How do Bebras Tasks Explore Algorithmic Thinking Skill in a Computational Thinking Contest?

Ana Liz Souto Oliveira
Federal University of Paraíba
Department of Exact Science
Paraíba, Brazil
Email: analiz@dcx.ufpb.br

Wilkerson L. Andrade
Federal University of Campina Grande
Software Practices Laboratory
Paraíba, Brazil
Email: wilkerson@computacao.ufcg.edu.br

Dalton D. Serey Guerrero
Federal University of Campina Grande
Software Practices Laboratory
Paraíba, Brazil
dalton@computacao.ufcg.edu.br

Monilly Ramos Araujo Melo
Federal University of Campina Grande
Laboratório de Neuropsicologia e Inovação Tecnológica
Paraíba, Brazil
Email: monillyramos@gmail.com

***Abstract*—This Research-to-Practice Full Paper examines the Bebras Challenge tasks, a Computational Thinking (CT) contest that does not require programming knowledge. Numerous studies in the last decade shed light on disseminating CT practices. However, there is still a gap in understanding CT as a construct, that is, CT as a complex cognitive dimension. Furthermore, few studies analyze how many constructs can be recognized inside CT by empirical data. Moreover, algorithmic thinking is one of the essential skills in CT, even when we do not apply programming activities. In this exploratory paper, we investigate the Bebras Challenge tasks. Our primary aim is to figure out how many and which are the constructs associated with CT skills derived from data, especially related to algorithmic thinking. The results point out three primary constructs supported by data. We claim that these constructs are a mix of CT skills, mainly algorithmic thinking, abstraction, generalization, evaluation, and pattern recognition. In addition, our study provides additional evidence supporting the idea that algorithmic thinking can be explored by different perspectives beyond "a set of order instructions", even without programming practices. Thus, this result reinforces the notion that CT can be understood as a mix of several skills to solve problems. Moreover, algorithmic thinking is a complex cognitive process of thought which may be explored through different strategies. We believe that these findings may help recognize which constructs and abilities underline CT, primarily algorithmic thinking without programming activities.**

I. Introduction

In 2006, Computational Thinking (CT) was proposed as a problem-solving process that explores fundamental concepts of Computer Science (CS) [1]. Over time, the researchers realized that CT is not a straightforward definition because it encompasses many different constructs beyond the CS concepts. Therefore, CT has been recognized as a complex and high-order thinking skill involved in problem-solving processes [2], [3].

Among several thinking skills, abstraction was claimed as the essence of computational thinking, but nowadays, mixed abilities have been pointed out as the core of CT [4], [5]. Abstraction is the capability to choose which aspects of a problem we should consider and which other we can overlook. Beyond abstraction, the Computing At School Institute has identified algorithmic thinking, decomposition, evaluation, and generalization as the bases of CT skills [6]. Algorithmic thinking consists of ordering steps to reach a goal or solve a problem. Meanwhile, decomposition is the act of separating the problem into more manageable parts. Finally, evaluation is the capability to find the best solution considering the resource, while generalization identifies similarities and addressess new problems based on previously known solutions.

Particularly, algorithmic thinking is fundamental in CT, both in programming and non-programming activities. During programming practice, the students learn how to write an algorithm through a textual programming language, such as Python, or a block-based visual programming language, such as Scratch [7]. On the other hand, algorithmic thinking can be stimulated independently from learning programming. More specifically, it may be explored by guided activities or problems/questions. For instance, Computer Science unplugged activities are a well-known approach to promote CT without programming as well as Bebras Challenge, an international CT contest [8]–[10].

Bebras Challenge[1] is a worldwide initiative to promote CT without programming practice as mandatory. The community started in 2004 in Lithuania, and, nowadays, Bebras is presented in more than 54 countries. Since then, Bebras' organizers have designed questions in which students should

[1] www.bebras.org

solve by exploring CT skills [11]. These questions are called tasks [12]. They also have claimed that one or more of five skills are needed to address the tasks. These skills encompass abstraction, algorithmic thinking, evaluation, decomposition, and generalization [13].

Although all those benchmarks, there is a dearth of evidence on whether CT skills can only be theoretical or empirically observed. Moreover, few studies analyze how many constructs can be recognized inside CT by data support [14], [15]. In fact, some studies indicate a positive, moderate, and statistically significant correlation between Bebras tasks and CT skills [16], [17]. However, only correlation results are insufficient evidence to be deep in CT abilities as a complex cognitive construct. It is important to encourage different statistical techniques to advance in skills allegations regarding theory and observed data.

The purpose of this exploratory study is to examine Bebras Challenge tasks intended to figure out how many and which are the constructs associated with CT skills, mostly related to algorithmic thinking in a non-programming context. Two research questions (RQ) have guided this investigation: (RQ1) How many constructs can be empirically detected in Bebras Challenge tasks and can be related to CT skills? This RQ1 concerns whether we can observe the five skills claimed by the Bebras organizers or different numbers, as claimed by the literature review. In this direction, we started from the data we obtained from students' answers and analyzed it according to statistical techniques, which pointed out how many constructs we can observe. Our goal is to check if the skills and constructs are the same as claimed by theory. (RQ2) How can algorithmic thinking be exploited in Bebras Challenge tasks? RQ2 investigates whether the data point only to one way to explore algorithmic thinking or whether the data suggest another way to exploit Algorithmic Thinking, when we look at the empirical data and the approach to solve tasks. We believe that these results may help understand which competencies and abilities underline CT, primarily algorithmic thinking without programming activities.

The outline of this study is as follows. Initially, we briefly explain the background in Section II. Then, we present the related work in Section III. Next, we detail the material and method in Section IV. After that, we show and discuss the results in Sections V and VI, respectively. Lastly, we address the concluding remarks in Section VII.

II. Background

In this Section, we briefly present the CT skills as well as an overview of Bebras Challenge. Finally, we sum up the exploratory procedure used in this research, called Principal Component Analysis.

A. Computational Thinking Skills

In the earlier definition, CT is a way humans solve problems using CS knowledge, but more proposals emerged over the years [1]. Hu claims that CT is a hybrid paradigm that encompasses not only algorithmic, logical, and analytic thinking but also mathematical, engineering, and creative thinking [18]. In another sight, Kalelioglu et al. recognize that CT is a problem-solving process that involves understanding the problem, plan, implement, and assess the solution [2].

Meanwhile, the Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE) established a CT operational definition associated with capabilities and skills [19]. CT is a problem-solving process that encompasses the following abilities: design and redesign problem that both computers and humans can solve it; analyzing and representing data using abstraction, for instance, models and simulations; exploring algorithmic thinking for both implementing and automating solutions as well as optimization resources; generalizing solution procedures to a similar problems domain.

Another educational association has identified a particular group of CT skills. The Computing At School Institute has proposed a core of five CT abilities [6]. They have included decomposition, evaluation, beyond generalization, abstraction, and algorithmic thinking. Decomposition encompasses the capability of seeing a complex problem and dividing it into parts. These parts are more manageable and easier to be understood and solved. Evaluation is a way of checking if the solution "fits for purpose". In other words, we should assess the properties of solutions to ensure that it is the best we can reach, considering the resources. Generalization is a strategy in which we recognize a previous solution and use it to solve a new similar problem.

Algorithmic thinking and abstraction are well-known procedures in CS, also highlighted in CT research. Abstraction consists of recognizing which details of a problem are essential and which can be suppressed. This new way of seeing the problem not only helps us to understand it but also reduce the complexity of its representation. Algorithmic thinking is an approach to produce a solution through a set of ordered steps. In this context, algorithmic thinking can also be seen as a merge of competencies related to design and understand algorithms [20]. These competencies encompass *(i)* the capability to design basic steps to solve a given problem, *(ii)* the capability to enumerate the elementary actions of a problem accurately, and *(iii)* the capability of creating and assessing suitable steps to solve a problem.

B. Bebras Tasks

Bebras is an international education community responsible for organizing a worldwide annual contest aimed at boosting students' interest in CT and CS [11]. Nowadays, around 50 countries have participated in the competition [13]. One part of the Bebras activities consists of designing questions and refining them in an annual workshop. These questions are called *tasks* and are developed in the English language originally. For example, Figure 1 shows an example of a task. "Social Network" task asks how many intersections are needed for linking A to G.

Although the task has no requirements for the content, it is usually designed around a specific CS concept [12]. However,

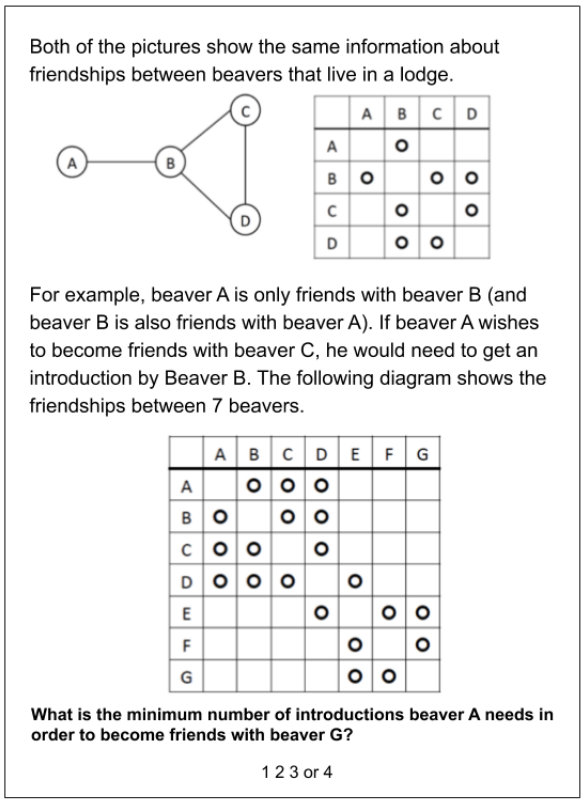

Fig. 1. Social Network question [21]

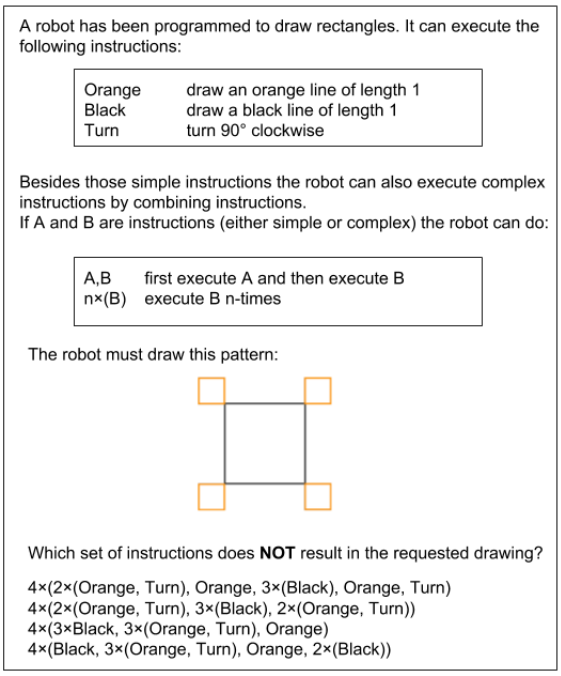

Fig. 2. Right Rectangles question [21]

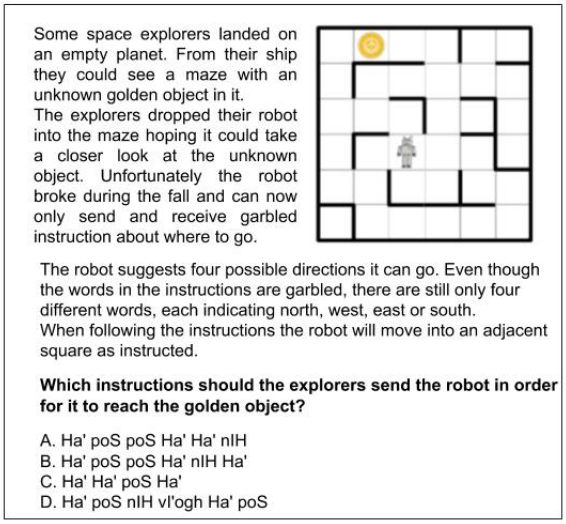

Fig. 3. Space maze question [21]

no previous knowledge in CS or programming is requested to answer them. For instance, "Right Rectangles" tasks (Figure 2) gives a set of instruction to a robot draws color rectangles. As we can notice, another characteristic is that the tasks are self-explanatory and self-contained [11]. In other words, tasks are easy to understand because they should describe what you need to know to answer them.

Beyond CS concepts, tasks are also designed to explore CT skills to solve them. Bebras' organizers have asserted that all tasks address one or more abilities, such as algorithmic thinking, abstraction, evaluation, decomposition, and generalization [13]. These skills are in accordance with those definitions adopted by Computing At School. For example, "Space maze" task (Figure 3) explores algorithmic thinking skill. Although there is no strict limit concerning how many skills one task can exploit, the organizers have suggested a maximum of three CT skills per task. Therefore, tasks encompass two-dimensional categorization, including CT abilities as well as CS content knowledge.

C. Principal Component Analysis

In this study, we are investigating CT skill as cognitive construct in Bebras Challenge. In this direction, we adopted a methodology called Principal Component Analysis.

Principal Component Analysis (PCA) can be understood as an approach of carrying out an exploratory factor analysis [22]. Factor analysis is a multivariate statistical technique that is suitable for examining the patterns of multidimensional relationships. It allows us to present the structure and interrelations of a group of variables. Factor is an essential concept in PCA because it represents the underlying dimensions of a set of observed variables, i.e., construct. One construct (factor) is related to variables by factor loadings, which is the correlation between variable and factor.

We summarize the PCA procedures in four-stage based on Hair et al. [22], as follows: *A - Checking the conceptual assumptions*; *B - Deriving a factor model*; *C - Rotating factors and choosing the factor model*; *D - Naming factors*.

A - Checking the conceptual assumptions: Initially, we should review some conceptual assumptions, such as sample size and two main statistical values. The minimum sample size is five subjects per question/variable [22]. Then, we calculate the Measure of Sampling Adequacy (MSA) and the Bartlett test of sphericity. The MSA shows the appropriateness of carrying out factor analysis (should be above 0.50). In contrast, the Bartlett test of sphericity presents the presence of correlations among the variables and their significance (the significance should be < 0.05).

B - Deriving a factor model: Then, we carry out the factor extraction to reach a factor model. A factor model shows how well the variables are clustered into the factors, often represented by a factor matrix. Considering the threshold value of the factor loadings, we can both include or exclude a variable from the factor model and decide the number of factors. Usually, the threshold value depends on the sample size, so loadings greater than 0.40 should be adopted for 100 participants [22].

C - Rotating factors and choosing the factor model: After identifying the significant loadings in the unrotated factor matrix, we should try to rotate factors in order to improve the interpretation of factors. The benefits of trying to rotate factors are, as Hair et al. said, "... to redistribute the variance from earlier factors to later ones to achieve a simpler, theoretically more meaningful factor pattern" [22] (page 111). Even with rotation, the total amount of variance extracted is the same, considering the rotated and unrotated solution. Thus, we should select the better factor modal, with or without rotated, based on the higher factor loadings.

D - Naming factors: When an acceptable factor model has been derived, we should nominate the factors. The process of naming the factors is theoretical and includes subjective interpretation of the factors in order to give a singular meaning to each one. The action of naming factors is based on the personal view of the researcher primarily. It is possible that another researcher may find a different name to the same factors considering her/his background, training in the area, and literature review.

III. Related work

Bebras tasks have been extensively investigated as an approach to promote CT. Beyond that, researchers have analyzed empirical data in order to confront the theoretical assumptions. Palts et al. have used data from the official Bebras contests to confirm the theoretical claim that five different CT skills encompass Bebras tasks [14]. Unexpectedly, the study did not confirm the allegation of five distinct abilities from empirical data, but they have pointed out two main recognizable factors (constructs). These factors were called algorithmic thinking (including algorithmic patterns and decomposition) and pattern recognition (including abstraction, generalization, and evaluation) [14].

The educational community has reported the Bebras tasks as a natural instrument for boosting and assessing CT abilities. For instance, Lockwood and Mooney have designed a test based on Bebras tasks to measure the CT skills progress [23], [24]. The objective was to assess CT skills in secondary school students and novice undergraduate students as well as understanding their view on CS and CT. Delal and Oner have developed unplugged computing activities based on Bebras Challenge to stimulated CT abilities in grade 6 students [10]. Besides, they have created a pre-test and a pos-test, composed of Bebras tasks.

Bebras tasks have also been used along with other instruments to assist in measuring CT skills, especially algorithmic thinking ability. Some studies have validated tasks with CT test (CTt) [3], which measures fundamental concepts of programming without code, such as sequences, loops, iteration, conditionals, and functions. For example, Wiebe et al. have investigated the psychometric relationship between Bebras tasks and CTt in order to indicate a more lean compact CT test [25]. As a result, they have proposed a pre-assessment instrument that does not require previous programming experience and focuses on algorithmic skills. In the same direction, Roman et al. have explored the convergent validity between Bebras tasks and CTt [16]. They found a positive, moderate, and statistically significant correlation between Bebras tasks and CTt.

Especially about algorithmic thinking, Futschek has argued that this skill is a core ability in Computer Science, which can be promoted apart from learning programming [20]. He has focused on problem definition and visualization of algorithms. He has shown how to solve paths in mazes and parallel sorting problems by creating pseudo-code algorithms.

IV. Materials and Methods

In this Section, we present the participants, the instrument, and the procedures of the study.

The participants were recruited from an introductory programming course at two universities in 2017 and 2018. All 233 participants were aged between 17 and 21 at the moment of data collection. The undergraduates were novices in Computer Science. They also were able to write basic programs in Python.

The instrument used in this study was the brochure of the national Bebras Challenge from the United Kingdom (UK) in 2014 [21]. This brochure contains questions and solutions of all age groups organized by the UK Bebras community. Therefore, we have selected fifteen questions designed for the older group (over 16 years old) in accordance with the age of the participants in our study. These questions were multiple choice or short answers questions. Table I shows the name of these tasks. Lastly, the questions were organized in a printed version.

Then, the students carried out the instrument with paper and pencil. We have scored one for each correct answer and zero for an incorrect answer. We excluded students with no answers in one or more questions from this analysis. In the end, we obtained 207 subjects who answered all fifteen questions.

After that, we conducted PCA intended to figure out a model of CT constructs. PCA is an exploratory statistical technique which points out factors or latent traits [22]. These factors

TABLE I
TASKS OF THE INSTRUMENT

Items	Tasks	Items	Tasks
i01	Ceremony	i09	Social network
i02	Log-art	i10	Height game
i03	Beavers on the run	i11	Meeting point
i04	Traffic in the city	i12	Best translation
i05	Storm proof network	i13	Broken machines
i06	Space maze	i14	True or false
i07	Footprints	i15	Right rectangles
i08	Puddle jumping		

TABLE II
PCA FACTOR MATRIX

Tasks	Factor 1	Factor 2	Factor 3
Social network	**0.67**	0.23	0.15
Footprints	**0.82**	-0.15	0.08
Right rectangles	0.12	**0.78**	-0.15
True or false	0.08	**0.73**	0.09
Broken machines	-0.11	**0.45**	0.27
Space maze	0.17	0.05	**0.79**
Beavers on the run	-0.19	0.18	**0.77**
Log-art	0.19	-0.06	**0.51**

become visible through the data. Moreover, it indicates which questions share common aspects by the coefficients (factor loadings). Thus, PCA's primary goal is to produce the factor matrix, i.e., the relations between tasks and factors. First of all, we have checked the conceptual assumptions. MSA and the Bartlett test of sphericity presented acceptable values above 0.70 and the significance < 0.05, respectively. Then, we have begun the PCA procedures with fifteen tasks. However, during the deriving of a factor model phase, we removed some tasks from the study due not to present good statistical values for PCA (for example, low values of factor loading or cross-loadings). This action of remove items with insufficient statistical values is indicated to reach an exploratory model. For that, we have adopted the factor loadings greater than 0.40, due to our sample's number, as suggested by Hair et al. [22]. Because of these reasons, seven tasks were dropped out of the analysis. At last, we have rotated factors and adopted the *oblimin* rotation for the reason that it demonstrated higher factor loadings than the unrotated matrix. Finally, taking into account the remaining items, we have identified the exploratory model of CT based on PCA.

V. RESULTS

In this Section, we show the statistical results.

After we carried out PCA, the factor matrix has indicated three main factors, as shown in Table II. We have highlighted the factor loadings related to each factor. As can be seen from Table II, Factor 1 has two questions with significant loadings, such as "Social network" and "Footprints", while Factor 2 has three questions with high factor loadings, "Right rectangles", "True or false", and "Broken machines". Lastly, Factor 3 has three questions, "Space maze", "Beavers on the run", and "Log-art".

Three main factors were showed up from the data. As suggested by Hair et al., the last step of PCA is to hypothesize the latent trait associated with each factor [22]. The researchers can execute this final step following what the literature claims. We agree with the Bebras' organizers that these latent traits are CT skills mainly. We have then scrutinized both the skills that can be used to address the set of tasks related to each factor and the brochure proposed by Bebras' organizers. After that, we may say that factor 1 is algorithmic thinking, abstraction, and generalization. Factor 2 is algorithmic thinking, abstraction, and evaluation. At last, Factor 3 is algorithmic thinking, abstraction, and pattern recognition.

VI. DISCUSSION

This Section explains the results as well as it addresses the research questions.

A. Computational Thinking Skills

Regarding our first research question (RQ1), "How many constructs can be empirically detected in Bebras Challenge tasks, and can be related to CT skills?", at first glance, we may expect that a single factor encompasses all tasks. This single factor would be Computational Thinking itself, i.e., we would think that CT is a single latent trait. However, we did not find this result. In fact, the statistical result has presented three main factors. So, we have moved on associating each factor to a single CT skill. Nevertheless, we have noticed that the one-to-one relationship (i.e., one factor to one skill) was not possible because more than one ability was used to address the solution. In other words, mixed skills are needed to answer those tasks.

The finding of our study has pointed out that three primary constructs were empirically detected from data. As the last step of PCA methodology suggests, we have named those factors based on the skills claimed by *(i)* Bebras' organizers, *(ii)* how to solve the tasks, and *(iii)* state of the art. All factors have presented algorithmic thinking and abstraction as common skills. Besides that, we have observed that generalization is explored in factor 1, evaluation in factor 2, and pattern recognition in factor 3.

Although factors can share some equal skills, the different meanings of the skill itself, along with the combination of them with a third skill, may explain the three prominent constructs. Besides, tasks can be solved by exploring different capabilities and skills, divergent from those initially designed. Because of that, it is not trivial to claim that a single ability is responsible for each factor. Therefore, the very nature of how Bebras tasks are created suggests that a mix of skills can be explored in each question.

Despite the fact that abstraction and algorithmic thinking skills have been pointed out in all factors, they may be explored in a way that is not the same as another. Thus, we have agreed with the Wing's claim that abstraction is the essence of CT [4]. In fact, abstraction is required to understand the problem and highlight which details are fundamental to reach the solution. Abstraction can also be used to analyze how the solution can be represented. For instance, abstraction is present when we have to obtain the solution by inference

from specific cases (factor 1); or evaluating solutions in order to choose the correct answer (factor 2), or identifying patterns inside the problems to reach the solution (factor 3). Wherefore, in the context of our study, we may say that abstraction is mixed with other CT skills assisting them in solving the problem.

B. Algorithmic Thinking Skill

We have believed that algorithmic thinking is a center ability in CT, regardless of programming. Looking at our findings, we have noticed that all questions encompass algorithmic thinking, but lightly divergent. These results may reveal different aspects of algorithmic thinking skills inside CT. Particularly, if we sift through how algorithmic thinking skills are explored in each task, we can notice some variation related to how the algorithm is required.

In "Social network" question (Figure 1), the graph helps to understand the relation of friendship in the adjacency matrix. Even if the student has no familiarity with matrix, the graph is a strategy to introduce the notation of relationship and connection. Thus, we believe that if the solver can see a visual representation of the connection rules, they may have a better description of how to solve the problem. In this case, the solver should test the shorter connection to link A to G (two introductions at least). One approach is to exploit the matrix intended to reach the relationship with the minimum number of links. Another solution is to rebuild the social network as a graph in order to see the visual representation. In both cases, firstly, the student should perceive how the relation of A, B, and C nodes is explained. Then, he/she moves to a more complex case (i.e., link A to G). In other words, he/she should use a previous solution to find the problem answer.

Different from the above-mentioned question, "Right rectangles" (Figure 2) presents algorithmic commands explicitly. In other words, procedures are represented using simple formal syntax. The visual representation, i.e., the draw with rectangles, is the goal of executing the instructions in a certain order. The student must understand the instructions and obey the order of commands. There is no other way to solve the question. Besides, some individual commands are repeatedly executed as well as the notion of angle is needed too. All alternatives generated the requested drawing from a distinct starting point, except the answer (the second alternative). Nevertheless, the student should execute the algorithm in the alternatives in order to answer the question (the wrong algorithm).

In the last question, "Space maze" (Figure 3), there are two possible paths in which the robot reaches the golden object by the maze, but only one is the correct alternative. The student should associate the commands in each option with the words indicating north, west, east, or south until finding the one that gets the golden object. Beyond the instructions, the student should check which combinations of commands answer the question (alternate A).

Finally, we can answer our RQ2: "How can algorithmic thinking skill be exploited in Bebras Challenge tasks?". Algorithmic thinking is exploited by a graph/matrix, by explicit commands, and by enciphering codes. Each of these kinds of tasks requires that the student have to reason differently. To summarize, in a graph or matrix, the rule of relations is indirectly represented by the image, while explicit commands demand execution in the correct order as they are declared. In the last case, encipher codes may represent an additional difficulty beyond running the code, but it is a sequence of steps. Therefore, the initial results have shown the possibility of distinguishing ways to exploit Algorithmic Thinking.

C. Threats to Validity

The three main limitations of this work are related to the adopted instrument, the subjects, and the sample size. Firstly, the initial proposal of Bebras Challenge is to increase the student's interest in CT and CS, not assess skills. Despite that, many studies have been used Bebras as a CT measure [8]–[10], [26], while others have seen it as a forthcoming international test in the Computer Science area [3], [13]. Secondly, although the Bebras target is pupils, the tasks can be applied to the general public. Indeed, studies have been used Bebras to measure CT in undergraduate students [23], [24], [27]. Lastly, regardless of the reduced sample size, we have respected the acceptable limits of 8:1 ratio of observations (students' answers) to variables (questions). Our study presented more than 13 subjects per question. In addition, the sample size greater than 200 students provides an adequate number for the calculation of the correlations between variables [22]. Despite the minor limits of this study, we believe that our findings may contribute to understanding CT, especially algorithmic thinking in a non-programming context.

VII. Concluding Remarks

This exploratory study investigated the underlying constructs of CT derived from a set of Bebras Challenge tasks, a worldwide contest of CT. Our data have suggested that it is not possible to identify the CT skills singly. In fact, three primary constructs have emerged from data, but they are a blend of CT skills, not a single one. We have recognized these constructs might be the following: *(i)* a mix of algorithmic thinking, abstraction, and generalization; *(ii)* a mix of algorithmic thinking, abstraction, and evaluation; and *(iii)* a mix of algorithmic thinking, abstraction, and pattern recognition.

Our study provided some additional evidence supporting the idea that algorithmic thinking can be explored by different perspectives, even without programming practices. Thus, the current findings suggest that algorithmic thinking skill has been exploited by a graph/matrix, or by explicit commands, or by enciphering codes. Each of these problems has required that the student think differently beyond "a set of order instructions". Thus, this result reinforces the notion that algorithmic thinking is a complex cognitive process of thinking. Therefore, we believe that algorithmic thinking is a crucial CT skill among all in which can be exploited independently of learning programming.

Finally, considering CT as one of the most important thinking skills for students' success in the 21st century, it is still little understood as a cognitive ability. In daily life, the problems are complex. Then, it is necessary to combine several cognitive systems for sense-making and problem-solving. In the future, when CT will be a well establish construct, we hope to be possible to better understand the cognitive and non-cognitive underlying factors of CT. For now, CT remains a poorly defined construct as well as related skills.

REFERENCES

[1] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006.

[2] F. Kalelioglu, Y. Gülbahar, and V. Kukul, "A framework for computational thinking based on a systematic research review," *Baltic Journal of Modern Computing*, vol. 4, no. 3, p. 583, 2016.

[3] M. Román-González, J.-C. Pérez-González, and C. Jiménez-Fernández, "Which cognitive abilities underlie computational thinking? criterion validity of the computational thinking test," *Computers in Human Behavior*, vol. 72, pp. 678–691, 2017.

[4] J. M. Wing, "Computational thinking and thinking about computing," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 366, no. 1881, pp. 3717–3725, 2008.

[5] V. Barr and C. Stephenson, "Bringing computational thinking to k-12: what is involved and what is the role of the computer science education community?" *Acm Inroads*, vol. 2, no. 1, pp. 48–54, 2011.

[6] A. Csizmadia, P. Curzon, M. Dorling, S. Humphreys, T. Ng, C. Selby, and J. Woollard, "Computational thinking: A guide for teachers," *Google Scholar*, 2015. [Online]. Available: http://community.computingatschool.org.uk/files/8550/original.pdf

[7] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," in *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*, 2012.

[8] W.-C. Kuo and T.-C. Hsu, "Learning computational thinking without a computer: How computational participation happens in a computational thinking board game," *The Asia-Pacific Education Researcher*, vol. 29, no. 1, pp. 67–83, 2020.

[9] J. del Olmo-Muñoz, R. Cózar-Gutiérrez, and J. A. González-Calero, "Computational thinking through unplugged activities in early years of primary education," *Computers & Education*, vol. 150, p. 103832, 2020.

[10] H. Delal and D. Oner, "Developing middle school students' computational thinking skills using unplugged computing activities." *Informatics in Education*, vol. 19, no. 1, pp. 1–13, 2020.

[11] V. Dagienė and G. Futschek, "Bebras international contest on informatics and computer literacy: Criteria for good tasks," in *International Conference on Informatics in Secondary Schools-Evolution and Perspectives*. Springer, 2008, pp. 19–30.

[12] C. Datzko, "The genesis of a bebras task," in *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*. Springer, 2019, pp. 240–255.

[13] V. Dagienė, G. Stupurienė, and L. Vinikienė, "Implementation of dynamic tasks on informatics and computational thinking," *Baltic Journal of Modern Computing*, vol. 5, no. 3, p. 306, 2017.

[14] T. Palts, M. Pedaste, V. Vene, and L. Vinikienė, "Tasks for assessing skills of computational thinking," in *ICERI2017 Proceedings*, ser. 10th annual International Conference of Education, Research and Innovation. IATED, 16-18 November, 2017 2017, pp. 2750–2759. [Online]. Available: http://dx.doi.org/10.21125/iceri.2017.0784

[15] A. L. S. O. Araujo, W. L. Andrade, D. D. S. Guerrero, and M. R. A. Melo, "How many abilities can we measure in computational thinking?: A study on bebras challenge," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, 2019, pp. 545–551.

[16] M. Román-González, J. Moreno-León, and G. Robles, "Complementary tools for computational thinking assessment," in *Proceedings of International Conference on Computational Thinking Education (CTE 2017), S. C Kong, J Sheldon, and K. Y Li (Eds.). The Education University of Hong Kong*, 2017, pp. 154–159.

[17] M. Román-González, J.-C. Pérez-González, J. Moreno-León, and G. Robles, "Extending the nomological network of computational thinking with non-cognitive factors," *Computers in Human Behavior*, vol. 80, pp. 441–459, 2018.

[18] C. Hu, "Computational thinking: what it might mean and what we might do about it," in *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*. ACM, 2011, pp. 223–227.

[19] I. S. for Technology in Education (ISTE) and C. S. T. A. (CSTA), "Computational thinking operational definition (on-line access)," 2014. [Online]. Available: https://id.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf?sfvrsn=2

[20] G. Futschek, "Algorithmic thinking: the key for understanding computer science," in *International conference on informatics in secondary schools-evolution and perspectives*. Springer, 2006, pp. 159–168.

[21] L. Howarth, P. Millican, C. Roffey, and S. Sentance, "UK Bebras Computational Thinking Challenge," 2014. [Online]. Available: http://www.bebras.uk/uploads/2/1/8/6/21861082/ukbebras2014-answers_1.pdf

[22] J. F. Hair, W. C. Black, B. J. Babin, and R. Anderson, *Multivariate data analysis*. Pearson, 2014, vol. 7.

[23] J. Lockwood and A. Mooney, "Developing a computational thinking test using bebras problems," *Proceedings of the CC-TEL 2018 and TACKLE 2018 Workshops, co-located with 13th European Conference on Technology Enhanced Learning (EC-TEL 2018)*, 2018.

[24] A. Mooney and J. Lockwood, "The analysis of a novel computational thinking test in first year undergraduate computer science course," *All Ireland Journal of Higher Education*, vol. 12, no. 1, 2020.

[25] E. Wiebe, J. London, O. Aksit, B. W. Mott, K. E. Boyer, and J. C. Lester, "Development of a lean computational thinking abilities assessment for middle grades students," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 2019, pp. 456–461.

[26] G. Chiazzese, M. Arrigo, A. Chifari, V. Lonati, and C. Tosto, "Exploring the effect of a robotics laboratory on computational thinking skills in primary school children using the bebras tasks," in *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality*. ACM, 2018, pp. 25–30.

[27] V. Dolgopolovas, T. Jevsikova, L. Savulionienė, and V. Dagienė, "On evaluation of computational thinking of software engineering novice students," in *Proceedings of the IFIP TC3 Working Conference "A New Culture of Learning: Computing and next Generations*, 2015, pp. 90–99.