third row, when the empty cell is replaced by each candidate solution. The solution which is associated with change patterns that maximizes the similarity with the change patterns identified in the first two rows, is chosen. The procedure of Lovett et al.[11] seems more intricate than the one presented here where the same simple and uniform principle is applied whatever the context.
3. Our approach makes use of both, rows and columns of the matrix. If a matching pattern is not found within the rows, we look for such a pattern within the columns. It is an open question to know if all Raven's tests can be solved only by observing the rows.
4. Finally, our approach provides a *constructive* process to build up an analogical proportion, defining step-by-step the final solution picture, without choosing it among a set of candidate solutions.

The combination of the two approaches might be investigated to try to solve the four tests where we (partially) fail, apart of the four tests that Lovett et al.[11] missed.

### 5.3. Other Approaches

Quite at the same time as the conference version of a work by Correa et al.,[14] another constructive approach for RPM has been proposed by Strannegård et al.[40] This work does not explicitly refer to analogy, but relies on a principle close to the idea of *recopy*. Indeed the authors summarize their method in the following way: "The proposed solving strategy relies on pattern matching. Problems can be processed either row or column-wise; patterns holding for both of the first two are applied to the third one, obtaining a prediction value for the solution cell." The performance of the method is measured on the standard progresssive matrices (SPM) where 28 over 36 problems are completely solved this way.

Apart from the symbolic approaches to RPM automatic solving, there is a another recent alternative involving only visual strategies that use iconic representations. This approach has been mainly investigated in a work by McGreggor et al.[41] where two methods are presented (the so-called *affine* and *fractal* methods) which share the same two intuitions: comparing images under a variety of transformations, and judging the similarity based upon features which arise from the images. Each of the methods compares images (or fragments of images) under a variety of transformations and, roughly speaking, proceeds as follows:

- Induce a best-fit composite transformation for a set of collinear elements in the matrix. This can work on raw scanned images, without any intermediary modification (i.e. grayscale PNG images).
- Apply this transformation to the parallel set of elements containing the empty element; the result is a predicted answer image.
- Compare this predicted image to the given candidate solutions for maximum similarity. In terms of similarity, the ratio model as described by Tversky[42] is used. The candidate solution having maximum similarity with the predicted image is considered as the final answer to the test.

These two *visual* methods show a robust level of performance both for SPM[43] and advanced progressive matrices.[41] Still, they are not constructive methods allowing to build a solution from scratch, without choosing among a set of already built candidate solutions.

From another perspective, Ragni et al.[44,45] have developed a system for solving RPM, implemented on top of a cognitive architecture ACT-R,[46] a theory for simulating and understanding human cognition. As it is the case with Carpenter's work, they have to identify, among a set of five rules, which one applies to a given picture. Each picture of the matrix is described via a static set of attributes such as shape, size, color, rotation, etc.

Despite the method is not constructive, the system performs quite well and, in fact, slightly better than BETTERAVEN, solving around 91% of the tested RPM. The authors have also added to their experiments explanations related to the success

or failure of the system, establishing a clear correlation between response times and the difficulty of the test.[45,47]

## 6. CONCLUSION

We have presented an approach based on a unique principle that enables us to solve almost all (advanced) RPM problems in a constructive manner. The few ones which have not been solved involve constraint propagation mechanisms and would apparently require a more sophisticated mechanism. In a number of cases, the approach can be successfully applied at the pixel level, providing the exact answer as it is expected.

It is remarkable that the approach relies on a simple and uniform principle, which follows from the logical modeling of analogical proportions, and in fact here from a specialized form of this logical modeling. However, the approach discussed in this paper has privileged a reading in terms of analogical proportions between rows and columns, but another reading in terms of continuous analogical proportions inside rows and columns may also be considered and would be worth investigating.

The approach presented here amounts to copy what can be observed in the matrix and to reproduce it if no other observation (in the context of other features) goes in the opposite way and prevents the copy. This computational approach based on analogical proportions may raise questions of interest from a cognitive science viewpoint. Is it the case that most of Raven test participants try an analogical reading of the matrix? If yes, of what type? (Between or inside rows and columns), what is the role of the copying in their choice of the solution? When solving such a test, what are the respective roles of similarities and differences between pictures? Answering all these questions may help to build more accurate computer models than the ones investigated in this paper.

Beyond IQ tests, the analogical proportion solving principle can be of interest in different reasoning tasks such as extrapolation,[48] classification,[6,49] and more generally, machine learning.[50]

## References


1. French RM. The computational modeling of analogy-making. Trends Cogn Sci 2002;6(5):200–205.
2. Miclet L, Prade H. Logical definition of analogical proportion and its fuzzy extensions. In: Proc Annual Meeting of the North American Fuzzy Information Proceedings Society (NAFIPS), New York, May 19–22, IEEE: NY; 2008. pp 1–6.
3. Miclet L, Prade H. Handling analogical proportions in classical logic and fuzzy logics settings. In: Proc. 10th Eur Conf on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'09)Verona, Italy, July 1–3, 2009. LNS, Vol. 5590; Verona: Springer; 2009. pp 638–650.
4. Prade H, Richard G. Reasoning with logical proportions. In: Lin FZ, Sattler U, Truszczynski M, editors. Proc. 12th Int Conf on Principles of Knowledge Representation and Reasoning (KR'10), May 9–13, 2010. Toronto, Canada: AAAI Press; 2010. pp 545–555.
5. Prade H, Richard G. Cataloguing/analogizing: a nonmonotonic view. Int J Intell Syst 2011;26(12):1176–1195.

6. Miclet L, Bayoudh S, Delhay A. Analogical dissimilarity: definition, algorithms and two experiments in machine learning. J Artif Intell Res 2008;32:793–824.
7. Prade H, Richard G, Yao B. Enforcing regularity by means of analogy-related proportions—a new approach to classification. Int J Comput Inform Syst Ind Manag Appl 2012;4:648–658.
8. Raven J. The Raven's progressive matrices: change and stability over culture and time. Cogn Psychol 2000;41(1):1–48.
9. Prade H, Richard G. Analogy-making for solving IQ tests: a logical view. In: Proc 19th Int Conf on Case-Based Reasoning, Greenwich, UK, Sept 12–15, 2011. Berlin: Springer; 2011. pp 561–566.
10. Carpenter PA, Just MA, Shell P. What one intelligence test measures: a theoretical account of the processing in the Raven progressive matrices test. Psychol Rev 1990;97(3):404–431.
11. Lovett A, Forbus K, Usher J. A structure-mapping model of Raven's progressive matrices. In: Proc. 32nd Annual Conf of the Cognitive Science Society, Portland, OR; 2010.
12. Evans Th. G. A program for the solution of a class of geometric-analogy intelligence-test questions. In Minsky MK, editor. Semantic information processing; Cambridge, MA: MIT Press: 1968. pp 271–353.
13. Correa Beltran W, Prade H, Richard G. Quand l'intelligence est juste une question de recopie. In: Actes 6ièmes Journées de l'Intelligence Artificielle Fondamentale (JIAF'12), Toulouse, France, May 22–24, 2012. pp 77–86.
14. Correa Beltran W, Prade H, Richard G. When intelligence is just a matter of copying. In: Proc 20th Eur Conf on Artificial Intelligence, Montpellier, France ; Aug. 27–31, 2012. Amsterdam, The Netherlands: IOS Press; 2012. pages 276–281.
15. Prade H, Richard G. From analogical proportion to logical proportions. Logica Universalis 2013;7(4):441–505.
16. Piaget J. Logic and psychology. Manchester, UK: Manchester University Press; 1953.
17. Hesse M. Models and analogy in science. In: Edwards P, editor. The encyclopedia of philosophy, Vol. V, New York: Macmillan; 1967. pp 354–359.
18. Stroppa N, Yvon F. Analogical learning and formal proportions: definitions and methodological issues. Technical report ENST D-2005-004, Paris, June 2005.
19. Stroppa N, Yvon F. Du quatrième de proportion comme principe inductif : une proposition et son application à l'apprentissage de la morphologie. Trait Autom Langues 2006;47(2):1–27.
20. Hesse M. On defining analogy. In Proc Aristotelian Soc 1959;60:79–100.
21. Prade H, Richard G. Analogical proportions and multiple-valued logics. In: Proc. 12th Eur Conf on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'13), Utrecht, The Netherlands, July 7–10, 2012. LNAI, Vol. 7958. Berlin: Springer; 2012. pp 497–509.
22. Miclet L, Barbot N, Prade H. From analogical proportions in lattices to proportional analogies in formal concepts. In: Proc 21st Eur Conf on Artificial Intelligence (ECAI'14), Prague, Aug. 18–22, 2014. Amsterdam, The Netherlands: IOS Press; pp 627–632.
23. Prade H, Richard G. Homogeneous logical proportions: their uniqueness and their role in similarity-based prediction. In: Proc. 13th Int Conf on Principles of Knowledge Representation and Reasoning (KR'12), Roma, June 10–14, 2012. Palo Alto, CA: AAAI Press; 2012. pp 402–412.
24. Post E. Introduction to a general theory of elementary propositions; 1921. Reproduced in Van Heijenoort J. editor, From Frege to Gödel: A source book in mathematical logic 1879–1931. Harvard University Press; 1977. pp 264–283. First ed., 1967.
25. Prade H, Richard G. Multiple-valued logic interpretations of analogical, reverse analogical, and paralogical proportions. In: Proc 40th IEEE Int Symp on Multiple-Valued Logic (ISMVL'10), Barcelona; May 26, 2010. pp 258–264.
26. Prade H, Richard G. A short introduction to computational trends in analogical reasoning. In: Prade H, Richard G, editors.Computational approaches to analogical reasoning: Current trends. Berlin: Springer; 2014. pp 1–22.
27. Evans Th, G. A heuristic program to solve geometry-analogy problems. In: Proc AFIP Spring Joint Computer Conf.; 1964. Vol. 25, pp 5–16.

28. Davies TR., Russell SJ. A logical approach to reasoning by analogy. In: Proc 10th Int Joint Conf on Artificial Intelligence (IJCAI'87), Milan: Morgan Kauffman, August 23–28, 1987. pp 264–270.
29. Gust H, Kühnberger KU, Schmid U. Metaphors and heuristic-driven theory projection (HDTP). Theor Comput Sci 2006;354(1):98–117.
30. Schmidt M, Krumnack U, Gust H, Kühnberger K-U. Heuristic-driven theory projection: an overview. In PradeRichard G, editors. computational approaches to analogical reasoning—current trends. Berlin: Springer; 2014. pp 163–194.
31. Gentner D. The mechanisms of analogical learning. In: Vosniadou S, Ortony A, editors. Similarity and analogical reasoning. New York: Cambridge University Press; 1989. pp 197–241.
32. Winston PH. Learning and reasoning by analogy. Commun ACM 1980;23(12):689–703.
33. Hofstadter D, Mitchell M. The Copycat project: a model of mental fluidity and analogy-making. In Hofstadter D and The Fluid Analogies Research Group, editors. Fluid Concepts and creative analogies: Computer models of the fundamental mechanisms of thought. New York: Basic Books; 1995. pp 205–267.
34. Holyoak K, Thagard P. Analogical mapping by constraint satisfaction. Cogn Sci 1989;13:295–355.
35. Hummel JE, Holyoak KJ. Distributed representations of structure: a theory of analogical access and mapping. Psychol Rev 1997;104(3):427–466.
36. Hernández-Orallo J, Martinez-Plumed F, Schmid U, Siebers M, Lowe DL. Computer models solving intelligence test problems: progress and implications. Artif Intell 2016;230:74–107.
37. Gentner D. Structure-mapping: a theoretical framework for analogy. Cogn Sci 1983;7(2):155–170.
38. Falkenhainer B, Forbus K, Gentner D. The structure-mapping engine: algorithm and examples. Artif Intell 1989;41(1):1–63.
39. Forbus K, Usher J, Lovett A, Lockwood K, Wetzel J. CogSketch: sketch understanding for cognitive science research and for education. Top Cogn Sci 2011;3(4):648–666.
40. Strannegård C, Cirillo S, Ström V. An anthropomorphic method for progressive matrix problems. Cogn Syst Res 2013;22–23:35–46.
41. Kunda M, McGreggor K, Goel A. A computational model for solving problems from the Raven's progressive matrices intelligence test using iconic visual representations. Cogn Syst Res 2013;22–23:47–66.
42. Tversky A. Features of similarity. Psychol Rev 1977;84:327–352.
43. Kunda M, McGreggor K, Goel AK. Taking a look (literally!) at the Raven's intelligence test: Two visual solution strategies. In: Proc 32nd Annual Meeting of the Cognitive Science Society, Portland, OR; 2010. pp 1691–1696.
44. Ragni M, Neubert S. Solving Raven's IQ-tests: an AI and cognitive modeling approach. In Proc 20th Eur Conf on Artificial Intelligence (ECAI'12), Montpellier, France, Aug. 27–31; 2012. pp 666–671.
45. Ragni M, Neubert S. Analyzing Raven's intelligence tests: cognitive model, demand, and complexity. In Prade H, Richard G, editors, Computational approaches to analogical reasoning—current trends, Berlin: Springer; 2014. pp 351–370.
46. Anderson J, Lebiere C. The atomic components of thought. Hillsdale, NJ: Lawrence Erlbaum Associates; 1993.
47. Stahl P, Ragni M. Complexity in analogy tasks: an analysis and computational model. In: Proc 19th Eur Conf on Artificial Intelligence (ECAI'10), Amsterdam: IOS Press; 2010. pp 1041–1042.
48. Schockaert S., Prade H. Completing rule bases using analogical proportions. In Prade H, Richard G, editors, Computational approaches to analogical reasoning - current trends. Berlin: Springer; 2014. pp 195–215.
49. Bounhas M., Prade H., Richard G. Analogical classification: A new way to deal with examples. In Schaub T, Friedrich G, and O'Sullivan B, editors. In: Proc 21st Eur Conf on Artificial Intelligence (ECAI'14), Vol. 263 of Frontiers in artificial intelligence and applications. Amsterdam, The Netherlands: IOS Press; 2014. p 135–140.

50. Wang H-Y, Yang Q. Transfer learning by structural analogy. In Burgard W, Roth D, editors. Proc. 25th AAAI Conf on Artificial Intelligence (AAAI'11), Aug. 7–11, 2011. San Francisco: AAAI Press; Menlo Park, CA. 2011. pp 513–518

# A APPENDIX

## A.1 The Second Case Mentioned in Section 4.3

In Figure A1, we exhibit the second case mentioned in section 4.3 which can be solved by using Post algebra. We only need two attributes, the first one with values *losange*, *star*, *pentagone*, and the second one with values *sprayed*, *empty*, and *filled*. We consider the following order:

$a_1$ : losange $\quad a_2$ : star $\quad a_3$ : pentagon

$b_1$ : sprayed $\quad b_2$ : empty $\quad b_3$ : filled

This leads to the encoding shown in A2. Using the equation-solving process, we get the encoding $a_3b_2$ (Figure A3), which is the encoding of the correct answer: an empty pentagon.

## A.2 Boolean Encoding of the Test of Figure 22

Back to the above-mentioned test in section 4.4, and exhibited in Figure 22, let us investigate a Boolean coding using the features *circle*, *triangle*, *square*,

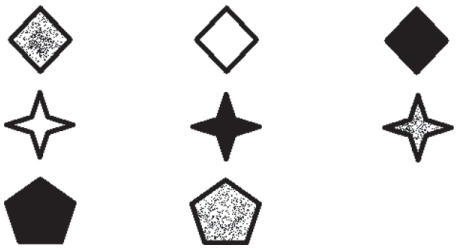


**Figure A1.** Modified Raven test 17.

| | | |
|---|---|---|
| $a_1b_1$ | $a_1b_2$ | $a_1b_3$ |
| $a_2b_2$ | $a_2b_3$ | $a_2b_1$ |
| $a_3b_3$ | $a_3b_1$ | x y |

**Figure A2.** Encoding with Post algebra.

**Figure A3.** Modified Raven test 17: actual solution.

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 11100 | 00011 | 11100 |
| 2 | 11100 | 11100 | 00011 |
| 3 | 00011 | 11100 | ????? |

**Figure A4.** A partial Boolean encoding of the test exhibited in Figure 22.

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 100 010 001 000 000 | 000 000 000 101 010 | 001 010 100 000 000 |
| 2 | 100 001 010 000 000 | 010 001 100 000 000 | 000 000 000 110 001 |
| 3 | 100 000 000 011 100 | 010 100 001 000 000 | ??? ??? ??? ??? ??? |

**Figure A5.** A complete Boolean encoding of the test exhibited in Figure 22.

*star*, and *hexagon* (where, for instance, *circle* get the value 1 if there is a circle in the picture, 0 otherwise). Each picture is then encoded with a sequence of five Boolean values: the complete encoding is shown in Figure A4. Using ANARAVEN algorithm, we get the solution 11100 which indicates that the answer should contain a circle, a triangle, and a square, but we do not know in what particular order. To try to get the exact solution, we have to define a more accurate Boolean encoding, taking into account the position of the corresponding figure in a picture. As we have three candidate positions, a total of $3 \times 5 = 15$ attributes is necessary, denoted $a_1, \ldots, a_{15}$ as below:

$a_1$ : circlePosition1 $a_2$ : circlePosition2 $a_3$ : circlePosition3

$a_4$ : trianglePosition1 $a_5$ : trianglePosition2 $a_6$ : trianglePosition3

$a_7$ : squarePosition1 $a_8$ : squarePosition2 $a_9$ : squarePosition3

$a_{10}$ : starPosition1 $a_{11}$ : starPosition2 $a_{12}$ : starPosition3

$a_{13}$ : hexagonPosition1 $a_{14}$ : hexagonPosition2 $a_{15}$ : hexagonPosition3

This leads us to the encoding shown in Figure A5. Looking for a solution for attribute $a_1$, we are led to complete the vertical pattern 00, which has already been observed in the previous column, leading to the unique solution 0. This agrees with an horizontal reading where the pattern to be completed is 10: This pattern appears in the previous row giving the unique solution 0 (no contradiction with the previous vertical reading). The same reasoning applies for $a_2$ since we have to complete the pattern 00 vertically and the pattern 01 horizontally: both of them have previously been observed leading to the unique solution 0.

When it comes to $a_3$, the corresponding vertical pattern is 10 not previously seen, and the horizontal one is 00 which has been observed on both the previous row, leading to 1 with the first row and 0 with the second row: this is a typical case where

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 100 010 001 | 010 000 001 | 001 010 000 |
| 2 | 100 001 010 | 010 001 000 | 001 000 010 |
| 3 | 100 000 000 | 010 100 000 | ??? ??? ??? |

**Figure A6.** A Boolean encoding of the modified Raven test 32.

| | | |
|---|---|---|
| $a_1b_2c_3$ | $a_4b_1c_3$ | $a_4b_2\ c_1$ |
| $a_1b_3c_2$ | $a_4b_1c_2$ | $a_4b_3c_1$ |
| $a_1b_4c_4$ | $a_2b_1c_4$ | x y z |

**Figure A7.** Encoding with Post algebra.

we have no answer. This situation still occurs for other attributes and ANARAVEN does not provide any global solution to this test.

### A.3 The First Case Mentioned in Section 4.4

Figure A8 exhibits the Raven test mentioned in the first paragraph of Section 4.4 for which ANARAVEN fails to find a solution. Starting from the Boolean encoding corresponding to the nine features below:

$$A\_Position1,\ A\_Position2,\ A\_Position3,$$
$$B\_Position1,\ B\_Position2,\ B\_Position3,$$
$$C\_Position1,\ C\_Position2,\ C\_Position3$$

We get the encoding given in Figure A6.

It is clear that attribute $B\_Position1$ (feature 4) leads to a contradiction (because of the contradictory vertical pattern $(0, 0)$ for feature 4) thus preventing ANARAVEN to give a solution.

Let us move to a Post algebra encoding using three attributes, one per position, with the following order:

$a_1 : A\_Position1 \quad a_2 : B\_Position1 \quad a_3 : C\_Position1 \quad a_4 : rien\_Position1$

$b_1 : A\_Position2 \quad b_2 : B\_Position2 \quad b_3 : C\_Position2 \quad a_4 : rien\_Position2$

$c_1 : A\_Position3 \quad c_2 : B\_Position3 \quad c_3 : C\_Position3 \quad c_4 : rien\_Position3$

The encoding of this test using Post Algebra is exhibited in Figure A7. It is easy to get the two first features by applying the method: we get $x = a_2$ ($B$ in the first cell) and $y = b_4$ (a blank in the second cell). Unfortunately, no solution can

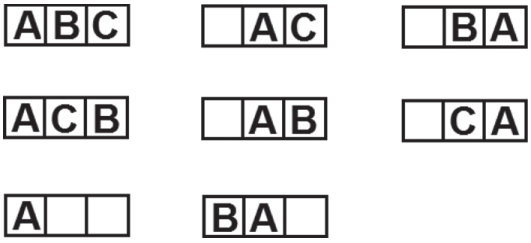



**Figure A8.** Modified Raven test 32.

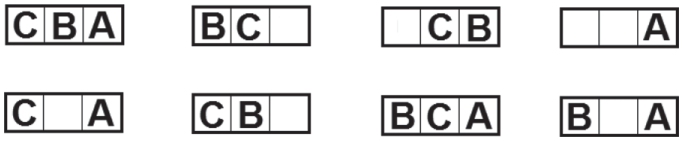



**Figure A9.** Candidate solutions for the test.

be found for the third feature. However, it can be seen that, among the candidate solutions reproduced in Figure A9 (remember that in a true Raven's tests, candidate solutions are proposed to the attendee), only one start with *B* and then a blank, which here will enable to reach the solution.