

HANDLING CONDITIONAL CALCULATION ALGORITHMS IN SPREADSHEET ENVIRONMENTS

Mária Csernoch¹, & Piroska Biró^{1,2}

¹*Faculty of Informatics, University of Debrecen (Hungary)*

²*Faculty of Economics, Socio-Human Sciences and Engineering,
Sapientia Hungarian University of Transylvania, (Romania)*

Abstract

First year students of informatics were tested on their problem-solving strategies in handling conditional calculations in spreadsheet environments. The test was carried out in the first week of the students' tertiary studies to reveal how K-12 education supports building schemata, and how students can call them forth when a problem arises. On the one hand, it is revealed that students cannot activate the algorithm behind the presented problems, they are stuck at the level of hard-coding and apply slow thinking approaches at a low abstraction level. This implies that despite the shared algorithm in the background, the results of the tasks decrease with the increases in the level of abstraction. On the other hand, with an increase in the level of abstraction, students prefer algorithm-driven solutions instead of the environment-specific built-in functions. Considering the results from the learning-teaching approaches, teaching programming should be a "whole programming" approach, where the focus is not on the tools of an environment, but on the development of students' computational thinking skills.

Keywords: *Computational thinking, spreadsheeting, SOLO categories, algorithm.*

1. Introduction

The present paper details how first year students of informatics handle conditional calculating problems in non-traditional programming environments. The research questions were how students can recognize and call forth schema (algorithm) stored in the long-term memory and how they can facilitate fast and slow thinking for reducing cognitive load. The skills they have are rather fundamental or rote and the focus is on conceptualizing or coding, on ideas or artifacts (Wing, 2006).

According to the National Curriculum and frame curricula of digital culture in Hungary (OFI, 2012, 2020a, 2020b, 2020c), students start their studies in informatics (Computer Science, CS) in Grade 3 and finish it in Grade 11 (with additional years in Grades 1, 2, 12, and 13 and various number of classes per week). In all grades, programming is compulsory, while spreadsheeting is taught from Grade 8 to Grade 11, and both subfields are compulsory in the maturation exams. In programming courses, conditional calculation is one of the fundamental algorithms covered across several grades. Considering spreadsheeting, conditional calculation is also covered with the conditional (statistical) and database functions. Consequently, the problem addressed in the present paper is thoroughly covered in informatics classes and frequently added to the maturation exams. Furthermore, spreadsheet programming had been introduced as early as mid-2000s (Abraham et al., 2009; Sestoft, 2011; Wakeling, 2007) and expanded in 2014 when Sprego (Spreadsheet Lego) was launched (Csernoch, 2014) and its effectiveness was proven later (Csapó et al., 2019, 2020; Máté et al., 2025). In general, the aim of teaching informatics is to develop students' computational thinking skills (Wing, 2006, 2017), which is in accordance with the purpose of the test.

2. Methods

A paper-based test of five spreadsheet tasks was presented to first year students of informatics after their K-12 studies, finished with a maturation exam. The informatics maturation exam is not compulsory, but 66% of the students completed it. The exams cover both programming and spreadsheet sections, where students should solve problems on a computer.

The "old-fashioned" form of testing is explained by the authors' intention to identify problem-solving strategies and to avoid students' aimless sequences of clicks (Máté et al., 2025), which was proven ineffective in previous studies.

2.1. Sample

First year students of informatics were tested in the first week of their tertiary studies. Altogether, 190 students participated, and their results on their spreadsheet problem-solving strategies were analyzed. The present paper provides the details of the test, the results of the students, and their approaches to the problems.

2.2. Tasks

Five spreadsheet tasks were presented in the test (T1–T5), along with a sample table (Figure 1). The table consists of 5 data fields and 235 records, of which 11 are visible. Furthermore, two variables were highlighted, Cells G2 and G3 that should be used in the formulas.

The tasks were concentrated around conditional calculations. Tasks T1–T3 were to create formulas based on text-based descriptions, while in Tasks T4 and T5, two formulas were presented, and students should explain what the formulas did.

Task T1 was a conditional calculation to check equality with a string constant (hard-coded). This task is considered Level 1. Task T3 was also a conditional calculation, but inequality was checked, and the comparable value is stored in a variable. This was a Level 2 task. Task 3 is an inequality with a variable, but the mathematical operation is changed from sum to average. This is a Level 3 task.

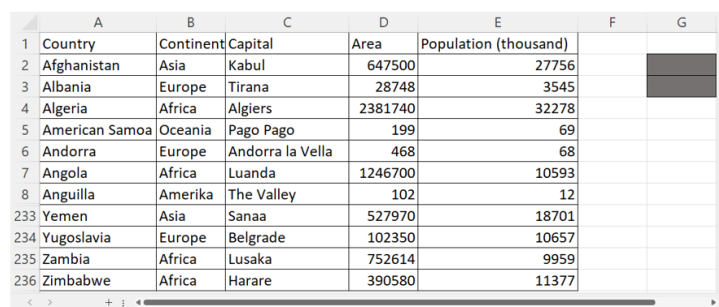
The two given formulas calculated the number of European countries whose first letter is A. There is an AND operation between the two conditions. In Task 4, embedded IF() functions, while in Task 5, a multiplication was used to carry out the AND.

The evaluation of the tasks was based on the SOLO categories of understanding (Csernoch et al., 2015). The same method is used in this case, which is summarized in Section 2.3. The three steps presented in Table 1 formulate the algorithm.

Table 1. The algorithm and sections of codes of the conditional calculation tasks of the test.

algorithm	sections of codes		
	T1	T2	T3
yes/no question	Continent = "Africa"	Population < G2	Area > G3
TRUE branch	1	Area	1
calculation	sum	average	sum

Figure 1. The sample table and the tasks of the test.



	A	B	C	D	E	F	G
1	Country	Continent	Capital	Area	Population (thousand)		
2	Afghanistan	Asia	Kabul	647500	27756		
3	Albania	Europe	Tirana	28748	3545		
4	Algeria	Africa	Algiers	2381740	32278		
5	American Samoa	Oceania	Pago Pago	199	69		
6	Andorra	Europe	Andorra la Vella	468	68		
7	Angola	Africa	Luanda	1246700	10593		
8	Anguilla	Amerika	The Valley	102	12		
233	Yemen	Asia	Sanaa	527970	18701		
234	Yugoslavia	Europe	Belgrade	102350	10657		
235	Zambia	Africa	Lusaka	752614	9959		
236	Zimbabwe	Africa	Harare	390580	11377		

T1: Create a formula that calculates the number of African countries in the table!

Formula:

T2: Create a formula that calculates the average surface area of those countries whose population is smaller than G2!

Formula:

T3: Create a formula that calculates the number of countries whose surface area is greater than G3!

Formula:

T4: What does the following formula do? {=SUM(IF(B2:B236="Europe",IF(LEFT(A2:A236)="A",1)))}

Explanation:

T5: What does the following formula do? {=SUM((B2:B236="Europe")*(LEFT(A2:A236)="A"))}

Explanation:

In general, there were two solutions to carry out the problems. First, the students recognized the algorithm shared with all five tasks. Second, if students did not recognize the algorithm, they could use the built-in functions of spreadsheets that can be COUNTIF(), COUNTIFS(), and AVERAGEIF(), AVERAGEIFS(). However, using the built-in functions one should be aware of the various syntactical rules considering whether equality or inequality is in the condition, whether the IF() or the IFS() versions of the functions were used, and whether the input values arrived from a constant (hard-coding, Tasks 1, 4, and 5) or a variable (Tasks 2 and 3).

The research questions were what problem-solving strategies students could select and whether they recognized the algorithm that the tasks shared or not.

2.3. SOLO categories of understanding

To the presented spreadsheet tasks, the SOLO categories of understanding (Biggs & Collis, 1982a, 1982b; Sheard et al., 2008; Structural Learning, 2025) were formulated in a previous testing series (Csernoch et al., 2015), where the following categories were set up:

- Ignored (I): the task is left empty, no traces of solutions were recognizable,
- Prestructural (P): irrelevant chunks were performed,
- Unistructural (U): one aspect of the problem is grabbed,
- Multistructural (M): at least two aspects of the problem are grabbed,
- Relational (R): more than one aspect of the problem is recognized, and the relationship between them is revealed.

Based on the SOLO categories, the results of the students were calculated on a scale of [0, 4]. The fifth category (Extended abstract) of SOLO does not apply to the test, since no generalization of the problem was asked.

3. Results

In general, it is found that none of the students completed all five tasks at Level R (Relational), while there were 36 students, who scored 0 in all five tasks. Considering all the tasks, the students average is 34%. The students' results expressed with the SOLO categories are presented in Tables 2 and 3 and in Figure 2. In Task T1–T3, the built-in function solution is marked with B, while the algorithm-driven solution with S (Table 2).

Table 2. The number of students achieving the different SOLO categories in the solutions using the built-in or the algorithm-driven solutions.

	T1 B	T1 S	T2 B	T2 S	T3 B	T3 S	T4	T5
I	123	122	162	122	149	139	66	118
P	6	2	11	18	2	11	8	8
U	32	38	16	35	30	18	44	48
M	2	25	1	10	6	18	0	1
R	27	3	0	5	3	4	72	15
average	24.21%	21.71%	6.05%	18.16%	12.11%	15.39%	50.53%	21.97%

Table 3. The number of students achieving the different SOLO categories in the solutions, where the built-in functions and the algorithm-driven solutions are combined in Tasks 1–3.

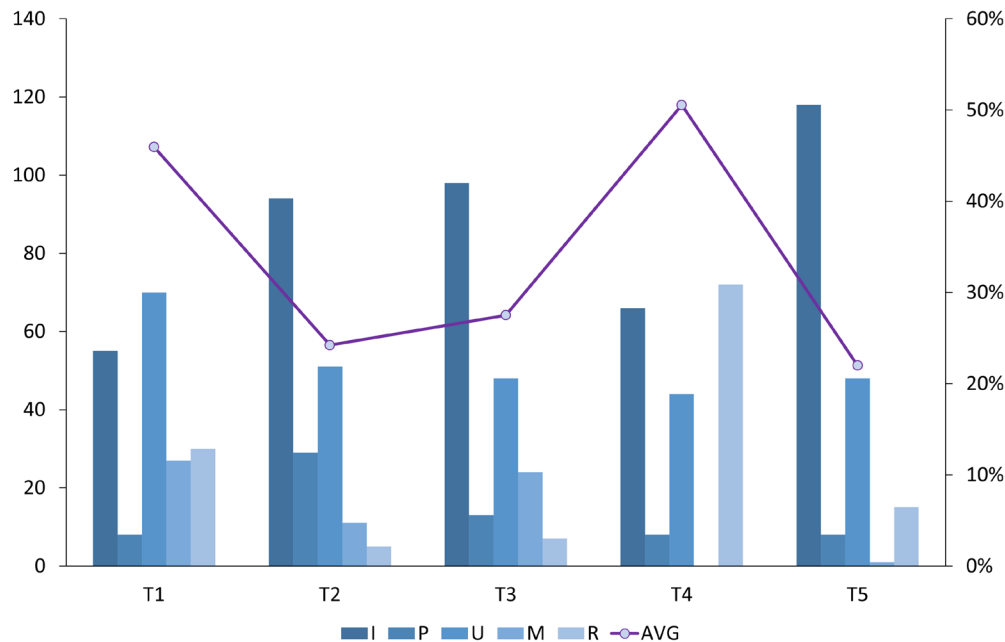
	T1	T2	T3	T4	T5
I	55	94	98	66	118
P	8	29	13	8	8
U	70	51	48	44	48
M	27	11	24	0	1
R	30	5	7	72	15
average	45.92%	24.21%	27.50%	50.53%	21.97%

It is found that in the formula-writing tasks (Tasks 1–3) by increasing the level of abstraction, the results were lower, even though the algorithm was the same. At Level 3 in Task 2, none of the built-in function-solutions were correct. On the other hand, 3, 4, and 5 correct solutions were found in Tasks 1, 3, and 2 as the level of abstraction increased.

It is also found that in Task 1 students preferred the built-in functions, but as the level of abstraction increased, they switched to the algorithm-driven approach, and more correct answers arrived from the algorithm-driven solutions than from their built-in pairs.

These results imply that the algorithm-driven solutions are more general. Consequently, it can be applied to various levels of abstraction.

Figure 2. The results of the five tasks based on the SOLO taxonomy.



Considering the decoding of the given formulas, the embedded IF() solutions were solved 50.53% on average, while the same algorithm presented with the multiplication reached only 21.97%. This result implies that students do not understand the connection between the two coding solutions of AND. Furthermore, only half of the students were able to recognize and explain the algorithm coded with an embedded IF() solution.

4. Conclusions

The results of the test demonstrate that first year students of informatics struggle with conditional calculation algorithms presented in spreadsheet environments, even though these algorithms are taught from a very early age in various programming environments. We can conclude that at the low abstraction level tasks, students focus on spreadsheet tools rather than on the algorithms behind the tools. However, when the level of abstraction increases, students prefer algorithm-driven solutions.

It is also remarkable that the hard-coded Task 1 caused the least problem, while the tasks with variables seem much more challenging. This might be due to the syntactical restrictions of the built-in functions and to the lack of schemata stored in long-term memory. The combination of these problems leads us to the hiatus revealed in the test. On the one hand, students cannot transfer knowledge through digital environments, which implies that their knowledge is tool-centered and tool-tied. On the other hand, it seems that no schemata are stored in long-term memory. Consequently, students primarily solve similar spreadsheet problems with slow thinking, risking erroneous and uncompleted results. Considering the results from the learning-teaching approaches, teaching programming should be “whole programming” approach (Soloway, 1993), where the focus is not on the tools of an environment, and problems should be “expressive and useful, easy to learn and use, embedded in reach cognitive context”, which is in accordance with the requirement of the development of students’ computational thinking skills.

References

- Abraham, R., Burnett, M., & Erwig, M. (2009). Spreadsheet programming. *Wiley Encyclopedia of Computer Science and Engineering*, 2804–2810. <https://doi.org/10.1002/9780470050118.ecse415>
- Biggs, J. B., & Collis, K. F. (1982a). *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*. Academic Press.
- Biggs, J. B., & Collis, K. F. (1982b). Origin and description of the SOLO taxonomy. In *Evaluating the quality of learning* (pp. 17–31). Academic Press. <https://doi.org/10.1016/b978-0-12-097552-5.50007-7>
- Csapó, G., Csernoch, M., & Abari, K. (2019). Sprego: Case study on the effectiveness of teaching spreadsheet management with schema construction. *Education and Information Technologies*, 25(3), 1585–1605. <https://doi.org/10.1007/s10639-019-10024-2>
- Csapó, G., Sebestyén, K., Csernoch, M., & Abari, K. (2020). Case study: Developing long-term knowledge with Sprego. *Education and Information Technologies*, 26(1), 965–982. <https://doi.org/10.1007/s10639-020-10295-0>
- Csernoch, M. (2014). *Programming with spreadsheet functions*. Műszaki Könyvkiadó.
- Csernoch, M., Biró, P., Máth, J., & Abari, K. (2015). Testing algorithmic skills in traditional and non-traditional programming environments. *Informatics in Education*, 14(2), 175–197. <https://doi.org/10.15388/infedu.2015.11>
- Máté, D., Kiss, J. T., & Csernoch, M. (2025). Cognitive biases in user experience and spreadsheet programming. *Education and Information Technologies*. <https://doi.org/10.1007/s10639-025-13392-0>
- OFI. (2012). *National base curriculum*. http://ofi.hu/sites/default/files/attachments/mk_nat_20121.pdf
- OFI. (2020a). *Digital culture: Frame curriculum for grades 9–12*. https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/Digitalis_kultura_K.docx
- OFI. (2020b). *Digital culture: Frame curriculum for grades 1–4*. https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/Digitalis_kultura_A.docx
- OFI. (2020c). *Digital culture: Frame curriculum for grades 5–8*. https://www.oktatas.hu/pub_bin/dload/kozoktatas/kerettanterv/Digitalis_kultura_F.docx
- Sestoft, P. (2011). *Spreadsheet technology*. IT University of Copenhagen.
- Sheard, J., Carbone, A., Lister, R., Simon, B., Thompson, E., & Whalley, J. L. (2008). Going SOLO to assess novice programmers. *ACM SIGCSE Bulletin*, 40(3), 209–213. <https://doi.org/10.1145/1597849.1384328>
- Soloway, E. (1993). Should we teach students to program? *Communications of the ACM*, 36(10), 21–24. <https://doi.org/10.1145/163430.164061>
- Structural Learning. (2025). *A teacher's guide to SOLO taxonomy: Five levels for understanding better*. <https://www.structural-learning.com/post/what-is-solo-taxonomy>
- Wakeling, D. (2007). Spreadsheet functional programming. *Journal of Functional Programming*, 17(1), 131–143. <https://doi.org/10.1017/s0956796806006186>
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35. <https://doi.org/10.1145/1118178.1118215>
- Wing, J. M. (2017). Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology*, 25(2). <https://doi.org/10.17471/2499-4324/922>