Comparing Computer Models Solving Number Series Problems

Ute Schmid[1] and Marco Ragni[2]

[1] Cognitive Systems Group, University of Bamberg,
ute.schmid@uni-bamberg.de
[2] Foundations of AI, Technical Faculty, University of Freiburg,
ragni@cs.uni-freiburg.de

Abstract. Inductive reasoning requires to find for given instances a general rule. This makes inductive reasoning an excellent test-bed for artificial general intelligence (AGI). An example being part of many IQ-tests are number series: for a given sequence of numbers the task is to find a next "correct" successor number. Successful reasoning may require to identify regular patterns and to form a rule, an implicit underlying function that generates this number series. Number series problems can be designed along different dimensions, such as structural complexity, required mathematical background knowledge, and even insights based on a perspective switch. The aim of this paper is to give an overview of existing cognitive and computational models, their underlying algorithmic approaches and problem classes. A first empirical comparison of some of these approaches with focus on artificial neural nets and inductive programming is presented.

1 Introduction

Over the last decade, there has been growing interest in computer models solving intelligence test problems. Especially, the proposal to establish a psychometric artificial intelligence (PAI; [3, 6]) with the aim to evaluate the intelligence of an artificial cognitive system based on its performance on a set of tests of intelligence and mental abilities motivated research in this domain [2].

One of the mental abilities considered by researchers as a fundamental constituent of general intelligence is inductive reasoning [22]. A well established, culture free test in this domain is Raven Progressive Matrices (RPM; [18]) where regularities have to be identified in a two-dimensional matrix of geometrical patterns. Another problem domain is inductive reasoning with numbers. In contrast to RPM, problems are represented in one dimension, that is, as a sequence, and a certain amount of mathematical knowledge is presupposed. Number series are, for example, included in two well known intelligence test batteries, namely the IST [1] and the MIT [25]. To solve RPM as well as number series problems, one has to analyze the given components, construct a hypothesis about the regularity characterizing all components, generalize this regularity and apply it to generate a solution.

Currently, there are different proposals of computer models solving number series problems which are studied in isolation. In our opinion it would be worthwhile to compare these models to gain insight into (1) the general power of the underlying algorithmic approaches with respect to scope and efficiency, and (2) their correspondence to human cognitive processes.

In the following, we first introduce the domain of number series problems in more detail and identify characteristics to classify such problems. Afterwards, we shortly present the computer systems developed to solve number series problems within artificial intelligence and cognitive systems research. A first empirical comparison of systems concludes the article.

2 Number Series Problems

A number series can be mathematically defined by a function mapping the natural numbers into the real numbers: $f : \mathbb{N} \rightarrow \mathbb{R}$. For intelligence test problems, typically the co-domain is restricted to integers. Series used in intelligence tests are usually restricted to the four basic arithmetic operations. Furthermore, numbers are restricted to small values which allow easy mental calculations [12]. Number series problems in intelligence tests are characterized as "having a unique solution" [1]. However, in general, there do not exist unique solutions for inductive problems [11]. A more precise characterization is that it must be possible to identify a unique rule from the given pattern which captures its regularity. Whether such a rule can be found depends on the length and kind of the given sequence. In intelligence tests, often five elements of a series are given and the test person or the program has to find the next element.

A variety of number series is illustrated in Table 1. Series can be generated by applying one operation to the predecessor, resulting in a simple linear function (see E1, Table 1). There can be alternating series where a different rule applies to elements on even and odd index positions (see E2, Table 1). Series can depend on more than one predecessor, which is the case for the Fibonacci series (see E3, Table 1). Series can be composed by nesting two series (see E4, Table 1). An example for a series which has no unique solution, if only 5 elements are given is presented as E5 in Table 1: One solution can be to double the second last element and subtract 2 or, equivalently, to decrement the second last element and then double it. Alternatively, a higher order rule can explain the pattern of the first 5 elements, where 2^1 is added once, 2^2 is added twice, 2^3 is added three times, and so on. A final example presents a series containing a mathematical pattern of the index (see E6, Table 1) which has been investigated by Hofstadter [10]. This last sequence is a typical example of a problem which is simple for humans but difficult for systems based on pattern induction because it has no simple closed representation.[3] Human performance depends on the complexity of the underlying pattern but can also depend on specific background knowledge. E.g., computer science students can often easily identify Fibonacci numbers or powers of two. The examples of Table 1 show that there are simple problems for humans (such as E6) but difficult for machines and vice versa depending on their

[3] The closed representation relies on a non-primitive recursive function: $f(n) = f(n - f(n-1)) + 1$.

Table 1. Examples for number series problems. The numbers in brackets represent for the given series two possible successor sequences.

ID	Series	General Rule $f(n)$	Type
E1	1, 4, 7, 10, 13, 16, 19, 22, ...	$= f(n-1) + 3,\ ,\ f(1) = 1$	linear
E2	2, 4, 3, 5, 4, 6, 5, 7, ...	$= if(even(n), f(n-1) + 2, f(n-1) - 1)$	alternating
E3	4, 11, 15, 26, 41, 67, 108, 175, ...	$= f(n-1) + f(n-2),\ f(1) = 4,\ f(2) = 11$	Fibonacci
E4	5, 6, 12, 19, 32, 52, 85, 138, ...	$= f(n-1) + (f(n-2) + 1),\ f(1) = 5,\ f(2) = 6$	nested
E5	8, 10, 14, 18, 26, [34, 50, 66,] ...	$= f(n-2) \times 2 - 2$ $= (f(n-2) - 1) \times 2$ $f(1) = 8, f(2) = 10$	not unique
	8, 10, 14, 18, 26, [34, 42, 58,] ...	$= f(n-1) + 2^n,\ f(1) = 8$	
E6	1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, ...	*write each number n n-times*	intuitive

underlying algorithmic principles. Based on these considerations, Number series may be characterized according to the following features:

Necessary background knowledge: To solve series, only knowledge of basic arithmetic operators (or even only of the successor function) is necessary. But series can become more efficiently solvable with mathematical knowledge such as knowing the factorial or checksum-functions.

Numerical values: Numbers are typically small in the context of psychometric tests. We can assume that humans have no problems with large values if they can be represented in a simple form, such as decimal multiples and we can assume that numerical values have less impact on performance of computer systems than of humans.

Structural complexity: Series can be solvable by application of one basic operation to the predecessor or might depend on complex relations between several predecessors.

Existence of a closed formula: Most number series of interest can be characterized by a closed formula as given in Table 1. However, some series, such as E6 in Table 1 can be easily described verbally while a closed form is highly sophisticated or even not known. Other problems even need a switch of perspective, such as *3, 3, 5, 4, 4, 3* which gives the number of letters of the verbal representation of the index.

We assume that these features influence performance of humans as well as machines, however not necessarily in the same way. In the context of psychometrics, difficulty of a problem is assessed by the percentage of subjects that solve the problem at hand on a given time. This measure does not capture characteristics of the number series and its impact on the cognitive or computational processes involved. An empirical investigation of the cognitive determinants of number series performance was presented by Holzman *et al.* [12]. It was shown that mathematical skill has an impact on performance for more complex series. A proposal to capture difficulty of number series problems based on a resource-bounded Kolmogorov complexity was made by Strannegård *et al.* [23] with a focus on structural complexity.

Table 2. Series solved by the anti-unification approach of Burghardt [4].

0,1,4,9	$f(n) = n * n$	0,1,2,1,4,1	$f(n) = if(even, n, 1)$
0,2,4,6	$f(n) = f(n-1) + 2$	0,0,1,1,0,0,1,1	$f(n) = even(n-2)$
1,1,2,3,5	$f(n) = f(n-1) + f(n-2)$	0,1,3,7	$f(n) = 2 \times f(n-1) + 1$

3 Systems Solving Number Series

Approaches for solving number series problems can be distinguished in systems which are specifically designed to solve this type of problems and in the application of general purpose algorithms or algorithms developed for a different problem domain. For both kinds of approaches, there are computer models which aim at performance criteria such as scope and efficiency and computer models which aim at simulation of cognitive systems.

Early Systems. The earliest computational approach for a cognitively-inspired AI systems solving number series is SeekWhence [10, 14]. Hofstadter aimed on an expert system which depends on a set of specific rules characterizing mathematical relations. Instead, his aim was to solve sequences using general principles such as pattern recognition and analogy. The system was able to identify well known sequences appearing interleaved. For example, given *1, 1, 3, 4, 6, 9* it recognizes the square numbers *1, 4, 9* and the triangle numbers *1, 3, 6*. Hofstadter was especially interested in sequences which do not require typical mathematical operations. One example is the index number problem (see E6 in Table 1). Another example is *1, 1, 1, 2, 1, 1, 2, 1, 2, 3*. To identify the inherent pattern of this sequence, chunking is necessary: *((1)) ((1)(12)) ((1)(12)(123))*. To solve such problems Mahabal [13] developed SeqSee influenced by the CopyCat system [10].

Sanghi and Dowe [19] presented a very simple program which was able to solve a variety of number series problems. This program was not intended as an AI or cognitive system as a demonstration that rather trivial programs can be able to pass an intelligence test. An approach developed in the context of automated theorem proving was applied to solve number series problems [4]: An algorithm for anti-unification of mathematical expressions was successfully applied to several number series, among them alternating series and Fibonacci (see Table 2).

Rule-Based Systems. In the last four years, two rule-based systems for solving number series were proposed. Siebers and Schmid [21] presented a semi-analytical approach where the term structure defining a given number series is guessed based on heuristic enumeration of term structure. To evaluate the approach, a generator of number series was realized (see also [5]) and the system was evaluated with 25,000 randomly created number series resulting in an accuracy of 93%.

A system based on similar principles is ASolver. However, this system takes into account plausible restrictions of working memory [24, 23]. Systems perfor-

```
nat   = 0  | s(nat)          eq Plustwo((s 0) nil) = s^3 0
[nat] = [] | nat:[nat]       eq Plustwo((s^3 0) (s 0) nil) = s^5 0
                             eq Plustwo((s^5 0) (s^3 0) (s 0) nil) = s^7 0
```

Fig. 1. Representation of a simple number series for IGOR2, `s` is the successor function.

mance was evaluated with 11 (non published) problems from the IQ test PJP and shown to outperform mathematical tools such as MAPLE and WOLFRAMALPHA.

An Inductive Programming Approach. Rule-based systems are specifically designed for solving problems from the number series domain. However, when being interested in systems which are able to general intelligent behavior, the challenge is to identify approaches which can be applied to different domains without specific adaptation and without a meta-algorithm which selects a suitable special purpose algorithm. The anti-unification approach of Burghardt [4] is a first example of a successful application of a system designed for a different domain to number series.

Another example is the inductive program system IGOR2 [9, 20] which learns functional (MAUDE or HASKELL) programs from small sets of input/output examples. For instance, given examples for reversing a list with up to three elements, IGOR2 generalizes the recursive *reverse* function together with helper functions *last* and *init*. IGOR2 is based on constructor-term rewriting and therefore, besides the examples for the target function, the data types have to be declared. For lists, the usual algebraic data type `[a] = [] | a:[a]` is used. To apply IGOR2 for induction of a constructor-function which correctly describes and continues a given number series, as a crucial first step, we have to decide how to represent the needed data types, equations, and background knowledge.

In a first investigation the effect of different representations on IGOR2's performance was investigated with 100 number series varied with respect to size of numerical values and structural complexity [8]. It turned out that the system performed comparably with all representation formats tested, in the following we only consider the format given in Figure 1: The system needs the data types for list and natural number as input. A number series problem is represented as a set of example equations. For instance, the sequence *1, 3, 5* is represented as three examples giving the sequence up to a given length as input and the next element as output.

IGOR2 can induce functions characterizing the infinite sequence without background knowledge, for series which can be characterized by incrementing or decrementing values of predecessors. For more complex series, more specialized mathematical operations can be pre-defined and the system can use them for rule construction. However, while the availability of mathematical knowledge typically will improve human performance [12], IGOR2' performance time and memory requirements increase when background knowledge is given.

A Neural Network Approach. All approaches introduced so far are based on symbolic computation. Such systems generate the solution of a number series problem by identifying the underlying regularity. The generalized rule to characterize the infinite sequence is explicitly constructed. That is, the symbolic systems are not only able to produce the next number, but to "explain" by the given function why this number was given. A limitation is that the set of functions that can be computed is rather restricted. In contrast, artificial neural networks (ANNs) are in principle able to approximate arbitrary functions. Ragni and Klein [16] investigated number series prediction with three-layered networks with error back-propagation and the hyperbolic tangent as activation function. The approach uses a dynamic learning approach: An ANN was trained on the given numbers and the missing number was the target value to be predicted. The number of training values of a pattern is equivalent to the number of input nodes i of the network used. Starting with the first number, a sub-sequence of training values was shifted through the number series. As corresponding target value, the next number of the sub-sequence was used. Since the last given value of the number series with length n remains as target value and at least one training and one test pattern is needed, the maximum length of a subsequence of training values is $n - 2$. Hence, for a network configuration with m input nodes $n - m$ patterns were generated. The first $(n - m) - 1$ patterns were used for training, while the last one remained for testing and thus predicting the last given number of the sequence.

4 Performance Comparison: Igor2, ANNs, and Humans

IGOR2's performance was tested with the series presented in [4] (see Table 2) and could generate correct solutions for all of them. We did not systematically evaluate performance time, since performance was very fast (some milliseconds) for all problems. Most series could be solved without background knowledge with the following exceptions: For Fibonacci, addition had to be pre-defined, for the square function, the square function had to be pre-defined, and for the series $f(n) = 2 \times f(n-1) + 1$, multiplication had to be pre-defined. IGOR2 could not solve problem E6 from Table 1 since it depends on identifying a pattern in form of a recursive function which is μ-recursive for this problem.

Furthermore, IGOR2's performance was tested against human performance on 20 series systematically varied with respect to structural complexity and numerical values [15]. Based on results of 46 subjects who participated in an online experiment, it showed that indeed, a lesser number of humans (34 out of 46) succeeded for number series with high numbers, such as *237, 311, 386, 462, 539*, characterizable as $f(n) = f(n-1) + n + 73$. Furthermore, due to the constructive representation of numbers, IGOR2 failed to solve this series. The largest constant for which IGOR2 could produce a result was for with problem was 36. Details of the study are given in [15],

The ANN approach was applied to the 73 SEQSEE number series problems described in section 3 and presented in [13]. As in a previous investigation [17]

settings were systematically varied from $500, 1000, 5000, 10000, 15000$, each with a learning rate ranging from .125 up to .875 with a step width of .125. The number of input nodes from 1 to 3 are varied, but the number of nodes within the hidden layer from 1 to 20. On average, over all number series, an increasing number of training iterations was counter-effective, that is, the number of solvable series was reduced. For 500 iterations 19 number series could not be solved by any configuration. For 15 000 iterations this number rose to 22. Over all types of configurations there remain 13 number series unsolved. Furthermore, the ANN approach was applied to number series given in intelligence tests: the 20 problems of IST, also investigated by Strannegård *et al.* [23] and the 14 problems of the MIT. Again number of input nodes, hidden layers, and learning rate were varied as above. Over all configurations 19 out of the 20 IST number series could be solved, one remains unsolved. For the MIT over all configurations 12 out of the 14 number series could be solved, two remain unsolved. Analyzing the networks show again, that 3 input nodes and about 5-6 hidden nodes with a low learning rate are the most successful ones. This pattern appears in all our benchmarks. Ragni and Klein [16] developed 20 number series as a benchmark for the ANN approach given in Table 3. The problems differed in the underlying construction principle and varied from simple additions and multiplications to combinations of these operations. One series (S12) is of the type studied by Hofstadter [10]: it is composed of the numbers which are not squares.

An empirical study with 17 human subjects was conducted[4]. Subjects received the series in randomized order on paper and had to fill in the last number of the series. With the exception of the low performance for the simple series S05, the empirical results support our assumptions: While humans deal easily with series based on a simple operation on the immediate predecessor, they have problems with series depending on more than one predecessor number (as the Fibonacci variant S11). Although humans can deal with alternating series for simple operations, they have problems if these series involve multiplications (S15) or a nested series depending on the index (S17). IGOR2 and the ANN approach were tested with the same problems. However, IGOR2 did only receive the first 5 elements of a series as input, the ANN was trained with 7 inputs and had to predict the 8th value. For some of the series, solution success of IGOR2 did depend on the chosen representation for the series. For some of the series, mathematical background knowledge was given to IGOR2 as described in section 3. Details of the empirical results for IGOR2 are given in [7]. Overall, there are six number series which could not be solved by IGOR2 and three number series which could not be solved by the ANNs. Among them is only one series (S06) which could be solved by neither approach.

5 Conclusions and Further Work

Number series form an excellent testbed for AGI-systems. An overview of systems solving number series problems show that some systems are designed to

[4] For more information please refer to [16]

Table 3. Empirical comparison of human performance ($n = 17$) with ANNs and IGOR2 (ID based on the order of the series as reported in [16]; results for humans are correct/incorrect/no answer).

			Responses		
ID	Number Series	Rule $f(n) =$	Human	IGOR2	ANN
05	2,5,8,11,14,17,20,23	$f(n-1)+3$	9/3/5	+	+
07	25,22,19,16,13,10,7,4	$f(n-1)-3$	16/0/1	+	+
19	8,12,16,20,24,28,32,36	$f(n-1)+4$	15/0/2	+	+
13	54,48,42,36,30,24,18	$f(n-1)-6$	16/1/0	+	+
08	28,33,31,36,34,39,37	$f(n-2)+3$	17/0/0	+	+
14	6,8,5,7,4,6,3,5	$f(n-2)-1$	16/0/1	+	+
20	9,20,6,17,3,14,0,11	$f(n-2)-3$	16/0/1	-	+
01	12,15,8,11,4,7,0,3	$f(n-2)-4$	15/0/2	+	+
11	4,11,15,26,41,67,108	$f(n-1)+f(n-2)$	8/1/8	+	+
09	3,6,12,24,48,96,192	$f(n-1)\times 2$	13/1/3	+	-
16	7,10,9,12,11,14,13,16	$if(even, f(n-1)+3, f(n-1)-1)$	14/0/3	+	+
18	8,12,10,16,12,20,14,24	$if(even, f(n-2)+4, f(n-2)+2)$	17/0/0	+	+
15	6,9,18,21,42,45,90,93	$if(even, f(n-1)+3, f(n-1)\times 2)$	14/1/2	-	+
17	8,10,14,18,26,34,50,66	$if(even, f(n-2)+6\times 2^i, f(n-2)+8)$	13/1/3	-	+
10	3,7,15,31,63,127,255	$f(n)=2\times f(n-1)+1$	12/3/2	+	-
04	2,3,5,9,17,33,65,129	$f(n-1)+f(n-1)-1$	13/1/3	+	+
03	2,12,21,29,36,42,47,51	$f(n-1)+12-n$	14/1/2	-	+
02	148,84,52,36,28,24,22	$(f(n-1)/2)+10$	12/2/3	+	+
06	2,5,9,19,37,75,149,299	$f(n-1)\times 2+(-1)^n$	6/4/7	-	-
12	5,6,7,8,10,11,14,15	*no squares*	10/1/6	-	+

model human cognitive processes while others aim at high performance. We introduced two approaches: The inductive programming system IGOR2 is a symbolic approach to learning declarative rules from examples and a sub-symbolic approach using ANNs to function estimation. We compared both approaches with human performance. It showed that the ANN approach could solve more problems than IGOR2. However, each ANNs must be trained for each series taking several thousand training iterations while IGOR2 could be applied to all series without adaptation. Furthermore, IGOR2 not only returns the next number but also the function which explains how the solution was generated. This is more similar to humans that can justify a given solution. This approach shows that there are many interesting questions left. To compare systems systematically, a benchmark set – a repository of problems with a difficulty measure independent of a specific systems might be necessary. A first step into this direction was made by Strannegård *et al.* [23] who characterized problem difficulty by bounded Kolmogorov complexity, but depending on a specific algorithm. Alternatively, human performance could be used as a guideline. Given the 20 series investigated, IGOR2 as well as the ANN could not solve all problems and they differed from human performance. However, the 20 series do not represent a systematic variation over the features characterizing problems as described in section 2. As a next step, we plan to compose a more systematic repository of

problems and to invite researchers to discuss and propose other number series problems – towards a systematic competition in this domain.

Acknowledgements

This work has been supported by a Heisenberg-fellowship to the second author and a grant within the SPP 1516 "New Frameworks of Rationality". The authors are grateful to Andreas Klein for helping in evaluating the ANNs.

References

1. Amthauer, R., Brocke, B., Liepmann, D., Beauducel, A.: Intelligenz-Struktur-Test 2000 (I-S-T 2000). Hogrefe, Goettingen (1999)
2. Besold, T., Hernández-Orallo, J., Schmid, U.: Can machine intelligence be measured in the same way as human intelligence? KI – Künstliche Intelligenz (2015)
3. Bringsjord, S.: Psychometric artificial intelligence. Journal of Experimental & Theoretical Artificial Intelligence 23(3), 271–277 (2011)
4. Burghardt, J.: E-generalization using grammars. Artificial Intelligence 165, 1–35 (2005)
5. Colton, S., Bundy, A., Walsh, T.: Automatic invention of integer sequences. In: AAAI/IAAI. pp. 558–563 (2000)
6. Hernández-Orallo, J., Dowe, D.L., Hernández-Lloreda, M.V.: Universal psychometrics: Measuring cognitive abilities in the machine kingdom. Cognitive Systems Research 27, 50–74 (2014)
7. Hofmann, J.: Automatische Induktion über Zahlenreihen – Eine Fallstudie zur Analyse des induktiven Programmiersystems IGOR2 (Automated induction of number series – A case study analysing the inductive programming system IGOR2). Master's thesis, University of Bamberg (December 2012)
8. Hofmann, J., Kitzelmann, E., Schmid, U.: Applying inductive program synthesis to induction of number series a case study with IGOR2. In: KI 2014: Advances in Artificial Intelligence. pp. 25–36. Springer (2014)
9. Hofmann, M., Kitzelmann, E., Schmid, U.: A unifying framework for analysis and evaluation of inductive programming systems. In: Goerzel, B., Hitzler, P., Hutter, M. (eds.) Proceedings of the Second Conference on Artificial General Intelligence (AGI-09, Arlington, Virginia, March 6-9 2009). pp. 55–60. Atlantis Press, Amsterdam (2009)
10. Hofstadter, D.: Fluid Concepts and Creative Analogies. Basic Books, New York (1995)
11. Holland, J., Holyoak, K., Nisbett, R., Thagard, P.: Induction – Processes of Inference, Learning, and Discovery. MIT Press, Cambridge, MA (1986)
12. Holzman, T.G., Pellegrino, J.W., Glaser, R.: Cognitive variables in series completion. Journal of Educational Psychology 75(4), 603–618 (1983)
13. Mahabal, A.A.: Seqsee: A concept-centred architecture for sequence perception. Ph.D. thesis, Indiana University Bloomington (2009)
14. Meredith, M.J.E.: Seek-whence: a model of pattern perception. Tech. rep., Indiana Univ., Bloomington (USA) (1986)
15. Milovec, M.: Applying Inductive Programming to Solving Number Series Problems – Comparing Performance of IGOR with Humans. Master's thesis, University of Bamberg (September 2014)

16. Ragni, M., Klein, A.: Predicting numbers: An AI approach to solving number series. In: Edelkamp, S., Bach, J. (eds.) Proceedings of the 34th German Conference on Artificial Intelligence (KI-2011). LNCS, Springer (2011)
17. Ragni, M., Klein, A.: Solving Number Series - Architectural Properties of Successful Artificial Neural Networks. In: Madani, K., Kacprzyk, J., Filipe, J. (eds.) NCTA 2011 - Proceedings of the International Conference on Neural Computation Theory and Applications. pp. 224–229. SciTePress (2011)
18. Raven, J., et al.: Raven progressive matrices. In: Handbook of nonverbal assessment, pp. 223–237. Springer (2003)
19. Sanghi, P., Dowe, D.L.: A computer program capable of passing I.Q. tests. In: Slezak, P.P. (ed.) Proc. Joint 4th Int. Conf. on Cognitive Science, and & 7th Conf. of the Australasian Society for Cognitive Science (ICCS/ASCS-2003). pp. 570–575. Sydney, NSW, Australia (2003)
20. Schmid, U., Kitzelmann, E.: Inductive rule learning on the knowledge level. Cognitive Systems Research 12(3), 237–248 (2011)
21. Siebers, M., Schmid, U.: Semi-analytic natural number series induction. In: KI 2012: Advances in Artificial Intelligence, pp. 249–252. Springer (2012)
22. Sternberg, R.J. (ed.): Handbook of Intelligence. Cambridge University Press (2000)
23. Strannegård, C., Nizamani, A., Sjöberg, A., Engström, F.: Bounded Kolmogorov complexity based on cognitive models. In: Kühnberger, K.U., Rudolph, S., Wang, P. (eds.) Artificial General Intelligence (AGI'13). pp. 130–139. Springer (2013)
24. Strannegård, C., Amirghasemi, M., Ulfsbäcker, S.: An anthropomorphic method for number sequence problems. Cognitive Systems Research 22-23, 27–34 (2013)
25. Wilhelm, O., Conrad, W.: Entwicklung und Erprobung von Tests zur Erfassung des logischen Denkens. Diagnostica 44, 71–83 (1998)