Solving Number Series with the MagicHaskeller

by

Matthias Düsel, Alexander Werner, Theresa Zeißner

REPORT

Submitted to the Cognitive Systems Group
Faculty of Information Systems and Applied Computer Sciences
University of Bamberg

KOGSYS-KOGMOD-M: KOGNITIVE MODELLIERUNG

Submission Date: 20.02.2012

Assessor: Prof. Dr. Ute Schmid

Table of Contents

Chapter 1
Introduction

As a part of many classical intelligence tests in psychology, solving number series pose an interesting and also challenging task for both humans and machines. Consequently, in this year's project of cognitive modelling solving number series took center stage. After being acquainted with analyses concerning the human way of solving number series, the idea arose how the MagicHaskeller as an inductive functional programming system is solving such series. Additionally and even more interesting the project aimed to examine how to put the expected human constraints into the MagicHaskeller and analyse its performance.
In chapter 2 of theoretical background the report first gives an insight into the idea of solving number series and its relevance, for example in intelligence tests. After that the functional programming language Haskell will be introduced since it sets the base for the MagicHaskeller, belonging to the group of programming automation. This inductive functional programming system is based on systematic exhaustive search and combines two approaches in its current release (version 0.8.6). The two approaches will be presented respectively and in combination.
Chapter 3 is examining the human way of solving number series in an empirical study. To survey evaluable data people were asked to solve number series both in a time-constraint test gaining quantitative data and in a think-aloud test gaining qualititative data.
The empirical study is finally followed by the implementation of number series in the MagicHaskeller in chapter 4, consolidating the knowledge of the previous two chapters. In other words, how to make the MagicHaskeller solve number series using human strategies observed in the empirical study.
The following report aimes both to demonstrate the work of our group during the project of cognitive modelling and give ideas and impulses for potential further work on this topic.

Chapter 2
Theoretical Background

The following section describes the diverse theoretical backgrounds which had an influence on our task. As cognitive modelling is a field of research in cognitive sciences, the adjacent disciplines are spread widely. We constricted our work to the fields of psychology and computer sciences.

2.1 Solving number series

2.1.1 Psychology

Solving number series has a long tradition in psychometry as a test of inductive reasoning. The number series completion task is one subitem of a intelligence test since the 60ths of last century up to now and used as a predictor for numerical abillities as a subfactor of intelligence. One big advantage of items from an intelligence test are their big norm samples.
Which steps are made by humans while solving number series? Holzmann described 4 stages of solving sequence problems[1](also see 2.1).

1. Relation detection (scanning for relations, first hypothesis)
2. Discovery of periodity (looking for repeating relations)
3. Completion of pattern description (relate remaining elements within a period)
4. Extrapolation (apply found rule to the sequence)

Another interesting aspect of solving number series is their difficulty. Therefore, it is very helpful to decompose a series to its critical parts. Holzmann proposed four characteristica of difficulty.

[1] taken from [Wes03, p.20]

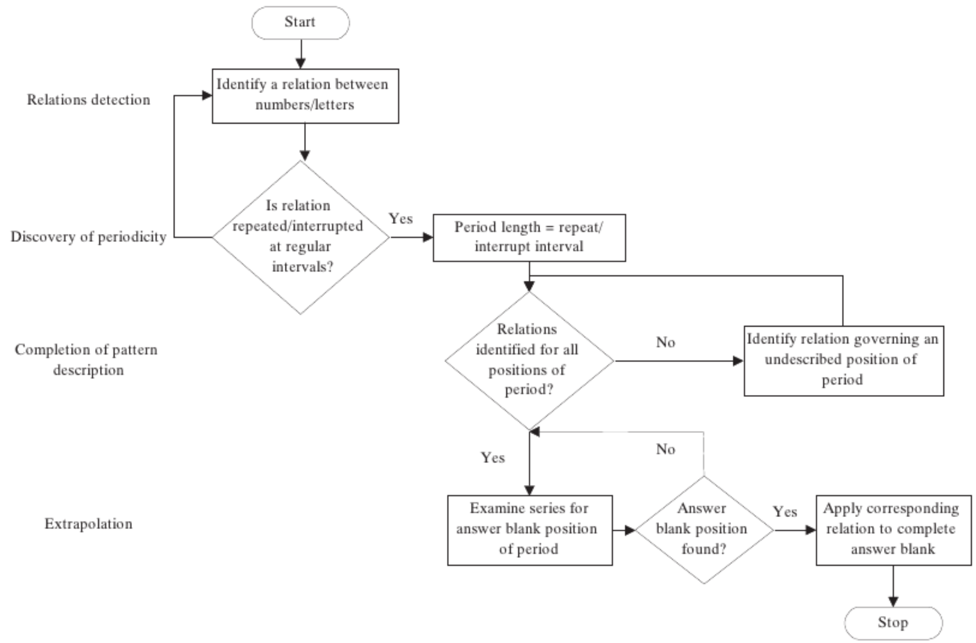

Figure 2.1: Stages of solving sequence problems (Holzmann). From [Wes03, p.20].

1. types of relations
2. magnitude of arithmetic operator
3. complexity of operation
4. length of the period

He was able to show, that number series from type (+) and (-) are solved more easily than (*) and (div). He got the same results for the magnitude while a lower magnitude of the operator like +4 was solved more easily than higher ones (e.g +14). Complexity of an operation is determined by the number of different arguments and is, together with the length of the period, one of the stressing components for the working memory.

2.1.2 Computer Sciences

In Computer Sciences we found some interesting projects, too. The most important one to us is seqsee[Mah09]. Mahabal describes his approach as concept centered and distinguishes it from pure brute-force programs like Superseeker which is used by the encyclopedia of integer series. Our approach will also show how an exhaustive program can be modelled to be a bit more cognitive.

2.2 Haskell and Cognitive Modelling

The MagicHaskeller is written in the functional programming language Haskell where the knowledge representation scheme is based on functions and recursion patterns. This leads to a high programming efficiency as it allows the programmer to work at a higher level of abstraction. Due to static type checking of Haskell, there is more chance to spot errors in the code before it runs. Another advantage of functional programs are code efficiency and a more robust character than imperative programs offer.

2.3 Inductive Functional Programming (IFP)

As an inductive functional programming system based on systematic exhaustive search, the MagicHaskeller belongs to the group of programming automation. According to Katayama [Kat11] programming automation can be defined as programming paradigm, "where recursive functional programs are synthesized through generalization from the ambiguous specification usually given as a set of input-output pairs" [Kat11].
At the moment, there are two approaches to Inductive Functional Programming. The MagicHaskeller was first a generate-and-test method based on systematic exhaustive search, but was recently extended by an analytical synthesis engine. The two approaches are to be introduced in the following.

2.3.1 Generate-and-test Approach

The generate-and-test approach "generates many programs(all the programs which can be constructed using the given primitive set) and picks up those that satisfy the [given] specification" [Kat11].

2.3.2 Analytical Approach

As an supplementary approach, the analytical approach "synthesizes programs by looking into the input-output pairs and conducting inductive inference" [Kat11].

2.3.3 Analytically-generate-and-test Approach

The MagicHaskeller combines both approaches. Since the year 2005, it was a "mere" generate-and-test program based in systemtic exhaustive search, but since its version 0.8.6 in 2011 it also supports the analytical search algorithm and therefore can be called "analytically-generate-and-test approach".
This combination of the two approaches leads to a analytical synthesizer that "generates many programs from the given insufficient set of input-output examples and picks up those that satisfy the predicate seperately given as a part of the specification" [Kat11]. The search is conducted in the "breath-first manner".

2.4 Application of MagicHaskeller

Invoke MagicHaskeller:

```
> ghci -package MagicHaskeller -XTemplateHaskell
```

Bringing module MagicHaskeller.LibTH (for exhaustive search) or alternatively MagicHaskeller.RunAnalytical (for analytical synthesis) into scope:

```
> :m +MagicHaskeller.LibTH   or
> :m +MagicHaskeller.RunAnalytical
```

For exhaustive search:
Initialize the environment and set the component library:

```
> init075
```

Function printAll "requests to print all the expressions which can be synthesized using the combinators in the component library that satisfy a given predicate"[Kat11]:

```
> printAll -> f "abc" == "aabbcc"
```

For analytical synthesis:
Function quickStart takes several input-output pairs and a predicate to filter the gen-

erated programs as STinput. Optionally, there can be defined background knowledge:

```
> quickStart [ d | f [ ] = 0 ; f [ a ] = 1 | ]
             noBKQ
             (f -> f " 12345 " == 5)
```

The main interest of our project, solving number series, will be the usage of the background knowledge function supported by the modules for analytical synthesis such as "RunAnalytical". This background knowledge was also given to our test subjects of the following empirical study.

Chapter 3
Empirical Study

Before we were able to model a human way of solving number series, we had to survey some data. Some quantitative data in order to check a little hypothesis of ours and qualitative data. Qualitative data was collected to analyse human approaches in an exploratory way. We tested six participants, which is way to less for the quantitative approach, but sufficient for the qualitative one. The materials we used can be found in Appendix A.

3.1 Quantitative approach

3.1.1 Intention and hypothesis

In the quantitative approach we tried to test a hypothesis of ours. Does it make a difference whether a person is focused on the main operators ((+),(-),(*) and (div)) while solving number series. Our alternative Hypothesis (h1):

People with certain background knowledge are able to solve more number series than those without.

3.1.2 Experimental setup

We used twenty number series from a standardized intelligence test (see also 3.2[1]). Normally, these number series are just one sub test out of many, but we used it exclusively. Nevertheless, we executed the test under the same frame conditions, such as the time limit (10 min.), and the same instructions. To check our hypothesis, we added the information ”you just have to use the operators ((+),(-),(*) and (div))” to the instruction page. We also controlled the variables ”age” (M = 25, SD = 3,03) and ”graduation” (at least university-entrance diploma) to make our results comparable to the norm tables which are included in the intelligence test. These additional data allowed us to use a test design without a control group.

[1] #op = number of different operators, #arg = number of different arguments, #ch.op = number operator changes, #ch.op = number argument changes

	N	Mean	SD
without operator knowledge	478	14.05	4.89
with operator knowledge	6	14.83	4.88

Table 3.1: Descriptive statistics of the two conditions.

3.1.3 Data analysis and results

Although our sample size was very small (N = 6) and uneven compared to the norm tables, we might have found some interesting results. In table 3.1.3 our results are compared to those in the norm tables.
You may see that there are almost no differences in mean and standard derivation between those two conditions. The result of our own sample t-test emphasizes our assumption ($t = 0.3936$, $df = 5$, p-value $= 0.7101$). So what are the consequences for our project? If we want to interpret this result, we will have to discard our alternative hypothesis that the additional knowledge about the search space boosts the number of correct solved number series. Furthermore, we may come to the conclusion that the search space is already restricted or focused to the basic arithmetic operators in the first place. Unfortunately, we can not confirm this conclusion by looking at our qualitative data, because all participants had to do the quantitative part first and might be focused on those operators, although we explicitly removed this restriction before the qualitative part of our survey. The detailed calculations can be found in the R-Script (comments are in German) in Appendix A.

Table 3.2: number series for the quantitative approach

#	number series	next	op.	#op.	#arg.	#ch. op	#ch. arg	solution
01)	2,5,8,11,14,17,20	23	+	1	1	0	0	+3
02)	1,3,6,8,16,18,36	38	+,*	2	1	n		+2, *2, alternating
03)	9,12,16,20,25,30,36	42	+	1	n	0	n	+3+4+4+5+5...
04)	18,16,19,15,20,14,21	13	-,+	2	n	1	n	-2+3-4+5-6+7...
05)	33,30,15,45,42,21,63	60	-,*, /	3	2	3	1	-3,/2,*3, alternating
06)	25,27,30,15,5,7,10	5	+,/	2	2	1	3	+2,+3,/2,/3, alternating
07)	11,15,18,9,13,16,8	12	+,/	2	3	1	2	+4+3/2, alternating
08)	5,6,4,6,7,5,7	8	+,-	2	2	2	1	+1-2+2, alternating
09)	8,11,7,14,17,13,26	29	+,-, *	3	3	2	2	+3-4*2, alternating
10)	35,39,42,21,25,28,14	18	+,/	2	3	1	2	+4+3/2, alternating
11)	55,57,60,20,10,12,15	5	+,/	2	2	1	2	+2+3/3/2, alternating
12)	57,60,30,34,17,22,11	17	+,/	2	n	1	n	+3/2+4/2+5/2...
13)	2,3,6,11,18,27,38	51	+	1	n	0	n	+1+3+5+7+9...
14)	7,5,10,7,21,17,68	63	-,*	2	n	1	n	-2*2-3*3-4*4...
15)	11,8,24,27,9,6,18	21	-,*, +,/	4	1	3	0	-3*3+3/3, alternating
16)	15,19,22,11,15,18,9	13	+,/	2	3	1	2	+4+3/2, alternating
17)	13,15,18,14,19,25,18	26	+,-	2	n	1	n	+2+3-4+5+6 -7+8...
18)	15,6,18,10,30,23,69	63	-,*	2	n	1	n	-9*3-8*3-7*3-6...
19)	8,11,16,23,32,43,56	71	+	1	n	0	n	+3+5+7+9...
20)	9,6,18,21,7,4,12	15	-,*, +,/	4	1	3	0	-3*3+3/3, alternating

3.2 Qualitative approach

In our qualitative approach the participants had to solve 15 number series, ten additional ones and five from the quantitative approach (see also 3.3). In order to document the results, we recorded the think aloud from the participants and transcribed them afterwards. The complete protocols can be found in Appendix A.
The analysis of the think aloud protocols unveiled some interesting aspects of human approaches to solving number series. In sum we could detect three properties of number series which provide information for the solving process.

- direction of slope
- size of the gaps (amount of the slope)
- knowledge about associations between certain numbers

We suppose that a first step while solving number series is a quick pattern match between the seen numbers and known relations between them. This step might already be a routine and therefore more implicit. As a result of this step, the direction of the slope determines the direction of the operator (rising or falling, monotony). After that we saw three different strategies.

- step by step
  - single operator
  - multiple operators
- excessive pattern matching

One strategy was to fill the gap between two numbers with one operation with the disadvantage to get big arguments. Therefore (+) or (-) was selected, because it is sufficiently fine grained to fill all gaps. The corresponding result was another number series (the series of arguments, e.g +1,+2,+4,+8). Then the pattern match method mentioned above is used again to detect a regularity. The interesting result beside the efficiency of this strategy is its recursive characteristics.
The participants, who used the second strategy, tried to fill the gaps with more operators, combining the advantage of low arguments and the costs of multiple operators. Big gaps had been filled in two steps. The first step was the "greedy step", consisting of a multiplication/division and the second step was the "precisely tuning step",

Table 3.3: number series for the qualitative approach (1-15) and for the implementation (1-20)

#	number series	next	op.	#op.	#arg.	#ch. op	#ch. arg	solution
01)	148,84,52,36,28,?	24	-,/	2	n			-(n/2)
02)	2,3,5,6,7,8,10,?	11	+	1				drop squares
03)	2,3,5,9,17,?	33	+	1	n	0		+predeccesor
04)	3,7,15,31,63,?	127	+	1	n	0		+successor
05)	9,20,6,17,3,?	14	-	1	1	0		-3, alternating
06)	0,1,4,13,40,?	121	*,+	2	2	1		*3+1
07)	99,18,36,9,18,?	9	+,*	2	n	n		checksum, *2, alternating
08)	10,45,15,38,20,?	31	+,-	2	2	1		+5,-7, alternating
09)	4,5,7,11,19,?	35	+,*	2	n			+(n*2)
10)	2,5,9,19,37,?	75	*,+,-	3	n	0		*2+1, *2-1, alternating
11)	2,5,8,11,14,17,20,?	23	+	1	1	0		+3
12)	1,3,6,8,16,18,36,?	38	+,*	2	1	n		+2, *2, alterniating
13)	33,30,15,45, 42,21,63,?	60	-,/, *	3	2	n		-3,/2,*3, alternating
14)	25,27,30,15,5,7,10,?	5	+,/	2	2	n/2	n	+2,+3,/2,/3, alternating
15)	11,8,24,27,9,6,18,?	21	-,*, +,/	4	1	n	0	-3,*3,+3,/3, alternating
16)	-2,5,-4,3,-6,?	1	-	1	1	0	0	-2, alternating
17)	16,22,34,58,106,?	202	+,*	2	n			+(n*2)
18)	1,1,2,3,5,?	8	+	1	n	0	n	+ previous item
19)	2,5,10,17,26,?	37	ˆ2, +	2	n	0	n	$(n^2) + 1$
20)	2,3,10,15,26,?	35	ˆ2, +	2	n	n	n	odd squares +1, even squares -1

which means the filling of the leftover to the target by using addition/subtraction (e.g. 15, 31 → *2 , +1).
A third strategy, which was observed, was an excessive use of pattern matching and associations. Corresponding numbers and their relations had been visualized by symbols to support this process. This strategy was able to solve number series with different alternating operations easily.

Chapter 4
Implementation

4.1 Transfer from empirical data

In the following part we implemented some of the empirical results. As a consequence of the quantitative and qualitative results, we focused on the following tasks:

- use of 4 basic arithmetic operators
- direction of slope determines operators
- use of single operators
- use of multiple operators
- knowledge about associations between certain numbers

4.2 First approaches

As mentioned above the module Run.Analytical offers the function quickStart which takes 3 arguments, a list of input/output pairs, a list of input/output pairs for the background knowledge and a function to test against.

```
quickStart [d| f 1 = 2;  f 2 = 3;  f 3 = 4;
               f 4 = 5; |] noBKQ (\f -> f 5 == 6)
```

Our first approaches were focused on the input/output pairs. We translated the number series into input/output pairs in the following manner:

```
1, 2, 3, 4, 5, 6, ? --> [d| f 1 = 2;  f 2 = 3;  f 3 = 4; f 4 = 5; |]
```

To reduce complexity, we resigned to make use of the background knowledge (noBKQ). As a test function we used the last pair of the number series.

```
(\f -> f 5 == 6)
```

Unfortunatelly, this implementation could only solve one of the 20 number series, which was just a successor function. We were not able to get a run with defined background knowledge, excepting the length function presented in demonstration paper[Kat11].

4.3 Second approach

After correspondence with the author of the MagicHaskeller, we were given a complete new approach of implementing our problem (solving number series) to the MagicHaskeller using the generate-and-test approach[1] He sent us a quick hack which could solve 7 of the 20 series. The approach uses the ProgrammGenerator of MagicHaskeller and uses a component library, containing the operators and the arguments for our series. If arguments between 1 and 3 are needed, "smallNats" can be chosen. Otherwise, "nats" must be used with performance costs.

```
seriesLib, smallNats, nats :: [Primitive]
seriesLib= $(p [|(series :: (->) Int ([Int->Int] -> [Int]),
[]::[Int->Int], (:)::(Int->Int)->[Int->Int]->[Int->Int],
(+)::Int->Int->Int, (-)::Int->Int->Int, (*)::Int->Int->Int,
div::Int->Int->Int)|])
smallNats = $(p [|(1::Int, 2::Int, 3::Int)|])
nats      = $(p [|(1::Int, succ::Int->Int)|])
```

The basic function is "series". It takes a start number and a set of mathematical functions and produces a continuous stream of a number series.

```
series :: Int -> [Int->Int] -> [Int]
series i = scanl (flip ($)) i . cycle
```

A typical call is shown in the following example. We build a number series, which starts at 1, alternately adds two, and doubles the value.

```
Prelude> take 7 $ series 1 [(+2),(*2)]
 [1,3,6,8,16,18,36]
```

[1] At this point very special thanks to Mr. Katayama for his quick and very helpful support.

To get the mathematical function of a given number series, we call the MagicHaskeller for example as followed:

```
filterSeries (\f -> take 7 (f (2::Int)) == [2,5,8,11,14,17,20::
    Int]) >>= MagicHaskeller.pprs
```

Now, the MagicHaskeller tries to construct a number series, which starts on 2.The first seven elements are [2,5,8,11,14,17,20]. It uses the defined mathematical operators (seriesLib) and numbers (smallNats) and generates number series with the function "series". For each generated number series, the first seven elements are matched against the test function (here [2,5,8,11,14,17,20]). All found solutions are printed out as shown in the table 4.1.
With this module[2], we were able to start implementing some empirical results to the MagicHaskeller. The four arithmetic operators are already in the library. To solve series #7, we implemented the concept of checksums as number association knowledge. To do so, we defined a function for checksums and added it to the component library.

```
checksum :: Int -> Int
checksum i
  | i == 0     = 0
  | otherwise = i `mod` 10 + checksum (i `div` 10)
```

We tried the same with a concept for square numbers, because many number series contain squared arguments, but we didn't succeed. More concepts we considered to implement are "even", "odd" and "prime numbers", because they are often used besides the basic operators.
Our team tried to implement some further functions to solve as many number series as possible. Unfortunately, we were only able to get solutions for eight out of the twenty given series, which is shown in the next section. In addition to that we had some more ideas to get the MagicHaskeller work for the other series. Due to the fact that no alternating number series(e.g. #05) could be solved, it would be a good idea to implement some kind of split function. This function should split a number series

[2]Series.hs can be found in Appendix B

into two, in case there is no solution after a specific time. Therefore the result for the series 9,20,6,17,3 would be 9,6,3 and 20,17. With this input, the MagicHaskeller is able find the solution (-3).
An enhancement to increase the performance would be a limitation of the operators which should be scanned. Is there for example a number series with continuously increasing values, it would be a waste of time to try to find functions with subtraction or division. This was partially realized, which is described in detail in the next section. These two additions were not implemented because of the lack of time and Haskell skills.

4.4 Final results

As mentioned in the last section, a limitation of operators effects an improvement of performance. This was tested with the number series [1,11,21,31,41]. At first, we defined a second set of operators, which we called seriesLibPos. In this set, there are only addition and multiplication.

```
seriesLibPos :: [Primitive]
seriesLibPos = $(p [|(series :: (->) Int ([Int->Int] -> [Int]),
[]::[Int->Int], (:)::(Int->Int)->[Int->Int]->[Int->Int],
(+)::Int->Int->Int, (*)::Int->Int->Int)|])
```

On a virtual Ubuntu system (Intel Core i7, 2GB RAM) we tested the series with the original seriesLib with all operators and after that with our modified set. With the original set, MagicHaskeller needs 38 seconds to find the first solution. In the second run with only the positive operators, the first solution was provided after some milliseconds. This test of course is not very precise, but it gives a good tendency.
A function, which detects automatically if number series have continuously increasing or decreasing values and therefore use a minimized set of operators, would be a good way to raise the performance of MagicHaskeller for solving number series.

Table 4.1: Solved number series

#	number series	first 4 solutions of MagicHaskeller
3	2,3,5,9,17,?	\a → series a [\b → b + (b - 1)]
		\a → series a [\b → b - (1 - b)]
		\a → series a [\b → (b - 1) + b]
		\a → series a [\b → (b + b) - 1]
4	3,7,15,31,63,?	\a → series a [\b → b + (b + 1)]
		\a → series a [\b → b + (1 + b)]
		\a → series a [\b → 1 + (b + b)]
		\a → series a [\b → 1 + (b * 2)]
6	0,1,4,13,40,?	\a → series a [\b → 1 + (b * 3)]
		\a → series a [\b → 1 + (3 * b)]
		\a → series a [\b → (b * 3) + 1]
		\a → series a [\b → (3 * b) + 1]
7	99,18,36,9,18,?	\a → series a [\b → checksum b, \b → b + b]
		\a → series a [\b → checksum b, \b → b * 2]
		\a → series a [\b → checksum b, \b → 2 * b]
9	4,5,7,11,19,?	\a → series a [\b → b + (b - 3)]
		\a → series a [\b → b - (3 - b)]
		\a → series a [\b → (b - 3) + b]
		\a → series a [\b → (b + b) - 3]
11	2,5,8,11,14,17,20,?	\a → series a [\b → b + 3]
		\a → series a [\b → 3 + b]
		\a → series a [\b → b + (a + 1)]
		\a → series a [\b → a + (b + 1)]
12	1,3,6,8,16,18,36,?	\a → series a [\b → b + 2, \b → b + b]
		\a → series a [\b → b + 2, \b → b * 2]
		\a → series a [\b → b + 2, \b → 2 * b]
		\a → series a [\b → 2 + b, \b → b + b]
17	16,22,34,58,106,?	\a → series a [\b → 2 * (b - (2 + 3))]
		\a → series a [\b → 2 * (b - (3 + 2))]
		\a → series a [\b → 2 * (b - div a 3)]
		\a → series a [\b → 2 * ((b - 2) - 3)]

Chapter 5
Conclusion and Future Prospects

To sum up, solving number series with MagicHaskeller is possible. The program generation is powerful and can perform more efficently by focusing on human constraints. Moreover we could show that certain special number knowledge can be added to the MagicHaskeller to perform more effectively. The implementation of the automatic split function and the automated selection of operators are future prospects.

Bibliography

[Kat11] Susumu Katayama. *MagicHaskeller: System Demonstration*. University of Miyazaki, 2011.

[Mah09] A.A. Mahabal. *SEQSEE: AConcept-CENTERED ARCHITECTURE FOR SEQUENCE PERCEPTION*. PhD thesis, Indiana University Bloomington, 2009.

[Wes03] G. Wesiak. Ordering inductive reasoning tests for adaptive knowledge assesments. *An Application of Surmise Relations between Tests. Graz: University of Graz*, 2003.

Appendix A

Appendix A

Instruktionen

Vielen Dank, dass Sie sich die Zeit genommen haben an unserer Untersuchung zu Zahlenreihen teilzunehmen. Die Untersuchung besteht aus drei Teilen und dauert ca. 45 Minuten. Im ersten Teil sollen Sie innerhalb einer bestimmten Zeit soviele Zahlenreihen fortsetzen, wie es Ihnen möglich ist. Im zweiten Teil sollen Sie dann ohne Zeitdruck Zahlenreihen lösen und währenddessen die Methode des 'lauten Denkens' verwenden. Um uns die Auswertung zu ermöglichen möchten wir Ihre ausgesprochenen Gedanken aufzeichnen. Die erhobenen Daten werden selbstverständlich anonym behandelt und nur zum Zweck dieser Untersuchung genutzt. Abschließend möchten wir Sie bitten, einen kleinen Fragebogen zu beantworten.

Einverständniserklärung

Hiermit erkläre ich ____,

(Vorname, Nachname)

- dass ich die Information zur Studie erhalten habe.
- dass ich ausreichend schriftlich und mündlich über die Studie informiert wurde.
- dass ich damit einverstanden bin, dass während der Untersuchung meine ausgesprochenen Gedanken aufgezeichnet werden und dass diese Aufzeichnungen anonym gespeichert werden.

Meine persönlichen Daten unterliegen dem Datenschutz und werden ausschließlich anonymisiert gespeichert.
Bamberg, den ____

(Unterschrift)

Erster Teil (quantitativ)

Im folgenden Teil legen wir Ihnen 20 Zahlenreihen vor. Sie haben einige Minuten Zeit soviele Zahlenreihen fortzusetzen, wie es Ihnen möglich ist. Ausführliche Beispiele folgen auf der nächsten Seite. Blättern Sie die folgenden Seiten erst auf ein Zeichen des Versuchsleiters um.

Hinweis: Es müssen nur die Operatoren +, -, * und :, also Addition, Subtraktion, Multiplikation und Division verwendet werden.

Zweiter Teil (qualitativ)

Lautes Denken In der heutigen Untersuchung möchten wir außerdem herausfinden, was Ihnen beim Lösen verschiedener Zahlenreihen durch den Kopf geht. Um dies herauszufinden, möchten wir zusammen mit Ihnen die Methode des ‚Lauten Denkens' verwenden. Bei dieser Methode

bekommen Sie von uns eine Reihe von Zahlenreihen vorgelegt, welche Sie möglichst zügig lösen sollen. Während Sie die Aufgabe bearbeiten, sollen Sie parallel dazu alle Gedanken aussprechen, die Ihnen währenddessen durch den Kopf gehen. Bitte sprechen Sie dabei ganz frei all Ihre Gedanken aus, die Ihnen durch den Kopf gehen, ohne diese vorher zu bewerten. Bei dieser Aufgabe gibt es keine richtigen oder falschen Gedanken. Für diesen Teil der Untersuchung

haben Sie soviel Zeit wie Sie benötigen. Der Fokus liegt in diesem Teil auf dem Lösungsweg und nicht der Geschwindigkeit. Versuchen Sie dennoch die Aufgaben möglichst zügig zu lösen.

Course: Kognitive Modellierung WiSe 2011/12 Date: 13. Dezember 2011
Lecturer: Prof. Dr. Ute Schmid Contributors: Matthias Düsel, Alexander Werner, Theresa Zeißner

Fragebogen

Abschließend möchten wir Ihnen noch einige allgemeine Fragen und Fragen speziell zum Lösen von Zahlenreihen stellen.

Demographische Daten

1. Alter:
2. Geschlecht:
3. höchster Bildungsabschluss:

spezifische Fragen zu Zahlenreihen

1. Wie lang sollte eine Zahlenreihe, damit Sie ein Muster erkennen?
2. Welche Hintergrundwissen haben Sie einfließen lassen?

Vielen Dank für Ihre Teilnahme!

A.2 R-Script - Quantitative Analysis

```
# Zusatzpakete installieren.

#install.packages("psych")
#install.packages("psy")

# Pakete laden.
library(psych)
library(psy)

# Rohdaten einlesen aus csv-Datei
daten <- read.csv2(file = "~/Dropbox/KogMod/Untersuchung/Ergebnisse/AlleErgebnisse_rein.csv",
header = TRUE, sep = ";")

# Datensatz einbinden
attach(daten)

########## Deskriptive Statistiken ############
describe(Alter)

#   var n mean   sd median trimmed  mad min max range skew kurtosis   se
# 1   1 6   25 3.03     24      25 1.48  23  31     8 1.18    -0.38 1.24

describe(X.QuantitativeKorr.)

#   var n  mean   sd median trimmed  mad min max range  skew kurtosis   se
# 1   1 6 14.83 4.88   15.5   14.83 5.19   7  20    13 -0.41    -1.56 1.99

# Vergleichsdaten Normstichprobe (Alter, Bildung kontrolliert)
# Standartwerte für 21-25 jährige Gymnasiasten M = 14,05 sd = 4,89 N = 478

# --> keine signifikanten Unterschiede zu erwarten

########## Mittelwertsvergleich t-Test ############

#Test auf Varianzgleichheit

x <- rnorm(6, mean = 14.83, sd = 4.88)
y <- rnorm(478, mean = 14.05, sd = 4.89)
var.test(x, y)

# F test to compare two variances
#
# data:  x and y
# F = 0.1322, num df = 5, denom df = 477, p-value = 0.0301

# --> Von Varianzengleichheit kann ausgegangen werden

t.test(X.QuantitativeKorr., mu=14.05, alternative="two.sided", var.equal = TRUE)
```

```
# One Sample t-test
#
# data:  X.QuantitativeKorr.
# t = 0.3936, df = 5, p-value = 0.7101
# alternative hypothesis: true mean is not equal to 14.05
# 95 percent confidence interval:
#   9.717222 19.949444
# sample estimates:
# mean of x
#  14.83333

#  p-value = 0.7101 --> nicht signifikant

# Hypothese, dass man mit Vorwissen besser abschneidet, kann vorerst als widerleg angesehen werden.
# Beachte aber: extremer Stichprobengrößenunterschied beeinflusst die Berechnung nicht unerheblich.

# Datensatz ausbinden
detach(daten2)
```

A.3 Think aloud protocols

148, 84, 52, 36, 28, ?; ()

VP_01 Also. 148, 84 ist auffällig, also wird der erste Schritt geteilt 2 sein. Der nächste schaut nicht nach einem geraden Teiler aus, also von 84 auf 52. Dann subtrahier ich das mal und komm auf 32 also hätten wir geteilt 2 minus 32. das passt dann doch nicht ganz weil als nächstes kommt wieder ein Additionsprung und es wird 16 abgezogen. Das ist dann eher ein Zeichen, dass die Differenz sich verändert, Beim ersten hatten wir dann -84 beim zweiten -32. Halt das erste ist ja gar nicht -84, ich bin wegen der 4 darauf herein gefallen. Die Differenz ist 64. Ok, jetzt ist es eindeutig. -64, -32, -16, -8, passt soweit und dann kommt als näxhstes -4, 28 -4 ist 24.

VP_02 Also jetzt schau ich erstmal ob ich ein Muster find. Ich sehe, dass alle absteigend sind und dann ... vielleicht käm ich mit zweiten weiter weil mich die Zahlen so abschrecken.

VP_03 ok, 148 84, vllt geteilt durch 2 ..zu viel. 84-52 sind 32, passt auch nicht. 6mal6 36,5mal5 25, geht auch nicht...kann ich zum nächsten gehn?

VP_04 ok von 148 auf 84: ich weiß, dass keine modulo operationen oder so in frage kommen, deswegen ist es wohl ein bruchteil davon, aber hm, seltsamer bruchteil; mal kurz die 84 von der 148 abziehen, dann komm ich auf 64... dann ist es vllt kein bruchteil, sondern 64 weniger, und dann fehlen 32 auf 52, ok, das ist die hälfte von 64, dann 16, das bestätigt sich mit den 36, dann fehlen 8 und hier fehlen 4, ich schreib die 24 daneben

VP_05 64 weniger oder, 32 weniger, des nächste 16 weniger, dann ist es 24. Ihr habt die 9 vergessen, die 4 fehlt ja auch, des nächste müsste denk ich die 11 sein, weil zahlen die durch Quadrate entstehen rausfallen

VP_06 –

Vorgehen –
Als erstes wird vor allem die absteigende Reihe erkannt. Die möglichen Operatoren werden auf div, mod und sub eingeschränkt. Große Zahlensprünge werden auf div zurückgeführt (div 2). Wenn Subtraktion getestet wird, stellt man nach den ersten zwei bis drei Subtrahenden identifiziert wurden, wird die Regelmäßigkeitkeit erkannt.

2, 3, 5, 6, 7, 8, 10, ?; ()

VP_01 Also 2, 3, 5 wäre +1, +2, +1, +1, wäre nicht ganz so regelmäßig von den Additionen zwischen den Zahlen, also jetzt vielleicht eher was anderes, +1, +2, +1, +1, +1, oder vielleicht doch, es könnte auch ein rhythmus sein mit + das es was gleichmäßiges ist wie, 112112 oder sowas, aber eigentlich klingt das nicht so. 5 ist aber auch die Summe aus 1 und 3, also könnte es auch sein, dass es mit der vorherigen Zahl zu tun hat. Also nach dem Motto 2 +3 = 5, was ist dann mit der 6 ,d ie wäre ja wohl eher 2 mal 3, das passt mit der 5 nicht zusammen, die 7 lässt sich auch hier nicht als summe darstellen, das spricht also dagegen, dass es hier mit der Summe zu tun hat. Es könnte natürlich sein, dass es hier noch größere Sprünge macht von der 8 auf die 10 wärens 2, auch irgendwie merkwürdig. Es könnte auch von einer Basiszahl ausgehen und sich dann in Intervallen annähern oder entfernen vielleicht, irgendwie -3 -2 +1 +2 +3, ne das würde nicht ganz passen oder es könnte etwas mit Primzahlen zu tun haben aber (die Intervalle) 1 ist ja keine Primzahl. Also von der 5 Ausgesehen -3 -2 +1 +2 +3 +5, erschließt sich mir auch nichts. (wird zurückgestellt) Fortsetzung: 2,3,5. Vielleicht ist das einder der Aufgabentypen, die ich erst später gecheckt habe also 2+3 = 5, Aber ich glaub in die Richtung hab ich auch schon überlegt 5 und 2, was ist denn mit der 6. Ne die ist zu doof, also doch irgendwas mit der Steigerung. Irgendwie +1+2+1+1, Warum geht das rauf und wieder runter, warum verebbt das und steigt am Schluss wieder um 2 an kann natürlcih sein das das ein Rhythmus ist der etwas holperig ist +1+2 +1+1+1+2+1+1+1 oder sogar steigern, aber das wissen wir nicht, weil die Reihe zu kurz ist. Dann schätz ich es ist wider +1, (also +1+2+1+1+1+2+1) also 11.

VP_02 Also die Intervalle wären 121112, und das würd ich acuh sagen, dass das das Muster ist. Weis ich aber nicht genau, da die Reihe abreist, aber es könnte mit der 11 weitergehen

VP_03 2+1, 3+2, 5+1, 6+1, 7+1, 8+2, 10, nach 3mal +1 kommt +2 und danach wohl wieder +1

VP_04 2,3,5,6,7,8,10, aha, 1mehr 2mehr... 3mal 1mehr, 1mal 2mehr...hm komplexe folge hier, ich fürchte fast, dass ich jetzt hier nicht genug daten hab, um meine hypothese zu testen, eine möglichkeit wäre: 1 einser schritt, 3 einser schritte, und dass dann hinter der 10 5 einser schritte kämen. aber, es könnte theoretisch auch was anderes sein. ich weiß auch nicht ob die reihe hier bei 0 losgeht, wenn ich das hätte, wüsste ich dass der als erstes ein einser schritt ist. das kann ich mit den daten glaub ich nicht endgültig lösen, ich kann aber nen tip abgeben, und mal die 11 hinschreiben

VP_05 –

VP_06 ich schau mir wieder die abstände an, 1 1 1 , da müsste die nächste Zahl 11 sein

Vorgehen –
Addition als erste Wahl. Zahlenreihen ist wird als aufteigend erkannt. Unregelmäßiger Anstieg führt zum Verwerfen von gesteigerten Argumenten. Muster wird gesucht und gefunden.

2, 3, 5, 9, 17, ?; ()

VP_01 Das schaut wieder einfacher aus, nach einer Steigerung in der Addition. +1 +2 +4 +8, also verdoppelt sich immer die Addition zwischen den Zahlen, also müsste jetzt + 16 kommen als nächstes. 17 + 16 wäre dann 33.

VP_02 2, 3, 5, 9, 17. Da würd ich sagen, dass man es mit den Exponenten zu tun hat, weil die Unterschiede, ... das wächst einfach, 1 2 4 8. Also eigentlich immer mal 2, also wäre das hier +16. Oder? Überprüfen. Ja 33.

VP_03 +1, +2, +4 ,+8, von 4 auf 8?, 2mal2 sind 4, 4mal2 sind 8, 8mal2 sind 16, 17+16, also 33

VP_04 das sieht nach einer exponentiellen steigerung aus, 1, 2, 4, 8, und zwar genau exponentiell, dann folgt 16, ich schreib die 33 hin

VP_05 was zur Hölle, die Zahlen plus den Vorgänger, 16 und 17 is 33

VP_06 2+3 gibt 5, ah 1 2 4 8 dann is die nächste zahl mit abstand mit 16 also 33, oder is das mal 2 minus 1

Vorgehen Als erstes wird auch hier die aufsteigende Reihe und der regelmäßige Anstieg der Reihe erkannt. Vorallem der Operator add tritt in den Vordergrund, da die Zahlen zu klein für ganzzahlige Multiplikation sind. Hier findet möglicherweise ein schneller Abgleich mit stark einstudierten multiplikationswerten (Grundschulwisse) statt, um diesen Operator auszuschließen. Beim Erkennen der Regel hinter der Steigerung der Addenden, könnten auch fest gelernte Zahlenkombinationen eine Rolle spielen.

Alternativ: Wird über den mult Operator versucht, die Distanz der Zahlen zu füllen. Feinabstimmung erfolgt dann erst über die Addition (mal 2 plus 1). Diese Vorgehen spricht weniger für Musterabgleich für ein überwinden einer synthese barriere (Dörner) Vorgehen, Zweiprozesstheorie Kluwe -¿ automatisiert vs kontrolliert (psychostud vs. neuling)

3, 7, 15 ,31 ,63 , ?; ()

VP_01 3,7,15. Da hätten wir +4+8+16, das schaut schon wieder so aus +32, also da verdoppelt sich die Summe die Addiert wird und nach +32 kommt +64. 63 + 32 wäre dann eins weniger als 128. 127.

VP_02 3, 7, 15 ,31 ,63. Ich schaue jetzt ob mir ein augenfälliger Bezug auffällt. *2 oder irgendsowas, wo ich die Zahlenreihe kenne aus der Grundschule. Ich suche eigentlich nach Mustern, markiere es und schaue ob es sich wiederholt. Aber bei Zahlen ist es schwierig Muster zu suchen. *2 +1, das sieht doch gut aus. Jetzt hab ich gesehn, dass 15 und 31 *2+1 ist und, dass es bei den anderen auch passt. Nochmal überprüfen. 126. 127. Aber es ein bisschen Glück dabei, hab ich das Gefühl ob man da grad draufkommt oder nicht.

VP_03 3+4 sind 7, +8 sind 15, +16 sind 31...plus 32 sind 63...4mal2 sind 8, 8mal2 sind 16, 16mal2 sind 32, 32mal2 sind 64, also 127

VP_04 von der 3 auf die 7 haben wir 4 mehr, dann 8 mehr, dann 16 mehr, 32mehr, hier 64 mehr, dann 127

VP_05 2 hoch 2 ist 4 +3 is 7, bei der nächsten addierst du , ne, immer des doppelte plus 1

VP_06 ja sieht man ja sofort, mal 2 plus 1, also is die nächste zahl 127

Vorgehen Anstieg, + , patternmatch im argument. oder: große Anstieg wird zuerst überbrückt und dann fein tuning.

9, 20, 6, 17, 3, ?; ()

VP_01 So 9,20 wäre +11, -14 +11 -14, das schaut nach einer Regelmäßigkeit aus, dann müsste jetzt wieder + 11 kommen 3+11 sind 14.

VP_02 Also da zum Beispiel fällt halt gleich auf, dass die so unterschiedlich groß sind und das die Reihen dann zwischen geschoben sind. Beider wären -3 aber fangen halt wo anders an. 9,6,3 und 20,17,14.

VP_03 11 differenz, dann 14, dann wieder plus 11, minus 14, also müsste wieder +11 folgen, 3+11 sind 14

VP_04 von 9 auf 20 sind 11mehr, 14 runter, wieder 11 mehr..dann sind wir bei 14 mir geht grad durch den kopf, dass sogar die ergebnisse ne zahlenreihe bilden könnten...aber die sind vermutlich eher ausgewählt

VP_05 –

VP_06 also die erste zahl immer minus 3 und die zweite auch, dann kommt da raus 14

Vorgehen –
Ertmal mit pasendem Operator ran und dann die Argumente tunen/muster erkennen: Oder direktes aufsplitten in 2 reihen

0, 1, 4, 13, 40 ,?; ()

VP_01 0, 1, 4, +1, allerdings wenn ne 0 dabei ist muss man aufpassen, kann auch immer was mit quadratsummen sein, aber die 13 passt da nicht rein, die ist ja nicht die Quadratsumme von irgendwas. Also doch eher klassisch vielleicht +1,+4,+9, aha, hier ist die Quadratsumme vielleicht dabei, dann haben wir noch +27, ah ok, es wird also irgendwas mit 3 sein. 0+1 wäre dann ja 3 hoch 0 und 3 hoch 1 und ne 1 mit drauf , 3 hoch 2 , 3 hoch 3, und 3 hoch 4 dann wären wir bei 121.

VP_02 Die steigt auch ziemlich schnell an, eigentlich. Die fällt mir jetzt bissl schwer, weil ich zwischen 4 und 13 nicht gleich den Zusammenhang seh. Zwischen 13 und 40 auch nicht. (lange Pause) Ich schreib mir grad die Hypothesen daneben, ein bisschen, aber ich komm nicht so richtig weiter. Also zum Beispiel *4 und man zieht noch eine bestimmte summe ab. Ich geh zur nächsten.

VP_03 0+1, 1+3, +9 sind 13, +27 auf 40, 1mal3 sind 3mal3 sind 9mal3 sind 27mal 3...81

VP_04 von 0 auf 1 auf 4 auf 13 auf 40, das sieht...nee doch nicht.... meine erste idee war, dass es mit quadratischen zahlen zu tun hat, ah das dreifache plus 1 jeweils...dann das 3fache von 40 sind 120 plus1 sind 121

VP_05 –

VP_06 ah, des is immer um 2er potenz, also hoch irgendwas kommt dazu, 0+1, dann 1+2, dann plus 9, dann plus 27 des nächste mal plus 36, dann sind wir bei 76

Vorgehen wieder + als operator gewählt und dann analyse der Argumente

99, 18, 36, 9, 18 , ?; ()

VP_01 99, 18, 99 ist ja 11*9 und 18 ist 2*9 und 36 ist 4*9, da hätte man dann 1*9 und 2*9 wieder. Aber von der 11 auf 2 erscheint mir ein bisschen merkwürdig. Warum springt es von 11*9 auf 2*9 und danach eher im ähnlichen bereich. 11 auf 2 und dann haben wir wieder mal 2, also das ist komisch finde ich. Vielliehct was anderes, aber mit der Differenz scheints mir auch nichts zu sein. Differz 81, wobei, dass geht natürlich auch wieder durch 9. -81, + 18 aber dann acuh wieder die Frage, was ist das Prinzip dahinter. Also vielleicht doch wieder womit ist die 9 multipliziert worden 11 2 4 1 2. da fällt mir jetzt aber auch keine Regelmäßigkeit unter den Zahlen auf. Wir haben dann noch etwas wie 2*18 ist 36. Des wäre dann schon komischer hin und hersprung was da zusammen kommt, die 9 ist ja auch einzeln drin aber trotzdem ist das auch nicht so das wahre hmm aber das 3er und 9er Prinzip ist schon ziemlich eindeutig. Außer es ist ne Finte. Dann seh ich auch nichts anderes was gut passen könnte. Könnte natürlich auch sien, das sich die Zahlen zueinander in ähnlichen Multiplikationsverhältnissen entwickeln, also, aber die 99 zur 18 passt schon nicht weil 11 zu 2, wie sollen die zueinander stehen, also hätten wir die 11 2 2 4 2, also irgendwie passt das alles nicht zur 11. Natürlich könnte irgendwas sein, wo am ende wieder 11 rauskommt. Ich lass die Aufgabe mal aus. Fortsetzung: Hier würde mich schon interessieren woher die 99 kommt, denn die sticht so arg aus der Reihe. Vielleicht ist es sowas wie 10*10-1, aber was wäre dann die nächste 2*10 -2, das wäre komisch. Vielleicht was mit anderen Zahlsystemen, das ist es aber denk ich auch nicht, dafür ist es auch zu weit weg. Dann haben wir hier noch was mit der 4 oder der 2, bei der 36 geht ja die 4 und auf 45 ..., aber das wäre auch nicht ganz plausibel vielleicht die 6, aber das passt auch nicht gnaz rein wegen der 9 und der 99, manchmal ist es auch so Rhytmen die überspringen eine Stufe, die 99 und die 36, ja das schaut gut aus, die 18 und die 9 passen ja auch ganz gut zusammen, Aber die 99 auf die 36 geht ja auch nicht super gut auf, vielleicht umgedrehte Ziffern, bei der 18 und der 9 wären es 9 aber das passt ja nicht. Aber es kann auch etwas mit der Quersumme sien. Ja das schaut gut aus. Wobei die Quersumme von 99 ist zwar 18 aber die von 18 nicht 36. aber von 36 wieder 9. Vielleicht ist der zweite Sprung immer auf die Quersumme drauf. was würde passen... *2. genau Die Quersumme 36 ist 9 *2 ist 18 und die Quersumme wäre wieder 9.

VP_02 Da fällt auf, dass da zwei 18en sind. Ah, können auch Quersummen vielleicht drin sein. *3 ...., Ja ... es müsste theoretisch mit der 9 weitergehen. Also 9, 18, 3 und 6, 9.

VP_03 99 und 18?... 18mal 2 sind 36, geteilt durch 2 und nochmal 2 also durch 4, dann 9mal2 sind 18; nochmal die allererste zahl 99geteilt durch 4..bleibt eines übrig, 99 sind auch 3mal33, 99minus18 sind 81, 81minus18 geht aber auch nicht können's auch kommazahlen sein, oder sind's nur ganze?

VP_04 sind alle durch 3 teilbar die zahlen hier, vllt ist das irgendein hinweis, oder es ist die quersumme, genau die quersumme, von 99 sind 18, das doppelte ist 36...quersumme wieder die 9, da wären wir also quasi bei nem fixpunkt angekommen, ab da immer 9 und 18 im wechsel

VP_05 –

VP_06 ja das sind alles Vielfache von 9, das is die Hälfte von dem und das die Hälfte von dem, aber das nicht von dem, keine ahnung, ok nächstes

Vorgehen 9 als Teiler wird sofort erkannt. alternativ: 3 als Teiler –¿ Hinweis auf Quersumme

10, 45, 15, 38, 20, ?; ()

VP_01 10,45,15, da hätten wir mal 3 geteilt durch 3, dann hben wir die 38, fällt aus der reihe, das kann es also nicht gewesen sein. Also haben wir auch +35 -20 + 23 -18, das erscheint mir noch fruchtbarer, das nähert sich irgendwie so ein bisschen an. Ne doch nicht ne. Das nähert sich doch nicht an, es gibt ja solche Zahlenreihen wo + und - im Wechsel ist, aber im Endeffekt kommen sich dann die Zahlen immer näher oder gehen weiter auseinander. Dann vielleicht anders. Wir haben einen dreierTeil vorne drin aber später nicht mehr, aber der 5er Teiler kanns auch nicht sien, weil die 38 tanzt aus der reihe. Mit der einfachen Addition passt auch nicht ganz, also das zwei vorherige immer was mit der nächsten zu tun haben, passt auch nict so gut, höchstens, dass der Teiler was mit der nächsten zu tun hat also mal 45 aber das geht gar nicht schön auf weil es sind immer nur ganze Zahlen bei diesen Reihen, würde also auch nicht so gut hier reinpassen. Der gemeinsame Nenner also 2 und 3 aber nen gemeinsamen Teiler haben die gar nicht und mit den Steigerungen. Vielleicht probier ichs mal mit den gnaz normalen Additionen, dass hab ich bei dieser Reihe glaub ich noch nicht gemacht. +35 -20 , ach doch hab oich ja schon gemacht was war es nochmal +23, dann haben die also auch keinen gemeinsamen Teiler. dann entwickeln die sich vielleicht doch irgendwie also -30 +23, Vielleich entwickeln sich ja die beiden reihen einzeln die Addition und die Subtraktion getrennt voneinander. Also wir hatten +35 -18 +23, das ist ein Sprung um 12., Ja genau das kommt gut. von 10 auf 45 haben wir +35 und +23 das sind 12 weniger und von 45 auf 15 gehts 20 runter und von 38 auf 20 gehts 18 runter. Achne es geht ja 30 runter und 18 also auch 12 weniger. So um das jetzt zu einer Reihe zu abstrahieren. Es geht Aufwärts,Abwärts,Aufwärts,Abwärts dann müsste es wieder Aufwärts gehen, da hatten wir zuletzt von 15 auf 38 waren es 23 dann 12 weniger wären 11 und 20 + 11 wären 31.

VP_02 So die nächste fängt mit einer 10 an, da sind auch wieder 0er und 5er, die sich häufen, also ... 10 15 20 wäre einer Reihe 45 38 wäre eine Differenz von 7, vielleicht kann man jetzt da 7 abziehen. Dann würde 31 hinkommen. Also zwei Reihen ineinander gesteckt. ich lass das mal so stehen.

VP_03 -

VP_04 hm....10 auf 45 ist mir jetzt zu schwierig, ich fang mal mit dem nächsten an, ein drittel, bzw 30 weniger, von 15 auf 38, das ist schon eingeschränkt, damit kann ich eher was anfangen, das doppelte plus 8... ah oder plus 4 und dann mal 2, also 10+5 mal 3 ist die 45 dann hat man ein drittel, dann plus 4 mal 2 wär die 38, dann komm ich aber durch das drittel hier nicht drauf, dann irgendwas anderes, von der 38 auf die 20 ist nochmal schwieriger... dann mach ich hier das dreifache + 15... nee, bisher ist es ja immer so gewesen: ein schritt nach oben und einen nach unten, eines ne punkt- das andere ne strichrechenart... da muss ich passen, da verlier ich zu viel zeit (Fortsetzung) so jetzt könnt ich mich noch um das letzte kümmern, vllt sind mir noch ideen gekommen zu 8. muss den schritt von 45 zu 15 und den von 38 zu 20 in beziehung setzen, die 45 und die 38 kann man schon in nen zusammenhang bringen, das eine ist 3mal15, das andere 2mal 19, vllt gibts hier auch nen faktor, der nicht zu buche schlägt, hm, plus 5 und das dreifache und von der 45 zur 15 ist es dann ein drittel plus 0, und hier ist es die hälfte plus 2 und dann wärs das einfache plus4, das interessiert mich aber gar nicht, was ich hier mache ist den schritt wieder nach oben zu gehen.....dann ist es die 25

VP_05 –

VP_06 die haben kein gemeinsames Vielfaches. die 38 fällt irgendwie aus dem Muster, vielleicht alle garaden zahlen plus 5 und alle ungeraden zahlen minus 7, dann wärs 31

Vorgehen nicht monoton -¿ Operator wechsel oder 2 reihen

4, 5, 7, 11, 19, ?; ()

VP_01 4, 5, 7, 11, 19, das schaut erstmal nach Primzahlen aus. Aber die 4 fällt aus der Reihe, dann ist es das nicht +1 +2 +4 +8. Ja das schaut nach ner klassischen Steigerung und Addition aus. Nach +8 kommt dann + 16 und das wäre dann 35.

VP_02 4, 5, 7, 11, 19. Da wird einfach immer eine Zahl mehr dazu addiert. 1 2 ... Ich kontrollier einfach nochmal 1 2 4 11. Oh doch nicht. Nochmal 2 4 8. Ah. Doch nicht so schlecht, dann müsste das eigentlich mit 16 weiter gehen. Das errinnert mich an eine Reihe von vorher. ... Ja ich lass es, also 35.

VP_03 -

VP_04 1,2,4,8, dann kommt als nächten 16 und wir sind bei 35

VP_05 –

VP_06 plus 1, plus 2m plus 4, plus 8, 16 also 35.

Vorgehen Primzahlenmuster wird erkannt un verworfen. Dann Standardprozedur Operator anwenden und reihe in den argumenten erkennen.

2, 5, 9, 19, 37 , ?; (75)

VP_01 2, 5, 9, 19, 37. 37 ist glaub ich 2*19 oder so. Achne ist doch was höheres, das wäre dann ... . Schauen wir erstmal was die Addition so ergibt. +3,+4,+10, das schaut mir auch nicht nach ner guten Reihe aus. Was ist denn die letzte. Manchmal verrät ja die größte Differenz am meisten durch die Steigerung, da haben wir also + 18, dass könnte wirklich was sein. Also +3,+4,+10, Ne doch nicht. Was könnte es noch sein. 2*5+9 wäre ja 19, aber 5*9 wäre ja schon 45, das kann es also auch nicht sein. Vielleicht wird auch immer was verdoppelt und dann draufgezählt. 2*2+5, 2*5+9, 2*9+19 usw. Also muss dann 2*19 +37 folgen, 38+37 sind 75.

VP_02 2, 5, 9, 19, 37. Die hier ist wieder aufsteigend. Hier ist halt wieder so komisch, das es für mich so krumm aussieht. Ja die ist schwieriger, da mir nichts ins Auge springt. 18, 10, 4, also ich schaue jetzt ob ichs durch Addition irgendwie lösen kann. 3. Ich schau mal nach einer anderen Gemeinsamkeit, Durch "mal" vielleicht. ... Ich probiers mal mit ner anderen vielleicht.

VP_03 -

VP_04 3mehr, 4, dann 10, 37? das doppelte +1 das doppelte -1...und jetzt das doppelte +1, ja sind 75

VP_05 –

VP_06 oh, das is immer um ne 2er potenz addiert, ne, auch nicht, ich hab keine Ahnung

Vorgehen Lücken zu groß für Addition. Multiplikation und Feinabstimmung. Argumentsreihe wird dann (nicht) identifiziert.

2, 5, 8, 11, 14, 17, 20, ?; ()

VP_01 2,5,8, also +3+3+3+3, dann wird wahrscheinlich 23 als nächstes kommen.

VP_02 Die ist auch aufsteigend, aber länger. Das sieht so aus als würde man da immer nur 3 addieren. Ja. Das wundert micht jetzt ein bisschen, weil ich das Gefühl hatte die werden immer schwerer. Vielleicht, hab ich auch etwas übersehen. Also ich denke es wird einfach immer 3 addiert.

VP_03 -

VP_04 3 und 3 und ....immer 3mehr

VP_05 –

VP_06 ne keine lust mehr

Vorgehen –
Addition und erkennen des Musters hinter der Argumentsliste

1, 3, 6, 8, 16, 18, 36, ?; ()

VP_01 1,3,6, die kommt mir bekannt vor, ich glaube die war schon in der letzten Aufgabe dran. +2+3+2 ,ne *2 haben wir dabei +2*2+2*2+2*2 also +2 38.

VP_02 Also 3 6 8 16 18 36. Das klingelt ein bisschen. Das muss irgendwas zu tun haben, ... mit ... Also mir fallen die Quadratzahlen auf 16 und 36. Aber ich muss erstmal probieren. 3, 6, *2. Ah das könnte sein *2+2. Ah genau. Aber das erklärt nicht ganz wie man von der 1 auf die 3 kommt. da muss man dann irgendwie mit mal 3 draufkommen +2. Jetzt hab ich soviel rumgekritzelt, dass es unübersichtlich wird. Ich schreibs nochmal ab. Also da knabber ich ein bisschen dran, weil die Zahlen sind entweder durch 6 oder durch 8 teilbar und zum teil auch durch 3. die 16er ist so eine Ausnahme. Ah doch ich hatte es ja schon. Man macht immer +2*2, das zieht sich dann durch. 38. Nach meiner Rechnung müsste es wieder +2 sein.

VP_03 -

VP_04 ok, 2mehr, das doppelte, 2mehr das doppelte...38

VP_05 –

VP_06 –

Vorgehen Vielfache werden erkannt. Anwendung der Operatoren (inkl. Feinabstimmung) dann Muster der Argumente

33, 30, 15, 45, 42, 21, 63, ?; ()

VP_01 33, 30, -3. 30 zu 15 ist geteilt 2, die Reihe war auch auf jeden Fall schon auf dem letzten Blatt. Auußer es ist noch ein Trick dabei, aber schaut erstmal so aus. -3 /2 *3 -3 /2 *3, dann kommt jetzt wieder -3, 63-3 = 60.

VP_02 Das sieht so aus, als würde da immer was durch 2 geteilt werden. Zwischen der 30 und der 15 und der 42 und der 21 auch. die anderen -3. Genau dann wird noch irgedwann 3 abgezogen. Dann würde hier wieder -3 kommen also 60. Ich hab mir hier Bögen zwischen den Zahlen gemacht, und welche Operation dahinter steht. Dann kontrollier ich das nochmal -3/2*3-3/2*3. Dann kommt also wieder -3.

VP_03 -

VP_04 3 weniger, die hälfte, das dreifache, 3 weniger, die hälfte, das dreifache, dann drei weniger sind 60

VP_05 –

VP_06 –

Vorgehen 2er Teiler wird erkannt. Feinabstimmung. Muster im Argumentwechsel

25, 27, 30, 15, 5, 7, 10, ?; ()

VP_01 25, 27, +2+3/2/3 dann wieder +2+3, dann kommt als nächstes, was kam nochmal als erstes. /3 nein /2. Beim anderen gab es auch so eine Aufgabe da kam dann 2,5 raus. also hier müsste 5 kommen, weil nach dem aufaddieren kan immer ein geteilt durch 2.

VP_02 Da fällt halt auf, dass es absteigend und aufsteigend ist. 30 15 5 schaut aus nach so einer 5er Reihe, dann ist aber die 7 so komisch reingezogen. 10,30 Also ich such jetzt wieder nach einer einfachen Beziehung zwischen 2 Zahlen, die sich vielleicht wiederholt. +2 würde 25,27 und 5,7 erklären. Also hier könnte man sagen +2+3 und dann /2. Ich mach mal so Bögen wie oben. Dann hab ich +2+3 dann hätten wir wieder /2 also 5. Nochmal überprüfen +2+3/2. Ah und hier noch /3. Aber stimmt trotzdem.

VP_03 -

VP_04 2mehr, 3mehr, die hälfte, ein drittel, 2mehr, 3mehr, die hälfte sind 5, ein drittel kann man gar nicht berechnen, hm

VP_05 plus 2 plus 3 geteilt durch 2 geteilt durch 3, is 5. minus 2 geteilt durch 2, wo bin ich jetzt? laut denken is Schwachsinn, mal 2 minus 1, mal 2 plus 1, mal 2 minus 1, 74 - 1, 23 hoff ich

VP_06 –

Vorgehen Kleiner Anstieg +, großer Abfall "'/"'

11, 8, 24, 27, 9, 6, 18, ?; ()

VP_01 11 auf 8 sind -3, dann *3,+3,/3. Ok das ist dann ja einfach, +3*3 und ann muss wieder -3 kommen. 18-3 = 15.

VP_02 Also mir ist grad ne 9er Reihe in den Kopf gekommen wegen der 27 der ... aber wohl doch nicht. Wegen der 27 und der 18. Das sind ja alles einstellige und zweistellige Zahlen durchmischt. Da würde ich dann nach einer ???? schauen. Ich seh aber keine Anhalt für einen ????. 8 und die 24 also ... ich find jetzt nichts "ah das sind alles Zahlen, die in einer 6er Reihe vorkommen" deswegen schau ich ob ich andere Beziehungen finde. 8 24 wäre *3, 6 und 18 wäre auch *3, 24 27 wäre eigentlich +3, jetzt schau ich ob ich das noch irgendwo find. Das dann nach der 27 die 9 kommt das wäre dann auch noch /3. Vielleicht bresteht es auch aus *3-3/3. Also +3*3+3/3 wäre schon eine ziemlich lange Reihe eigentlich. Ach -3. Aber trotzdem lang. Ah doch. Ich glaub, das man verschiedene Sachen macht mit der 3. Also -3*3+3/3, -3*3 dann müsste wieder + 3 kommen also 21.

VP_03 -

VP_04 drei weniger, das dreifache, drei mehr, dann sind wir bei 21

VP_05 –

VP_06 –

Vorgehen Entweder hangeln von Zahl zu Zahl oder Zahlen die Zusammenhang haben in der Reihe finden.

Appendix B

Appendix B

B.1 Series.hs

```
{-# LANGUAGE RelaxedPolyRec #-}
module Series where

import MagicHaskeller

series :: Int -> [Int->Int] -> [Int]
series i = scanl (flip ($)) i . cycle


-- checksum concept
checksum :: Int -> Int
checksum i
  | i == 0     = 0
  | otherwise = i `mod` 10 + checksum (i `div` 10)

-- square concep
square:: Int -> Int
square x = x*x


-- myodd, myeven :: Int -> Int
-- myodd x = odd x
-- myeven x = even x


-- mysucc, mypred :: Int -> Int
-- mysucc x = succ x
-- mypred x = pred x


-- just positive operators
mseriespos, mseriespos' :: ProgramGenerator pg => pg
mseriespos  = mkPGOpt (options{constrL=True}) (seriesLibpos ++ smallNats)
-- alternative definition
mseriespos' = mkPGOpt (options{constrL=True}) (seriesLibpos ++ nats)


-- split:: [Int] -> [Int]
-- split = fst (foldr (\x (ys, zs) -> (x : zs, ys)) ([], [])) ++
           snd (foldr (\x (ys, zs) -> (x : zs, ys)) ([], []))



mseries, mseries' :: ProgramGenerator pg => pg
mseries  = mkPGOpt (options{constrL=True}) (seriesLib ++ smallNats)
-- alternative definition
mseries' = mkPGOpt (options{constrL=True}) (seriesLib ++ nats)

seriesLib, seriesLibpos, smallNats, nats :: [Primitive]
seriesLib = $(p [|(series :: (->) Int ([Int->Int] -> [Int]), []::[Int->Int],
(:)::(Int->Int)->[Int->Int]->[Int->Int], (+)::Int->Int->Int, (-)::Int->Int->Int,
(*)::Int->Int->Int, div::Int->Int->Int, checksum::Int->Int, square::Int->Int)|])
seriesLibpos = $(p [|(series :: (->) Int ([Int->Int] -> [Int]), []::[Int->Int],
(:)::(Int->Int)->[Int->Int]->[Int->Int], (+)::Int->Int->Int, (*)::Int->Int->Int)|])
smallNats = $(p [|(1::Int, 2::Int, 3::Int)|])
nats      = $(p [|(1::Int, succ::Int->Int)|])


-- usual version
filterSeries    pred = filterThen pred $ everything (mseries :: ProgGen)
filterSeriesPos    pred = filterThen pred $ everything (mseriespos :: ProgGen)

-- alternative definition
filterSeries'  pred = filterThen pred $ everything (mseries' :: ProgGen)
```

```
{- definitions not using constrL
-- usual version
$(define ''ProgGen "Series" (p [|(series :: (->) Int ([Int->Int] -> [Int]),
 []::[Int->Int], (:)::(Int->Int)->[Int->Int]->[Int->Int], (+)::Int->Int->Int,
(-)::Int->Int->Int, (*)::Int->Int->Int, div::Int->Int->Int, 1::Int, 2::Int, 3::Int)|]))
-- alternative definition
$(define ''ProgGen "Series'" (p [|(series :: (->) Int ([Int->Int] -> [Int]),
[]::[Int->Int], (:)::(Int->Int)->[Int->Int]->[Int->Int], (+)::Int->Int->Int,
(-)::Int->Int->Int, (*)::Int->Int->Int, div::Int->Int->Int, 1::Int, succ::Int->Int)|]))
-}
```