*ECAI 2012*
*Luc De Raedt et al. (Eds.)*

*doi:10.3233/978-1-61499-098-7-666*

# Solving Raven's IQ-tests: An AI and Cognitive Modeling Approach

**Marco Ragni** and **Stefanie Neubert** [1]

**Abstract.** Human reasoners have an impressive ability to solve analogical reasoning problems and they still outperform computational systems. Analogical reasoning is relevant in dealing with intelligence tests. There are two kinds of approaches: to solve IQ-test problems in a way similar to humans (i.e., a cognitive approach) or to solve these problems optimally (i.e., the AI approach). Most systems can be associated with one of these approaches. Detailed systems solving geometrical intelligence tests, explaining cognitive operations based on working memory and producing precise predictions and results such as error rates and response times have not been developed so far. We present a system implemented in the cognitive architecture ACT-R, able to solve analogously developed problems of Raven's Standard and Advanced Progressive Matrices. The model solves 66 of the 72 tested problems of both tests. The model's predicted error rates correlate to human performance with $r = .8$ for the Advanced Progressive Matrices and $r = .7$ for all problems together.

## 1 Introduction

Geometrical intelligence tests such as Raven's Progressive Matrices require analogical reasoning. Raven's intelligence test consists of a $3 \times 3$ matrix and eight possible solutions. Each but the last cell in this matrix contains differently arranged geometrical objects. Consider the following problem (cp. Figure 1):

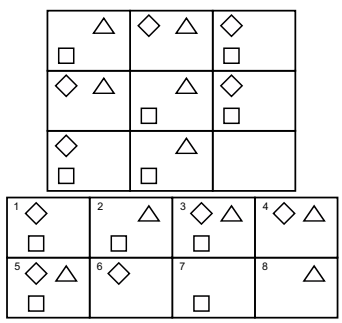


**Figure 1.** A geometrical reasoning problem. The solution is no. 4.

This is an example of one kind of Raven's Progressive Matrices, more examples can be found in Figure 3. The task of the reasoner is to identify an implicitly given geometrical function and apply this function to derive a solution. There are two kinds of tests with similar problems differing in their reasoning difficulty. On the one hand, there is the Standard Progressive Matrices test (short: SPM). This test is designed to measure average intelligence. It consists of five sets each with twelve problems. The reasoning difficulty of the problems increases within a set; the difficulty of the sets altogether is also increasing [8, p. 37]. Gifted adults are tested by the Advanced Progressive Matrices test (short: APM). It consists of two sets with 12 training problems and 36 test problems. Human reasoning performance decreases almost monotonically. Hence, the first problems can be solved by almost all participants, whereby the last problem is answered incorrectly by most human reasoners [9, p. 33].

One of the first computer programs to solve simple geometrical analogy problems was developed by Evans [5, p. 271]. The problems solved were only one-dimensional and hard coded. Recently, Cirillo and Ström [4] developed a computational program, that is able to solve 28 out of the 36 problems of the SPM of the subsets C through E by computing the solutions without considering the given possible solutions [4, p. 32]. Their program does not model arbitrary geometrical problems and they excluded all aspects of human problem solving. Carpenter and colleagues [3] analyzed rules necessary to solve the APM and implemented this model – called BETTER-AVEN – in the architecture 3-CAPS [3]. This model tries to solve the problems like an above average student by managing a larger set of goals in working memory as opposed to another previous model (FAIRAVEN). Lovett and colleagues [12] combined CogSketch [7] (for the visual analysis) and the Structure Mapping Engine [6] (for the computation of the analogies) to a model which is able to solve Evans' problems but additionally includes some assumptions about human cognitive processes [12, p. 1194]. Consequently they extended their model to solve most problems of the SPM (Sets B to E) [11]. Problems the program could not solve were considered difficult for humans [11, p. 2765]. An interesting approach combining a symbolic approach with probabilistic components is Hofstadter and Mitchell's Copycat [13, 10]. It has a representation for long-term memory (the slipnet) and working memory (workspace) and uses domain dependent rules to identify patterns. These rules do not always fire and can show a non-deterministic behavior. It has so far not been applied to geometrical reasoning problems but only for reasoning with letters. We will see, however, that ACT-R 6.0 offers – to some extend – as well some probabilistic behavior.

Taken together, there are programs that try to solve the SPM problems in an AI-fashion [5, 4] and programs which solve them similarly to humans [11, 3]. None of these approaches have been applied to APM and SPM problems at the same time and none of these approaches uses working memory assumptions, makes response time predictions or is generalizable to arbitrary analogical problems (probably besides [11, 13]). To overcome this problem and to have a program which is able to solve the problems in general

[1] University of Freiburg, Germany, email: {ragni, neuberts}@informatik.uni-freiburg.de

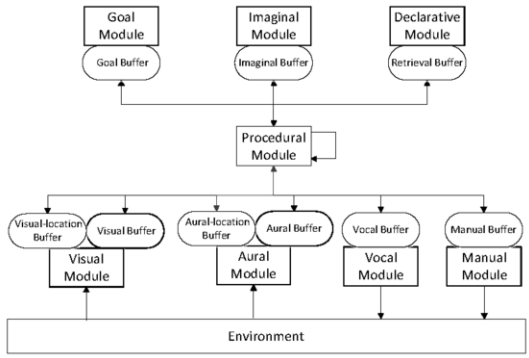


**Figure 2.** The cognitive architecture ACT-R [1]. The architecture assumes different modules and buffers, which have precise defined roles.

and is at the same time cognitively more adequate, we decided to use the cognitive architecture ACT-R [1, 2]. ACT-R is a production rule system (which is Turing complete) and consists of a symbolic layer and a sub symbolic layer. A model/program can be specified by production rules. Information knowledge is represented by so-called chunks. These chunks are n-tuples coding the information about the objects and their relative position. ACT-R assumes a modular structure of the mind. Each module is responsible for specific kinds of information and there are buffers attached to each module. The *visual* deals with perceiving objects; the *goal* is for general control issues; the *imaginal* is for problem specific representations, and the *declarative* stores declarative knowledge. More information can be found in [1, 2].

The remainder of this article is structured as follows: In the next section, we will briefly introduce the cognitive and AI model implemented in the cognitive architecture ACT-R and the rules, which are able to solve such problems. An evaluation and discussion concludes the article.

## 2 Cognitive Model

A typical geometrical analogical problem consists of a matrix and a solution. Each cell contains geometric objects which can be classified by the following attributes: shape (e.g., triangle, circle, etc.), size (e.g., represented numerically 1, 2, 3), number of sides (e.g., a triangle has 3 sides), width, height (for objects which are not of the same height and width, e.g., a rectangle), color (e.g., black, white etc.), texture, line art (e.g., solid, dotted), rotation (e.g., 0 degree), position (inside a cell), and quantity. Consider the triangles of Figure 1. The shape is *triangle*, size 1, number of sides 3, width 1, height 1, color *white*, texture *white*, line art *solid*, rotation 0, position 2 and quantity 1.

To compare attribute values, the model saves the name of the attribute as well as three values in a chunk. If, for example, triangles with a rotation of 0, 45, and 90 degrees exist in the first row, the model creates a chunk of the following type: *(rotation, 0, 45, 90)* representing the change in the triangles across the row.

Some geometrical intelligence test problems have an increasing number of objects in a row, e.g., one triangle in the first cell and three in the second cell, in this case the model must conclude that it is likely that the next cell will contain 5 triangles. Such arithmetical knowledge is relevant to solve such problems. Consequently, we added different number series of two number consisting of all single-digit to the model's declarative memory. This is the standard way to use fact knowledge, which has in the case for small numbers not to be calculated anymore [1]. Each number series encodes if the series increases or decreases and the function. Declarative memory contains addition facts with two addends and a sum as well. With the number series, the model is able to validate attribute values when applying *quantitative pairwise progression* (see below). So if a row contains objects with the sizes 1, 2 and 3, the model calls the number series (1, 2) and (2, 3), which both contain the direction *increasing* and the factor $+1$.

With the addition facts, the model is able to check the objects' quantity when applying the rule *figure addition*. If a row contains two, one and three objects, the model calls the addition fact (2 + 1 = 3) to validate the assumed rule. The objects are grouped according to the *matching names heuristic* [3, p. 417]. Hence, objects are grouped according to their names appearing in the shape attribute. This grouping of objects is given with the encoding of the problem.

### 2.1 Identified Rules and Model Application

To solve the problems of the Progressive Matrices, various relational rules describing the relations of the objects in a row or column are necessary. Carpenter and colleagues [3] and Cirillo and Ström [4] identified five rules which are sufficient to solve almost all problems of the Progressive Matrices [3, p. 408]. We briefly introduce the rules:

**Rule 1: Constant in a row / Identity.** If all attributes of the objects in a row are identical, then the rule *constant in a row* can be applied. This rule inserts the identical object of the previous cell into the final cell. The objects can of course differ among the rows.

The model checks if the objects differ in some attributes by means of a pairwise comparison of objects within each row. The solution is constructed by taking the attribute values of the previously considered object.

**Rule 2: Distribution of three values / Distribution of three entities.** If an attribute of the objects in a row differs and the same values of this attribute occur in every row, then the rule *distribution of three values* can be applied. In Figure 3a the attribute *texture* differs. The attribute values occurring in each row are *black*, *striped* and *white*.

The model stores the attribute name and the existing values in declarative memory. In the other cells the objects' attribute values are checked for concordance with the first row. The order in which the objects appear is not relevant. In problem 3a, the attribute name *texture* and the values *white*, *black* and *striped* are stored. In the second and the third row the appearing texture values are compared to the stored values. The model then retrieves the stored values. The value that has not appeared in the third row is then entered into the solution. In the depicted example, a square with the missing texture value *black* is created. In Figure 4 the model's processing of a problem requiring *distribution of three values* is depicted.

**Rule 3: Quantitative pairwise progression / Numeric Progression.** If the values of an attribute are increasing or decreasing continuously in a row then the rule *quantitative pairwise progression* can be applied. This rule is used in Figure 3b, where the objects' rotation increases by 45 degrees in each cell.

Once the model has associated two objects in a row, the model retrieves a number series containing the changes of the attribute values. The model then stores the information about the increase or decrease

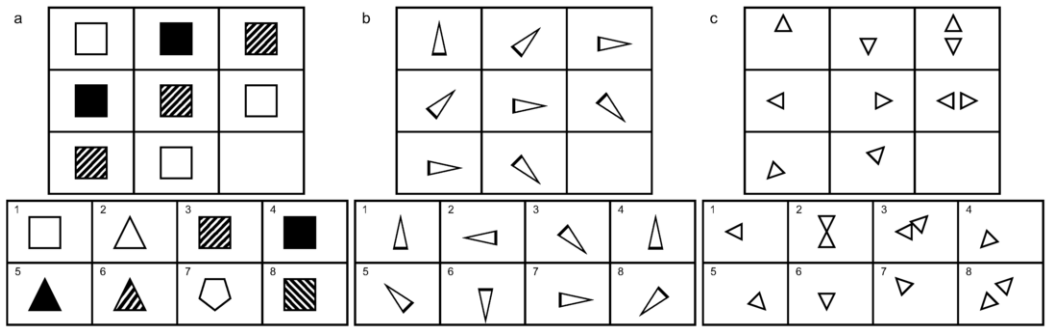


**Figure 3.** Problems illustrating the different rules. In 3a the rule *distribution of three values* is required. The solution is no. 4. Problem 3b requires the rule *quantitative pairwise progression*. The solution is no. 6. In Problem 3c the rule *figure addition* is required. The solution is no. 8.

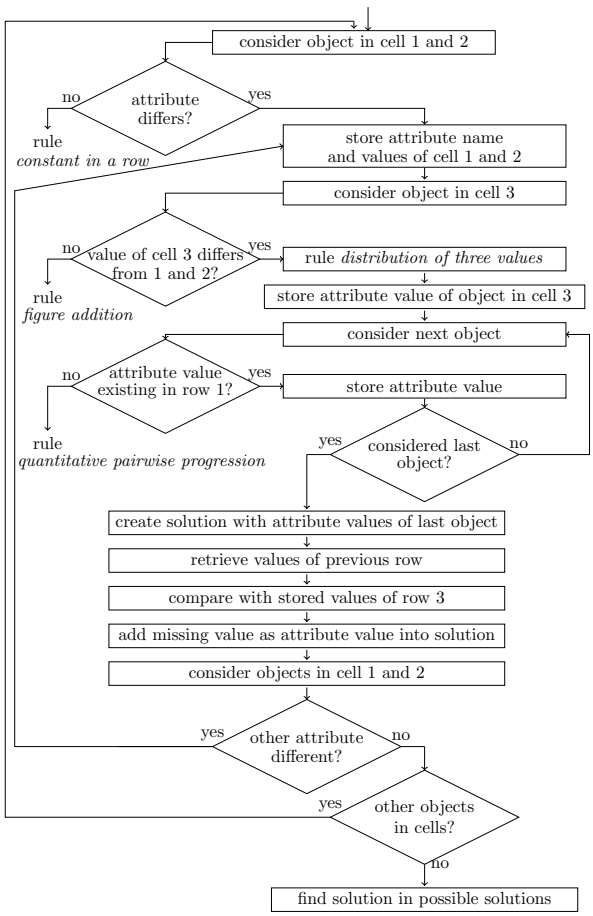


**Figure 4.** The model's processing of a problem requiring the rule *distribution of three values*.

of the values and the functional change. Again, for the associated objects in the second and the third cell of this row a matching number series is retrieved. This procedure is repeated in every row. Problem 3b gives an example. It first requires a call of a number series according to the first two associated objects' rotation with the values 0 and 45, the direction *increasing* and the function is $+45$ degrees. In the following, a number series with direction *increasing* and $+45$ is retrieved. This contains the values 45 and 90 degrees. To construct the solution, a number series is retrieved from declarative memory. This series contains 135 as the first value, referring to the object's rotation in the matrix cell number seven. The second value is 180, which is identified as the solution.

**Rule 4: Figure addition / Binary OR.** If all objects of the third column appear in their respective rows in the first two columns, the rule *figure addition* can be applied. An example is depicted in Figure 3c.

There are two possibilities to apply the rule *figure addition* for the model. On the one hand, the considered objects can be identical except of their quantities. The quantity of the last object results from the addition of the first two objects' quantities. The model tries to retrieve an addition fact that contains the first two objects' quantities as addends and the quantity of the last object as sum.

On the other hand, if additional object attributes differ, the model checks for each object, if this object is in the third cell of the current row. No addition facts have to be retrieved. For each object in the last row, a separate solution is created. This type of rule is required to solve problem 3c. Accordingly, the rule *figure subtraction* can be applied if the objects in a column are subtracted from the objects in another column to get the objects of the third column.

**Rule 5: Distribution of two values / Binary XOR.** The rule *distribution of two values* is applied if there are exactly two equal values and one differing value of an attribute in a row. This rule is necessary to solve the problem in Figure 1. There are exactly two associated objects in a row, the third object is null.

For the model, there are also two possibilities to apply this rule. On the one hand, two identical objects are in a row, therefore one of the cells is empty. If in the last row a single object appears, the model constructs a solution identical to this object. If there are two objects, no solution is created. The problem in Figure 1 depicts this type of the rule. In each row, there are exactly two diamonds, triangles and squares.

If there are two equal and one differing attribute values in a row, the model also assumes *distribution of two values*. The existing attribute values are stored. In the second and third row the model checks, whether the same values appear once or twice. To construct the solution, the missing value is identified using the values in the third row and the stored values. This kind of rule is required in problem 5a. The texture values in each row are *black* twice and *white* once. In the third row, a black and a white object appear, hence the missing texture value is *black*.

The description of the rules focuses on rows, which is sufficient

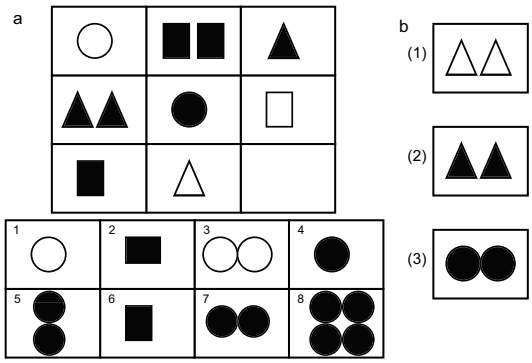


**Figure 5.** The model processes 5a several times. It considers, analogously to humans, one attribute at a time: first the quantity (1), the texture (2), then the number of sides (3), and there is no predefined order. The steps are depicted in Figure 5b. The solution is no. 7.

to solve the Standard and Advanced Progressive Matrices. Hence, the application to columns is not implemented in this model. For some problems several rules can be applied. For example, if there are three objects in the first cell, two objects in the second cell and one object in the third cell, both rule *quantitative pairwise progression* with a decreasing quantity and rule *distribution of three values* with the quantity values 3, 2 and 1 can be applied.

## 2.2 Procedure of solving a problem

The cognitive model initially chooses a first object in the first cell. All objects that share the same shape with this object are considered first. All other remaining objects are ignored. Then, the model considers the associated object in the second cell, which is then compared to the first object. If an attribute differs then the attribute name and the two existing values are stored. Next, the third object is compared to the second object and its attribute value is also stored. The stored values are interpreted as a rule now. In the following, the model analyses the second row as it did the first row, but the attribute values are compared to the stored values to find possible discrepancies and another rule if required. The two objects in the third row are analyzed as in the previous rows. In addition, the missing attribute value is called from declarative memory to construct the solution. If further attributes differ in the current group of objects, the procedure is repeated. To construct the new solution, the according attribute value is adapted in the existing solution. When all remaining attributes are equal, possible further objects, ignored up to now, are analyzed as described. When no more objects are left, the model moves on to find the correct solution.

## 2.3 Processing an example

We demonstrate the developed model on the example depicted in Figure 5a. This example requires the application of the rule *distribution of three values* and *distribution of two values*. The model first identifies one differing attribute, namely *quantity*. The quantity values of the third and the first cells are identical, wherefore *distribution of three values* and *quantitative pairwise progression* cannot be applied. The model assumes *figure addition*, consequently in the third cell there should be two objects with quantities 1 and 2. There is no second object in the cell, however – this marks a conflict. The model now assumes *distribution of two values*. The values occurring in the

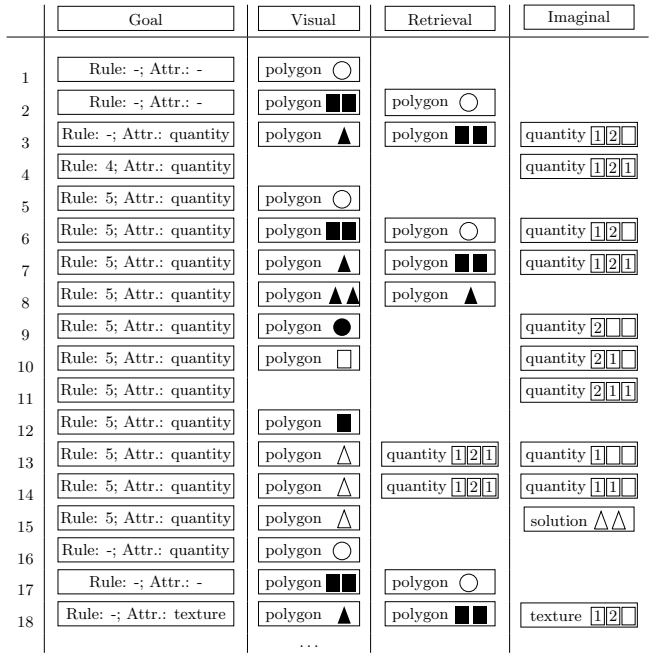


**Figure 6.** Contents of the buffers when processing problem 5a. The model first considers an object in the first cell, which appears in the visual buffer. When considering the object in the second cell, the model retrieves the first cell's object from the retrieval buffer. In the goal buffer, the model stores the differing attribute (*quantity*). In the imaginal buffer, a chunk is created containing the values of the first row. First, the model assumes rule 4 (*figure addition*). In conflict to rule 4, no second object exists in cell three. Hence, the model assumes rule 5 (*distribution of two values*). Ensuing, the model considers the second row (line 8 - 11), storing the values in the imaginal buffer again. To construct the temporary solution, one of the chunks is retrieved (line 13). The procedure described is repeated for the attributes *texture* and *number of sides*.

first row (1, 2 and 1) are stored, as well as those in the second and the third rows. To identify the missing value, the previously stored values are retrieved. The solution is constructed with the missing quantity value 2. The resulting solution is depicted in Figure 5b (1).

The model considers another differing attribute now, *texture*. The missing value, *black*, is identified as described above. The existing solution is retrieved and adapted by entering the identified texture value. Figure 5b (2) shows the resulting solution.

The last differing attribute is *number of sides*. The attribute values of the first row (1, 4 and 3) are stored. The model assumes *distribution of three values*, since all values are different. In the second row, the values are also stored and checked for occurrence in the first row. In the third row, the model retrieves the stored values to identify the missing value 1, which is entered into the existing solution. Figure 5b (3) shows the resulting solution. Since all remaining attributes are equal and no more objects are existing in the cells, the model moves on to find the solution. The buffer contents are shown in Figure 6.

**Analysis of complexity.** The complexity of each problem depends on the type of the rule needed to solve the problem, the number of required rules and the difficulty of associating objects covered by one rule.

The order of the rules mentioned above corresponds to the prioritized order of the subjects according to Carpenter et al. This was

identified in various signs, like the simplicity of *constant in a row*, the preference of *figure addition* over *distribution of two values* and the preference of *quantitative pairwise progression* over *distribution of two values* [3, p. 421]. Therefore, BETTERAVEN tests the rules in the order depicted above, except for *quantitative pairwise progression*, which is tested before *distribution of three values* [3, p. 421]. The exact reason for this preference is not explained, however. The model prefers *distribution of three values*, because this rule requires less calls to the declarative memory and hence is simpler to apply. In addition, if the model tested *quantitative pairwise progression* before *distribution of three values*, a lower correlation with the subjects' correctness as well as a lower correlation of the model's response times with its error rates is achieved. The error rates of the subjects also depend on the number of rules required [3, p. 411]. Thus, a problem that most solve correctly requires only one rule, but a problem difficult for the subjects requires the application of up to eight rules.

If the cells of a problem contain more than one object, another complexity is to associate the objects covered by one rule. In most of the problems, the objects are associated clearly. However, in the Advanced Progressive Matrices there are problems in which the association of the objects may be difficult. The grouping of the objects follows the *matching names heuristic* [3, p. 417], whereby the objects are grouped according to their names. The objects' names are determined in their shape attribute. Hence, in Problem 1 squares as well as diamonds and triangles are grouped.

## 3 The Evaluation

The cognitive model was tested on different sets of problems which are functionally equivalent to the Set II of the Advanced Progressive Matrices and Sets C through E of the Standard Progressive Matrices. The empirical data was collected from Heller et al. [8, 9].

### 3.1 Cognitive Approach

The conclusions described below correspond to the best achievable correlation to the Standard and Advanced Progressive Matrices with individual parameters. The correlation was computed using Pearson's correlation coefficient.

**Standard Progressive Matrices.** The model is able to solve 35 out of the 36 problems. The correlation between the percentage of problems solved correctly by subjects and by the model is $r = .5$. This correlation is achieved with the parameter values listed in Table 1, column *SPM*. Towards the standard values of ACT-R, learned chunks are forgotten faster, but the point where a chunk is forgotten, is later. As a consequence the chunks learned early on are more difficult to retrieve than the most recent.

**Table 1.** Parameter values of the best correlations with the different IQ-tests and the parameter setting to solve all problems correctly.

| | Tested range | Total | SPM | APM | Maximal correctness |
|---|---|---|---|---|---|
| Noise | [0.3, 0.5] | 0.5 | 0.5 | 0.4 | 0.0 |
| Retrieval threshold | $[-0.3, -1]$ | $-0.6$ | $-0.6$ | $-0.6$ | $-1$ |
| Decay parameter | [0.9, 0.99] | 0.94 | 0.97 | 0.9 | 0.95 |
| Pearson's Correlation $r$ | | **.70** | **.5** | **.8** | |

**Advanced Progressive Matrices.** The model solves 31 out of the 36 problems of the Advanced Progressive Matrices Set II. We get a correlation of $r = .8$ between the subjects' data and the parameter values listed in Table 1, column *APM*. Only for the three hardest problems, does the visual-num-finsts parameter have to be set to 32 (this means that the model is able to recognize 32 objects as considered), with a standard value of 4. This relatively high value can be associated with some kind of reasoning difficulty: the more different objects have to be kept in working memory the more difficult a problem is.

The first problems can be solved by the model, as by the subjects, with few errors. The reasoning difficulty associated with a higher error rate increases with the increasing number of the problem. High deviation at some problems can be explained by the fact that the difficulty of this problems consists of the difficulty in grouping the objects covered by one rule, because the objects look similar. The application of the rule at the group of objects is comparatively simple or fewer rules are required. Because the grouping is pretended, this type of problems is easier for the model than for subjects.

#### 3.1.1 Response Times

The model requires 13.03 seconds, on average, to solve a problem, with a standard deviation of 3.75 seconds. A correlation of $r = .54$ between the model's response time and its error rates was achieved. Hence, the model requires more time to solve a more complex problem than a simple problem. This is explained by the requirement of more rules to solve the problems as well as the greater amount of time a rule requires. In addition, more time is needed if the model first assumes the wrong rule, after which it finds a conflict and has to start again. This often occurs if the rule covers a higher number of objects, like *figure addition*.

Note that the time the model needs to find the correct solution depends considerably on the number of the solution. The reason is that it takes more time to consider each putative answer – requiring to consider every possible solution if the correct solution is the last one, but only one possible solution if it is the first one.

#### 3.1.2 Comparison of Performance to BETTERAVEN

Overall, our model solves 66 out of 72 problems, whereof 31 out of the 36 problems of the Advanced Progressive Matrices Set II. Carpenter and colleagues' model BETTERAVEN solves 7/12 problems of Set I and 25/36 of Set II of the Advanced Progressive Matrices [3]. Two of the tested problems can not be solved by BETTERAVEN, as these problems do not adhere to one of the five rules identified by Carpenter and colleagues [3, p. 421]. The performance of an additional 9 problems were not reported. Seven out of these 9 problems are solved by our model.

In contrast to BETTERAVEN our model produces error rates. This allows for conclusions about the cognitive reasoning difficulty of the respective problem. The correlation between the model's and the subjects' error rates is $r = .7$. Further, the model produces response times that are higher for difficult and lower for simple problems. The predicted response times correlate with $r = .54$ with the error rates.

#### 3.1.3 Performance at both the Standard and Advanced Progressive Matrices

The highest overall correlation with identical parameter settings is $r = .7$ for all problems of the Standard and Advanced Progressive Matrices together. The respective parameter values are listed in column *Total* in Table 1. Compared to the parameter values achieving

the highest correlation at the Standard Progressive Matrices, the parameters induce faster forgetting (i.e., the chunk activation falls below the retrieval threshold). This is explained by the fact that the solutions created by the model can be retrieved with greater difficulty. Focusing on the parameter values achieving the highest correlation at the Advanced Progressive Matrices, forgetting knowledge is slowed down. Hence, the model forgets created solutions more slowly and is able to retrieve them especially in complex tasks more often. Hence, the model is more accurate in solving the problems. Since the model is not forced to consider the cells' objects in a given order, the performance is decreased in problems where the order of the considered objects is important. Testing conflicting rules first can lead to the neglect of previously created correct solutions, which cannot be remembered in the solution finding process, as the creation occurred too long ago.

## 3.2 AI Approach

If we eliminate several human-like features such as the ability to forget identified patterns, eliminate reinforcement learning and the calculation of utilities for different production rules we allow a deterministic processing of the model. This in turn can be used to test how many problems can be solved overall. Different production rules can be still in conflict. The parameter values to achieve the maximal performance is the *visual-num-finsts parameter* of 32 and the parameter values listed in Table 1, column *Maximal correctness*. Hence, the model is able to solve 90.7% of all tested Matrices problems.

# 4 Conclusion

Intelligence tests such as Raven's Progressive Matrices still pose problems for computational systems and cognitive theories. Our motivation is to develop a system that is able to solve geometrical reasoning problems both from a formal, i.e., to solve as many problems as possible, and a cognitive side, i.e., to solve them like humans do – reproducing their errors and explaining why some problems can be solved more easily than others. There are several reasons for this: First, ambiguous situations which call for several production rules may lead to the application of the wrong rules. Second, a reasoner has to keep track of all the identified patterns. This is reflected in ACT-R by the *visual-num-finst* parameter. It tells ACT-R how many items the model can keep track of. After that many items are attended, the model forgets if the first attended object already was attended. Hence, the model considers this object as unattended. For problems containing many objects, the *visual-num-finst* parameter has to be set to a value much higher than the standard value given by ACT-R. For instance, to solve the last APM problem, a *visual-num-finst* parameter value of 32 is required, whereas the standard value is 4. This change of the default parameter is necessary as otherwise the model considers objects which were already investigated. Changing this parameter value can be avoided by storing the x-coordinate and pretending a fixed order in which the objects are considered. This forces the model, however, to consider the objects in the first cell from left to right, thus, no random order is possible anymore.

Six of the problems cannot be solved by the model. To solve four of them, the model has to take account of the solution possibilities and choose the one matching best the self created solution. The model currently constructs an exact solution and is not able to solve a problem where creating an exact solution is not possible. According to cognitive science, it is apparent that subjects take into account possible solutions and choose the one matching the self created solution best. Two further problems can be solved with two additional rules. On the one hand, particular attributes of the first and second column are used to construct the object of the third column. For example, the shape of the first column as well as the rotation of the second column yields the object in the third column. On the other hand, the quantity of the objects is added, where objects of one type count positive and objects of another type count negative. However, each of the two additional rules would be applied on a single specific problem only. We wanted to present a model which solves the Progressive Matrices using general rules. Thus, we did not implement rules which only match a single specific problem.

Overall, the model is able to solve 66 of the 72 problems. The error rates of the single problems provided by ACT-R are similar to human data, hence problems which are complex for the subjects are also complex for the model. The predicted response times correlate with the subjects' error rates. Again, solving complex problems requires more time for the model. The problems on which the model failed, can be solved by implementing two additional rules and taking into account the possible responses and choosing the one best matching the created solution. Further, the model is able to simulate an above-average intelligent subject who is able to solve 90.7% of the problems. Future work will investigate if fluid concepts (as conceptualized in Copycat [13]) can be used to solve geometrical analogy problems and how to extend ACT-R 6.0 with such concepts.

# REFERENCES


[1] J. R. Anderson, *How can the human mind occur in the physical universe?*, Oxford University Press, New York, 2007.

[2] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, 'An integrated theory of the mind', *Psychological Review*, **111**(4), 1036–1060, (2004).

[3] P. A. Carpenter, M. A. Just, and P. Shell, 'What One Intelligence Test Measures: A Theoretical Account of the Processing in the Raven Progressive Matrices Test', *Psychological Review*, **97**(3), 404–431, (1990).

[4] S. Cirillo and V. Ström, *An Anthropomorphic Solver for Raven's Progressive Matrices*, Master's thesis, Department of Applied Information-Technology, Chalmers University, Goeteborg, Sweden, 2010.

[5] T. G. Evans, 'A Program for the Solution of a Class of Geometric-Analogy Intelligence-Test Questions', in *Semantic Information Processing*, chapter 5, 271–351, Marvin L. Minsky, (1968).

[6] B. Falkenhainer, K. D. Forbus, and D. Gentner, 'The Structure-Mapping Engine: Algorithm and Examples', *Artificial Intelligence*, **41**, 1–63, (1989).

[7] K. Forbus, J. Usher, A. Lovett, K. Lockwood, and J. Wetzel, 'CogSketch: Open-domain sketch understanding for cognitive science research and for education', in *Proceedings of the Fifth Eurographics Workshop on Sketch-Based Interfaces and Modeling*, eds., C. Alvarado and M.-P. Cani, (2008).

[8] K. A. Heller, H. Kratzmeier, and A. Lengfelder, *Matrizen-Test-Manual, Band 1. Ein Handbuch mit deutschen Normen zu den Standard Progressive Matrices von J. C. Raven*, Beltz Test, Göttingen, 1998.

[9] K. A. Heller, H. Kratzmeier, and A. Lengfelder, *Matrizen-Test-Manual, Band 2. Ein Handbuch mit deutschen Normen zu den Advanced Progressive Matrices von J. C. Raven*, Beltz Test, Göttingen, 1998.

[10] D.R. Hofstadter, *Fluid Concepts & Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*, Basic Books, 1995.

[11] A. Lovett, K. Forbus, and J. Usher, 'A Structure-Mapping Model of Raven's Progressive Matrices', in *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*, eds., S. Ohlsson and R. Catrambone, pp. 2761–2766. Cognitive Science Society, (2010).

[12] A. Lovett, E. Tomai, K. Forbus, and J. Usher, 'Solving Geometric Analogy Problems Through Two-Stage Analogical Mapping', *Cognitive Science*, **33**(3), 1192–1231, (2009).

[13] M. Mitchell, *Copycat: A Computer Model Of High-Level Perception And Conceptual Slippage In Analogy-Making*, Ph.D. dissertation, Computer Science Department, Indiana University, Bloomington, 1990.