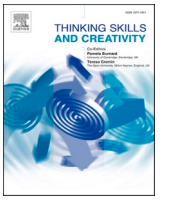
Thinking Skills and Creativity 48 (2023) 101284

Contents lists available at ScienceDirect

Thinking Skills and Creativity

journal homepage: www.elsevier.com/locate/tsc



Characterising algorithmic thinking: A university study of unplugged activities☆

Adrián Bacelo [a,*], Inés M. Gómez-Chacón [b]

[a] Didáctica de la Matemática, Facultad de Ciencias Matemáticas, Universidad Complutense de Madrid, 28040 Madrid, Spain
[b] Instituto de Matemática Interdisciplinar, Didáctica de la Matemática, Facultad de Ciencias Matemáticas, Universidad Complutense de Madrid, 28040 Madrid, Spain

ARTICLE INFO

Keywords:
Algorithmic thinking
Theory of mathematical working space
Computational thinking
University teaching
Statistical Implicative Analysis (SIA)

ABSTRACT

Algorithmic thinking is a type of thinking that occurs in the context of computational thinking. Given its importance in the current educational context, it seems pertinent to deepen into its conceptual and operational understanding for teaching. The exploration of research shows us that there are almost no studies at university level where algorithmic thinking is connected to mathematical thinking, and more importantly, to characterise it and be able to analyse and evaluate it better. The aim of this research is to characterise algorithmic thinking in a university context of the Bachelor's Degree in Mathematics by unplugged tasks, offering a model of analysis through categories that establish connections between mathematical and algorithmic working spaces in three dimensions, semiotic, instrumental and discursive. The results confirm the interaction between these dimensions and their predictive value for better programming performance. The study also adds novel considerations related to the role and interaction of mathematical and computational thinking categories involved in algorithmic thinking.

1. Introduction

In the last decades, there has been an increase of the implementation in school curriculum of computational thinking (CT) (Adell et al., 2019; Fraillon et al., 2020) and the development of proposals for activities and projects that improve students' problem-solving and analytical skills through some key concepts in CT (Moreno-León et al., 2015; Shin et al., 2021).

Although there is no consensus on the definition of CT, we can identify studies that to consolidate an operational definition (Shute et al., 2017; Stephens & Kadijevich, 2020). Aho (2012) defines computational thinking as 'the thought process involved in formulating problems so their solutions can be represented as computational steps and algorithms'. This definition shows the connection between the typology process, the algorithmic process and the sequencing required for its execution. CT is part of the set of essential skills that a student must master in order to solve problems in the digital age. This may include several key concepts such as abstraction, decomposition, pattern recognition and algorithms.

In regard to algorithms, it is of importance to clarify the concept of an algorithm as it is part of the CT. There are several debates about the concept (Hill, 2016), as a distinction can be made from computational, historical and philosophical perspective. In this article, as most papers that work with algorithms, because of their clarity, we consider the classical definition of algorithm given by

* Corresponding author at: Universidad Complutense de Madrid (Spain), Facultad de Ciencias Matemáticas, Plaza de Ciencias 3, 28040 Madrid, Spain.
E-mail address: abacelo@ucm.es (A. Bacelo).

https://doi.org/10.1016/j.tsc.2023.101284
Received 12 December 2022; Received in revised form 12 March 2023; Accepted 15 March 2023
Available online 16 March 2023

Knuth (1997): 'a finite set of rules that gives a sequence of operations for solving a specific type of problem'. He further states that there are five important characteristics: finiteness, precision, input, output, and effectiveness.

Furthermore, we consider algorithmic thinking (AT) as a specific type of thinking that occurs in the context of computational thinking: 'A logical, organised way of thinking used to break down a complicated goal into a series of (ordered) steps using available tools' (Lockwood et al., 2016). Knuth (1985) associates AT with computer science and identifies its contribution to the reasoning in various mathematical fields, distinguishing between algorithmic and mathematical thinking.

Several authors have studied the relationship between CT and AT, as well as the relationship between each of them and mathematical thinking (Stephens & Kadijevich, 2020). Sneider et al. (2014) investigate the link between mathematical and computational thinking, exploring specific and shared abilities. Lafuente Martínez et al. (2022) develop a test in adults to assess AT capacity by selecting different specific CT skills.

Couderette (2016) indicates that teaching AT is quite difficult due to the involvement of two disciplines: mathematics and computer science. We agree with this author, and with Stephens & Kadijevich, (2020), and support the idea that the study of AT should be seen from both disciplines because of the advantages it has in mathematical learning. In this research our focus is on the conceptual understanding of AT and the processes associated with it in the context of programming (Figueiredo et al., 2021; Stephens, 2018). Different studies at pre-university and university level put the accent on this direction. For example, at the pre-university level, the remarkable studies seek to highlight the connections between CT and logical-mathematical thinking through algorithmic processes (Montes-León et al., 2020). Other studies propose methodologies based on process aspects, such as the metaphor-based methodology, comparing programming with cooking (Pérez-Marín et al., 2020). At the university level, we find studies such as those by Cachero et al. (2020), which indicate the need for training in aspects related to AT, as it is not developed naturally. The synthesis studies by Tang et al., (2020) conclude that at university education compared to research at pre-university level requires more attention.

These results, together with our experience, observing real situations in the classroom or even experimental situations, show that the lack of connection between experience in CT/AT and mathematical formalism, techniques, etc., and that the connection between solving and reasoning processes in programming is worrying.

Motivated by this perspective, the research described focuses on the nature of AT in unplugged tasks by developing the implicit relationships between semiotic, instrumental and discursive dimensions as students solve tasks in which they must build an algorithm. The field studied is based on data from a group of university students, from the Bachelor's Degree in Mathematics, performing unplugged tasks.

2. Theoretical framework

This section presents the basic theoretical concepts supporting the study: the concept of AT and its relationship with Mathematics; and the theoretical framework of *Mathematical Working Spaces*, explaining the different dimensions that occur and their connection with the *Algorithmic Working Space*.

2.1. Algorithmic thinking

Algorithms have existed and have been studied since the beginning of Mathematics, nevertheless, its current development is due to the progress in computing. According to Peña Marí (2006), the appearance of programming languages has opened a new paradigm for algorithmic problem solving. These languages made possible the use of notions that were so far impossible to use on paper, such as the storage of intermediate results to link some calculations with others or the efficiency of the algorithm. They also added instructions that made it possible to program algorithms that were not feasible before: computational variables, Booleans, characters, assignments, etc. (Peña Marí, 2006), confirming consequently the correspondence between mathematical logic and programming (Ferreira-Szpiniak et al., 1997). In this study, AT is considered as a particular form of mathematical reasoning, using the previous definition: 'A logical, organised way of thinking used to break down a complicated goal into a series of (ordered) steps using available tools' (Lockwood et al, 2016) and also as a type of thinking that can be developed independently of learning programming (Futschek, 2006).

Knuth (1985) attempts to clarify the role of algorithm in mathematics by choosing and analysing a problem among all the problems that may appear on page 100 of famous books. For each of them, he carries out an analysis considering among others what kind of knowledge is used to solve it, the behaviour of the value functions, the possible generalisation, and the relation with infinity. It is interesting to notice the categories that characterise AT for the author: the manipulation of formulas, the representation of reality, the reduction to simpler problems, abstract reasoning, information structures and algorithms (Knuth, 1985). Knuth (1985) does not come to any conclusions, although he misses in his own analysis the presence of the notion of complexity (related to the 'cost' of performing the algorithm) and assignment operations (such as x: = x + 1), taking steps towards the constitutive elements to be taken into account in AT.

As we have seen with CT, the aim of the research on AT is to enhance it and to work on it in different ways. To do so, we must distinguish between two types of activities: plugged and unplugged (Erümit & Sahin, 2020). Plugged activities are those that make use of technological elements such as computers, tablets or mobiles. Unplugged activities are those that we work on using non-technological elements and that make use of analogue elements for their development.

Within the plugged activities, Cooper et al. (2000) have developed the Alice technology that allows us to improve AT using a favourable environment. Another example is given by Hromkovič et al. (2016), who propose three activities using computers arising from their own experience over the years in the classroom, and with which there is a great opportunity to work more deeply on the concepts associated with AT. Finally, GeoGebra is a very useful tool when working on AT, as shown by van Borkulo et al. (2021) in their

study done at pre-university levels.

Similarly, we found research on the impact of unplugged activities on this type of thinking. At pre-university levels, setting well-chosen algorithmic problems to visualise their resolution as a game or programme is effective in improving AT (Futschek, 2006). Similarly, Burton (2010) designs unplugged activities, focusing on different aspects of algorithms. Also at university level, Posso's (2022) research is of interest as he concludes the necessity to articulate unplugged activities with plugged activities since the impact of proposals of only plugged activities is not very significant for the development of AT in students. What is interesting is that, when performing unplugged and plugged activities at the same time, the group that performs unplugged activities shows significant improvement while the groups with plugged activities do not show this significant improvement (Kirçali & Özdener, 2022).

Consequently, given the literature and the aim of the study, we choose to carry out unplugged activities. Firstly, due to the rare appearance of this type of activities in university education (Kim, 2018). And secondly, because with this type of activities we can make a study of the arguments, steps and processes used by students (Battal et al., 2021), which will be very useful for the characterisation we want to carry out.

2.2. Algorithmic thinking and mathematics education

In the field of mathematics education, when we talk about CT, we want to highlight an idea that goes beyond the use of technological tools to solve a task. We are talking about students learning CT skills. These skills include recognising patterns, designing and using abstractions, decomposing patterns, determining which tools are appropriate for analysing or solving a problem, and defining algorithms as part of a solution. Stephens & Kadijevich (2020) update research references about CT and AT concepts. They explore the similarities and differences in computer science and mathematics education in relation to CT. They present some prevailing definitions of CT and connect them to the closely related construct of AT. By identifying uses in the curriculum from different countries, they highlight the rich interface between algorithms and mathematics, for example in the fields of proof and conjecture, where this mutual dynamism could be further cultivated in the mathematics curriculum.

Among common components to computer science and mathematics education that stand out in several definitions given to CT, it can be highlighted the aspects of decomposition, abstraction and algorithms (Shute et al. 2017). These common components are present in a model proposed in mathematics education by Hoyles and Noss (2015) who assumed that CT is based upon decomposition, abstraction, pattern recognition, and AT so as to enhance mathematics learning through revisiting programming. In addition, the work of Drijvers et al. (2019) explores the potential of the abstraction-modelling-problem-solving triad in the curriculum.

In relation to AT, Lagrange and Laval (2019 and 2023) emphasise the study of algorithms as mathematical objects, their formal language dimension and how this formal notation for algorithms or 'language' is a vehicle for abstraction rather than for execution on a computer. Their theoretical and empirical investigation in high school seeks to identify appropriate ways to link computer programming and algorithmics to mathematical learning. They explore the suitable algorithmic and mathematical working spaces in which students develop an understanding of the Intermediate Value Theorem, and the Bisection Algorithm.

The nature of algorithms allows for a constructive approach to mathematical concepts through language (Lagrange, 2020). At pre-university level, Lockwood and De Chenne (2020) find that through the use of conditionals, students' reasoning improves mathematical understanding of the underlying concept. Cohors-Fresenborg (1993) introduces the concept of algorithm making students solving computational problems using a recording machine. Gal-Ezer & Lichtenstein, 1996 show how AT and mathematical thinking complement each other, using examples from set theory. Besides, there are numerous studies on the assessment of AT at these educational levels (Kanaki & Kalogiannakis, 2022; Gubo & Végh, 2021; Georgiou & Angeli, 2021). However, there is a significant gap in the literature regarding AT studies at university stages. Among the few that we can find, we have for example Korkmaz et al. (2015) that analyse AT in university students founding that the older the students, the lower their algorithmic skills. Additionally, we can find models of AT applied to teaching discrete mathematics at university levels (Liu & Wang, 2010). This underlines the importance of working along these lines at university level.

2.3. Mathematical and algorithmic working spaces

Our approach to the algorithmic and computational will have as a basis for its understanding the theory of Mathematical Working Spaces. According to Kuzniak et al. (2016) a Mathematical Working Space (MWS) refers to an abstract structure that is organized to account mathematical work in an educational setting. The framework of the MWS is a didactic theory strongly supported by mathematical contents which allow characterising the way the concepts make sense in each work context when the individual solves a task. It achieves this by closely combining epistemological aspects - purely mathematical criteria- and cognitive aspects - related to the processes and ways of doing adopted by the individuals who solve a task, around the unifying notion of mathematical work. These two aspects are identified and described according to two planes: the epistemological plane and the cognitive plane. These planes are articulated through a genesis process, specifically three dimensions:

Semiotic: use of symbols, graphics and concrete objects understood as signs.

Instrumental: construction by means of artefacts (geometric figure, graphics, programme…).

Discursive: justification and proof using a theoretical frame of reference.

The MWS theory distinguishes three levels for understanding the complexity of mathematical work in a school context. The reference MWS is a characterisation of mathematics and mathematical work specified by an educational organisation. The suitable

MWS organises the mathematical work, in line with the reference MWS, which will be proposed to the pupils. The aim of the teachers who set up the suitable MWS is that their students carry out mathematical work corresponding to the reference work. It is at this level of the suitable MWS that mathematical learning takes place. In effect, students are confronted with a mathematical task with their own knowledge and cognitive processes, which are shaped by their personal MWS. Nonetheless, the work done may not conform to what is expected and defined by the educational organisation. As a consequence, the aim of implementing suitable MWS is to align the students' personal MWS more closely with the expectations of the educational organisation.

This theory has been adopted in different studies in which the articulation of different types of WSs workspaces has been considered (Montoya Delgadillo and Vivier, 2014; Gómez-Chacón & Kuzniak, 2015) and in particular for algorithmic content (Lagrange & Laval, 2019 and 2023).

Programming activities involve two *Working Spaces*: the *Mathematical* and the *Algorithmic*. Lagrange & Laval (2019 and 2023) use this *MWS* theory to account for how the linkages between the two spaces gives meaning to the concepts involved in design and analysis AWS. This study focuses on the suitable algorithmic and mathematical working spaces in which students develop an understanding of the intermediate value theorem, and the bisection algorithm with students aged 16–19. This extended *MWS* framework considers semiotic and instrumental dimensions, as well as content and mode of reasoning, in different domains of activity and their interaction in a mathematical activity.

In our work we use the MWS theory to analyse students' work in unplugged activities for two aspects: 1) to explore the conditions students need for the development of cognitive skills associated with AT through programming and 2) to identify how students can establish connections between the algorithmic dimension and mathematics in each of the three dimensions: semiotic, instrumental and discursive.

The theory of *MWS* has the potential to make us reconsider issues such as the above by differentiating between what is properly instrumental - here the implemented representations and algorithms and the users' awareness of these - and what belongs to other dimensions such as the semiotic and/or discursive that have a strongly mathematical underpinning. The instrumental dimension (in terms of *MWS*) is not isolated from the other dimensions but is articulated with other dimensions in the working space where the artefact is used (Gómez-Chacón & Kuzniak, 2015; Nechache & Gómez-Chacón, 2022).

3. Research questions and methodology

The aim of this research is to characterise AT in a university subject context (computer programming and mathematics) considering the three dimensions, semiotic, instrumental and discursive, which support the *Algorithmic Working Spaces*.

The research questions raised are the following:

RQ1. What aspects do characterise students' algorithmic working space in its semiotic, discursive and instrumental dimensions in unplugged tasks?

RQ2: Which variables that support the semiotic, discursive and instrumental dimensions do best predict success in a task? Is there a relationship with MWS?

The research is carried out in real classroom at university level, and in this way, it is closely linked to practice. The methodology is framed within Design research with a focus on learning processes and task design (Cobb et al. 2003; Prediger et al., 2015). A combination of qualitative and quantitative methods to approach the subject of the study is used.

Two types of analysis are carried out in this study. The first is exploratory, descriptive and interpretative, and involves mainly an inductive content analysis of the data, with categories and interpretation based on the dimensions of MWSs. This analysis is carried out by means of judge triangulation, a cross-checking of the solutions by three researchers. The first step of the data analysis is to classify what characterises the AT process used by the learners and to identify classes of responses according to the set of categories (see Section 3.1). To establish the categories, the dimensions proposed for AT analysis are considered, highlighting the elements of the epistemological plane introduced by the mathematical work and the inductive analysis of Task 1 Test A (see Section 2). This content analysis serves as the basis for the final classification of categories that underpins the study and which we specify in Section 3.3. The second type of analysis is based on the implicative statistical analysis for which the data matrix had to be compiled and coded according to these categories (see Section 3.2 and 3.3).

3.1. Study group and presentation of the tasks

The context is a course, called Computer Science (UCM), which is offered at the Complutense University of Madrid to students of the degree in mathematics, 60 students and the lecturer. It is a basic introductory course in programming. Python is used as the programming language. The specific competences that the student must acquire are:

- Writing simple programs and general procedures that solve simple classical programming problems depending or not on some parameters.
- Writing simple recursive programs and reasoning about their operation using induction.

In the following figure, we will present the tasks on which the study is based on, and we will made a didactical analysis from the point of view of the relationship between both AWS and MWS suitable workspaces. This analysis allows to carry out a post analysis of students' works to establish as conclusions recommendations for a suitable work.

The tasks we present (Fig. 1) are taken from two final exams of the first semester (Test A) and the second semester (Test B). The

items of these exams are representative of those proposed in the course (the inventory of problems or tasks of the course is more than 100) and contain in their solutions elements associated with AT and algorithms (Section 2.1), serving as a basis for developing the intended work.

We can observe that the two tasks of Test A deal with polynomials and the third task, corresponding to Test B, deal with character strings. The three tasks require a mathematical base and a fluent handling of the associated concepts, and likewise, the solution associated with the three tasks is a basic algorithm that contains the categories to be analysed. In their solutions, the concepts of AT can be appreciated: computational variables, loops, efficiency, assignments, etc.

> **Task 1**: Given any two polynomials in the same mathematical variable of finite length with integer coefficients, determine the sum of these two polynomials.

3.1.1. A priori analysis task 1: connecting MWS and AWS

To find elements that characterise AT, we are going to analyse the answers given by the students to Task 1. On that account, first of all, we are going to analyse the possible solutions to the proposed problem, both from a mathematical and a programming point of view.

The solution to the task is an algorithm that allows us to add polynomials once both polynomials have been introduced in the algorithm itself.

If we analyse the task from a mathematical perspective (thinking about a formal mathematical solution), the aim is to obtain an algorithm that allows us to add two given finite polynomials. The result will be a polynomial obtained by adding the coefficients of the monomials that have the same degree in both polynomials to be added and leaving unchanged the monomials of a certain degree that only appear in one of them.

If we approach the task from what we learnt in the first pre-university school levels, a candidate algorithm would be first completing the polynomials with all possible degrees (with reference to the largest one) and then adding the coefficients of both polynomials one by one (Fig. 2).

As we have indicated, this approach corresponds to basic levels of thinking and, above all, it is not efficient from a programming point of view. In fact, this approach to the problem had already been carried out by the students previously in a practical exercise, so what was actually required was an improvement of this algorithm. For that reason, a more efficient algorithm involves comparing the degrees and if they are in both, it is added to the final polynomial. If it is only in one, it is added to the final list directly (Fig. 3)

In this case, the semiotic dimension in the *Algorithmic Working Space* (*AWS*) is marked using iteration (while) and alternative (if) elements and the computational variables that we define, which vary as the algorithm progresses due to the assignment that is made, unlike what happens with the mathematical variables (MWS *Mathematical Working Space*). Then, AWS involves expressions and mathematical knowledge related to polynomials, such as the degree, coefficients or length. On the semiotic level, we could encounter problems in those students who do not have a strong foundation in algebra and in the handling of polynomials. In the instrumental dimension, as we are constructing an algorithm that must be executed, we need some device (software) to be able to do it, and this will greatly help the student to understand how it works and to give the abstraction that took place meaning. The discursive dimension is characterised by questions related to the efficiency and complexity of the algorithm itself: is it finite, is the solution going to be correct, are there shorter ways of doing it, do I use functions and assignment correctly?

Mathematically, given any two polynomials (with n distinct mathematical variables, coefficients in any numerical set and finite or

Number	Task Statement	Description
Task 1 (Test A)	Given any two polynomials in the same variable of finite length with integer coefficients, determine the sum of these two polynomials.	Mastery of the concept of polynomial and its properties. An algorithmic process is sought that results in the sum of the polynomials only taking into account the monomials that appear in both polynomials.
Task 2 (Test A)	Given any two polynomials in the same variable of finite length with integer coefficients, determine the product of these two polynomials.	Mastery of the concept of polynomial and its properties. The algorithmic procedure must be based on an auxiliary function that multiplies a polynomial by a monomial.
Task 3 (Test B)	Given a character string s and a list of characters of length 1 seps, write an iterative function that returns a list of character strings that are the substrings of s separated by some character of seps.	Mastery of strings and basic programming language. The algorithmic operation to solve problems involves dividing it into a smaller problem supported by an auxiliary function that tells you if an element appears in the indicated list.

Fig. 1. Task statements and brief description.

```
if size(p1) > size(p2)
        k=size(p1)
else
        k=size(p2)
end if
i==k
while i>-1
        if p1[i][0]=i
                p1_aux.append (p1[i][0], p1[i][1])
                i=i-1
        else
                p1_aux.append (i, 0)
                i=i-1
        end if
end while
j==k
while j>-1
        if p2[j][0]=j
                p2_aux.append (p2[j][0], p1[j][1])
                j=j-1
        else
                p2_aux.append (j, 0)
                j=j-1
        end if
end while
h==k
while h>-1
        p-append (h, p1_aux[h][1]+ p2_aux[h][1])
        h=h-1
end while
return p
```

Fig. 2. Programming task 1 in Python – Mode 1.

infinite length) they can always be added together. In our case, we have added restrictions: a single mathematical variable, integer coefficients and finite length. Therefore, the discursive dimension appears in a very strong way: properties of addition can play a role in the operations; the impossibility of operating elements of different degrees and the search for a mathematical solution rather than an algorithmic one. The mathematical variables as something fixed and not in continuous change as in the algorithm, the formalism of the sum of monomials and the concepts of degree and coefficient appear in the semiotic dimension. The instruments that appear here are paper and pencil to make calculations and the use of the calculator if the coefficients are complicated. It should be noted that the polynomials used in the computer algorithm are ordered from highest to lowest degree, while mathematically they do not have to be.

There are clear links between the two *Working Spaces (WSs)*, where if one *Working Space* is given effectively, it clearly benefits the other *Working Space*. However, it does influence how the other *Working Space* is viewed and how tasks in that spectrum are worked on.

For example, if the MWS is complete (a circularity between dimensions in the work), when we move to the AWS, we will obtain a clear, simple and effective algorithm by having a broad and clear knowledge of the task. On the other hand, if that WS has not been developed in one of the dimensions, we will see that gap in the AWS. For example, if the semiotic dimension has any gaps, in the algorithms we will see failures in its semiotic dimension, i.e., in the definition of computational variables and conditions for loops.

If we study from the other point of view, the *AWS*, when we move on to the *MWS* we will see that the student has been able to acquire a specific tool in order to be able to add two finite polynomials in one mathematical variable, with integer coefficients. The MWS opens more possibilities with respect to working with polynomials, but by applying what has been learned in the AWS, the student can become organised in a more visual and clearer way for him/her. Similarly, both spaces are connected, but here the passage

```
i=0
j=0
p=[]
while i<size(p1) and j<size(p2)
        if p1[i][0]==p2[j][0]
                p.append (p1[i][0], p1[i][1]+p2[j][1])
                j=j+1
                i=i+1
        elif p1[i][0] < p2[j][0]
                p.append (p2[j])
                j=j+1
        elif p1[i][0] > p2[j][0]
                p.append (p1[i])
                i=i+1
while i<size(p1):
        p.append(p1[i])
        i = i + 1
while j<size(p2):
        p.append(p2[j])
        j = j + 1
return p
```

Fig. 3. Programming task 1 in Python – Mode 2.

from one to the other does not cause as many problems as in the previous case. This is because we move from something symbolic and concrete to something formal, where we can see the results, exemplifying and generalising with a greater basis and work on the concepts in a clearer way, always bearing in mind the abstraction that occurs in the process.

3.2. Methods of data analysis

Students' responses to the problems were analysed and coded through qualitative data processing using content analysis to define the categories indicated in section 3.4.

Initially, solutions given to Task 1 of Test A were taken, focusing on the semiotic, discursive and instrumental dimensions, inductively obtaining categories describing these dimensions and identifying the most common difficulties. These solutions reveal deficiencies or poorly constructed and erroneous ways of working. In this selection of student' difficulties and errors, we take into account the basic elements to be considered when performing an algorithm (Section 2.1) and through the lens of the MWS, the students' development of suitable mathematical/algorithmic work is studied as a gradual and progressively process bridging the epistemological and cognitive planes and it could be associated in all three dimensions.

Fig. 4 displays an outline-summary with the dimensions and the most typical errors appearing in each category. Figs. 5 and 6 show

an extract of a protocol exemplifying the analysis.

One of the most frequently repeated erroneous ways of working is taking the number of monomials as a function of the degree of the polynomial, which indicates a lack of mathematical knowledge when adding two polynomials. In doing so, several of the proposed algorithms do not add any of the monomials by restricting it to the degree of one of the polynomials. This may be an indicator that for students the use of abstraction and generalisation to write the algorithm in a programming language constitutes a difficulty.

Another misstep to be highlighted is the fact of trying to add the two polynomials directly by means of the addition symbol in the algorithm. This is contrary to the programming syntax, as this symbol adds numbers, not other elements. It also seems to indicate that students need a better understanding on how we should communicate with a computer through programming languages. As well, linked to this idea, we have some examples of students comparing elements that have different variables, such as an integer with a vector.

3.3. Implicative data analysis

The students' responses to the tasks were analysed and coded through qualitative and quantitative data processing using the categories listed below.

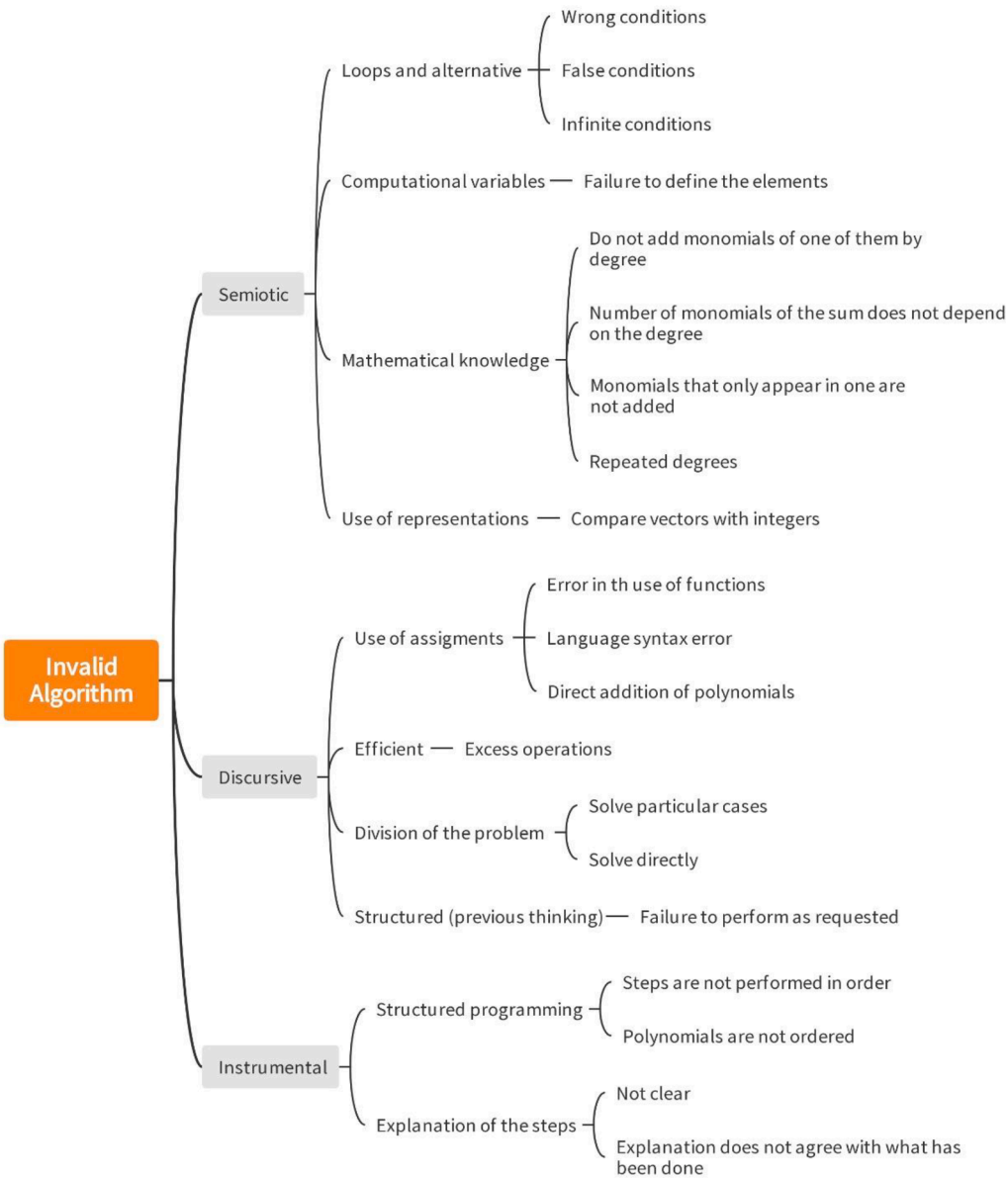


Fig. 4. Outline-summary of task 1, test A, dimensions and categories.

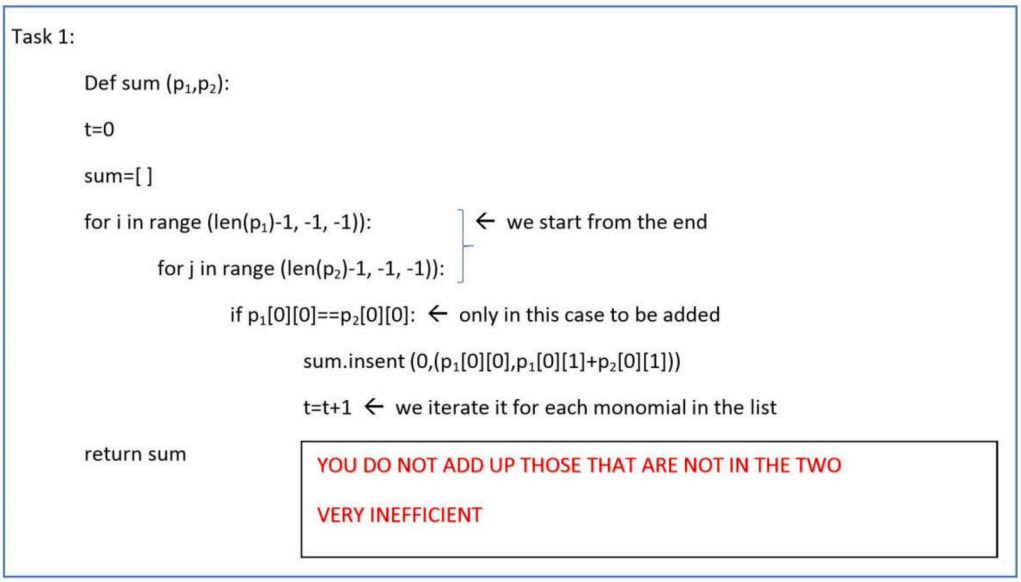


Fig. 5. Extract of protocol with categories as efficient and mathematical knowledge.

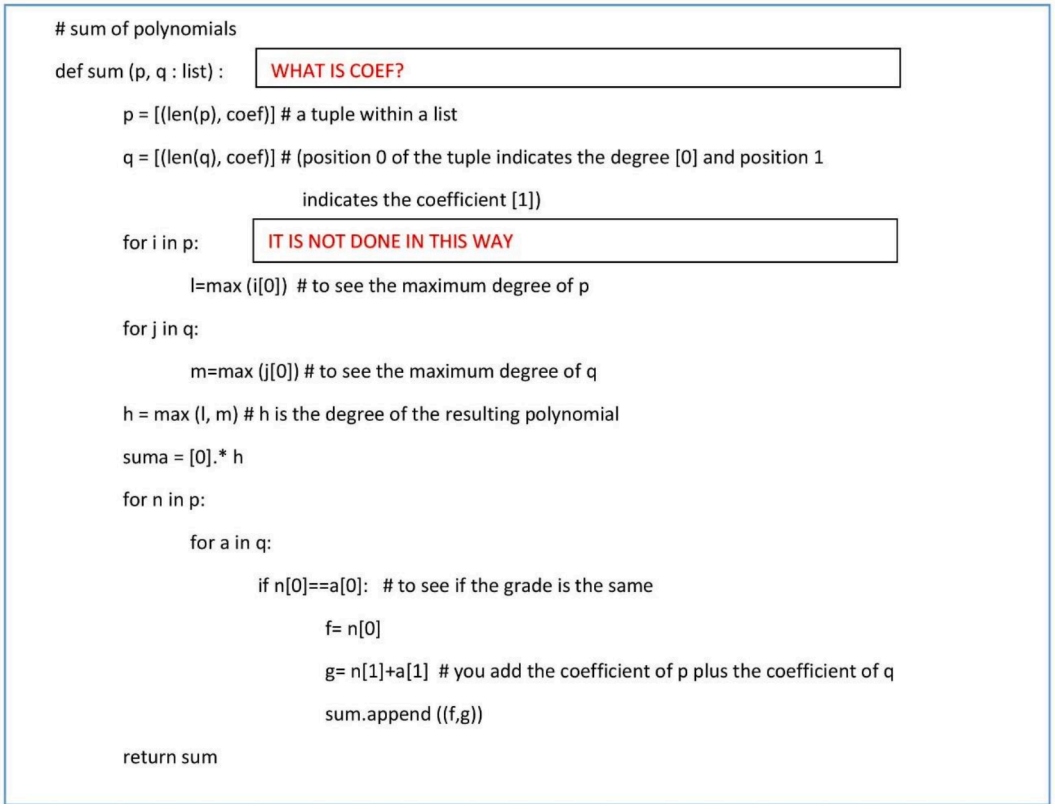


Fig. 6. Extract of protocol with categories as structured and use of assignments.

For each of the dimensions, we consider the definition and the most recurrent elements to be considered in the development of AT and, above all, the definition of algorithm, always associated with mathematics.

For the semiotic dimension we consider the following categories:

S1. Use of loops (while) and alternatively (if)
S2. Use of computational variables
S3. Mathematical knowledge
S4. Correct use of representations

With the first category, we consider the importance of repetitions and conditionals for the functioning of an algorithm. It is also basic, and it is reflected in the second category, to make use of auxiliary computational variables that change as the algorithm progresses. The third category is related to the needed basis for tackling the mathematical problem and the prior study of the basic notions associated with it. Finally, the fourth category represents, in the field of programming, the need to relate elements that are within the same mathematical category.

Within the discursive dimension we have:

D1. Correct use of assignments
D2. Efficient
D3. Division into smaller problems
D4. Structured (previous thinking)

Here we have basic elements that also appear directly in the characteristics explained by Knuth (1997). Thus, regarding the first category, we have in mind the importance of knowing which functions or syntax should be used so that the assignment of elements is done correctly. Concerning the second, we highlight the importance of the speed and number of instructions that the algorithm has, which is also related to the third category, which represents the need to solve the problem in simpler subdivisions that allow us to get to the initial problem. Finally, for a correct solution, it is necessary to be clear about how we want to tackle the problem and from where, in order to then overturn it in the algorithm, which is represented in the fourth category.

Finally, if we look at the instrumental dimension we have:

I1. Structured programming (language)
I2. Explanation of the steps

Here it is clear that everything directly 'connected' comes into play. As a first category we have the use of the computer language that we use when programming the algorithm. As a second category the explanation of each of the actions that are being done, in order to find the coherence between what is thought and what is programmed.

REND will be denoted by achievement, considered as the grade attributed to a correct resolution.

Furthermore, all categories were compiled and coded in a matrix for the implicative analysis carried out with the CHIC software (Couturier et al., 2000).

The procedure for the implicative data analysis method (Gras et al., 2008) begins with a group of individuals (the 60 subjects) described by a finite set of binary variables (S1. Use of loops (while) and alternatively (if), D1. Correct use of assignments, etc..). The question proposed is: do subjects known to be characterised by a tend also to exhibit b?

According to Gras et al. (2008), learning begins with inter-related facts and rules that progressively form learning structures. That is precisely the aim of the present study, finding relations between categories (listed previously) while furnishing information on the factors involved in the structure of AT. Gras defines rules that can be described in learning processes. These rules describe a hierarchical, orientated and non-symmetrical learning structure, that can be obtained with cohesive hierarchical implicative classification (CHIC) software (Couturier et al., 2000). The result is three types of diagrams that contain different types of information. a) Similarity trees group variables on the grounds of their uniformity, allowing for interpretation of the groupings with which the variables are handled. Each level on the resulting graph contains groups arranged in descending order of similarity. b) Hierarchy trees are used to interpret classes of variables defined in terms of significant levels along the lines of similarity, identifying association rules and levels of cohesion among variables or classes. c) Implication graphs are constructed around both an intensity index and a validity index to show associations among implications that are significant at specific levels.

4. Results

The results of the study that answer the research questions by identifying aspects that characterise AT in its semiotic, discursive and instrumental dimensions and that influence better achievement are described below.

4.1. Characterising algorithmic thinking

To answer the first research question (RQ1), we analysed the three tasks, two from Test A and one from Test B of the total group of students, exploring the similarities and cohesion between the variables to identify clusters and establish R-rules that allow us to assert associations between variables and between dimensions.

4.1.1. Task 1 of test A

Fig. 7 shows the results of the similarity analysis in Task 1. The variables (listed in 3.3.) grouped into two classes are underlined in

red.

We can distinguish two clusters containing the most similar variables to each other.

The variables gathered up to level 7 have a similarity index of more than 50% and the significant node is the union of variables from the semiotic and discursive dimensions: (S1 (S4 D1)). Hence, the inference we can draw from these relations is that 'people who use loops and alternatives (S1) tend to make correct use of representations (S4) and establish correct assignments (D1) and vice versa'. The variables gathered at the level with an index higher than 80% are the union of variables of the discursive dimension ((D2 D3) D4), the deduction we can obtain is: 'usually people who pay attention to efficiency (D2) and divide the problem into simpler problems (D3) tend to structure the previous thought (D4)'.

We observe in the cohesive tree (Fig. 8), the 11 variables have been structured in different classes that establish R-rules based on the cohesion between them.

This non-symmetrical classification of the classes brings us closer to the reality of the data, only those variables with an implicative intensity greater than 0.5 have been joined and then organised according to cohesion. We highlight the cluster ((D2 D3) D4) set at a level higher than 0.884 cohesion and in red in the tree, as it is a significant node. The significance of the cluster is in the association of variables of the discursive dimension and allows us to state: 'Generally, people who ensure efficiency (D2) and divide into simpler problems (D3) tend to have adequately structured (previous thinking) (D4)'.

4.1.2. Task 2 of test A

Fig. 9 shows the results of the similarity analysis in Task 2 of Test A.

As it can be seen, we can distinguish ten clusters containing the most similar variables. However, if we look at the values of the similarity indices and the significant levels at levels 1, 5, 8 and 10.

The variables gathered up to level 8 of classification have a similarity index of over 63% and the significant node is the union: ((REND (S3 (D2 I1))) (S2 D4)). Thereupon, the inference we can draw from these relationships is that 'good solvers who have mathematical knowledge (S3), they set up the algorithm efficiently (D2) and the programming language is used in a structured way (I1) tend to define computational variables (S2) and show prior structured thinking (D4)'.

The cohesive tree (Fig. 10) shows those variables with an implicative intensity greater than 0.78. The significant nodes are at levels 1 and 3. We highlight the cluster ((D2 I1) S2) established at a level above 0.992 of cohesion and significant. The meaning of the class is: 'Almost always, people who ensure efficiency (D2) and program in a structured way with precision in the language (I1) usually have set the computational variables (S2)'. This cluster integrates variables from all three dimensions, showing the relationship that an instrumental and discursive domain favours the semiotic dimension.

4.1.3. Task 3 of test B

Fig. 11 shows the results of the similarity analysis in Task 3 Test B. Variables grouped into three classes are marked in red.

The variables gathered up to classification level 9 have a similarity index of more than 69% and the significant node is the union: (D3 I2). Thus, the inference we can draw from these relationships is that 'quite a few of those who divide the problem into simpler problems (D3) usually perform the explanation of the steps (I2)'.

The variables gathered up to level 6 of classification have a similarity index higher than 93% and the significant node is the union: ((REND ((S3 D4) (D1 (D2 I1)))) S4), where variables from all three dimensions are involved. Then, the deduction we can draw from these relationships is that: 'people who solve well (REND), have mathematical knowledge (S3), perform structured prior thinking (D4), make correct use of assignments (D1), pose the algorithm efficiently (D2) and the language is used in a structured way (I1) tend to make correct use of representations (S4)'.

In the cohesive tree (Fig. 12) we highlight the cluster ((S3 D4) S2) set at a level higher than 0.99 of cohesion and the cluster ((D2 I1) (D1 S4)) set at a level of cohesion: 0.987. The meaning of this last cluster can be expressed by associating variables from the three dimensions as: 'Almost always, people who ensure efficiency (D2) and program structurally (I1) tend to make correct use of assignments (D1) and correct use of representations (S4)'.

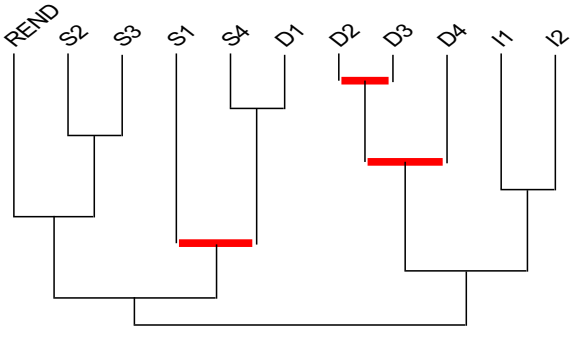


Fig. 7. Similarity tree task 1, test A.

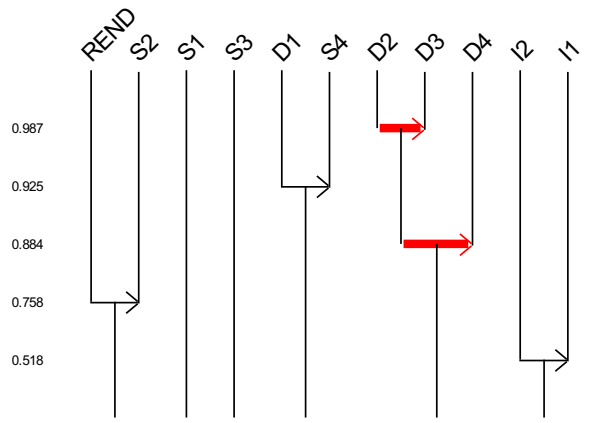


Fig. 8. Cohesive tree task 1, test A.

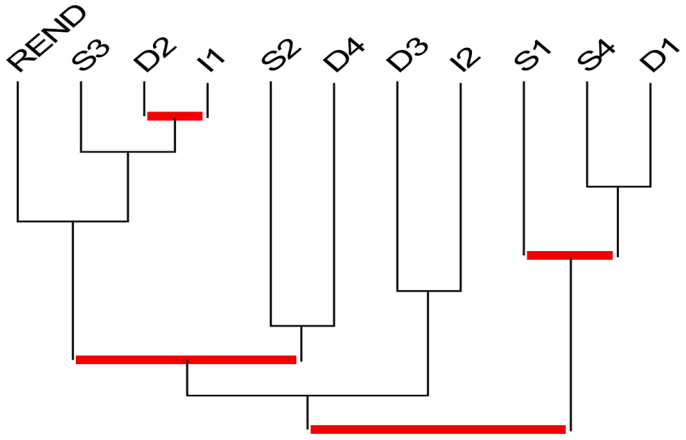


Fig. 9. Similarity tree task 2, test A.

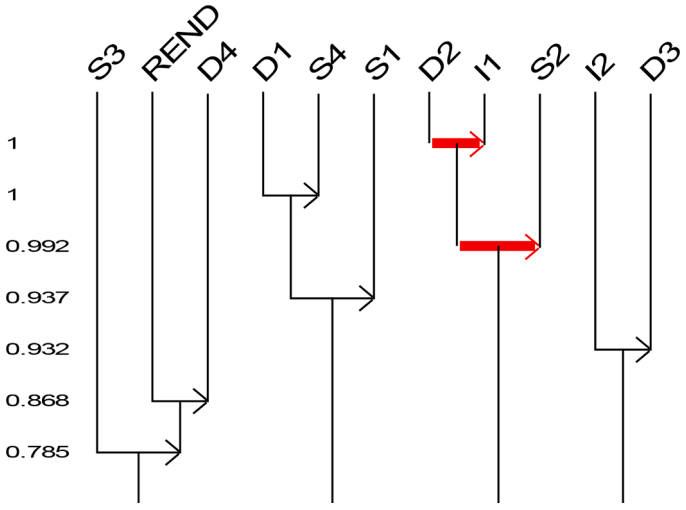


Fig. 10. Cohesive tree task 2 test A.

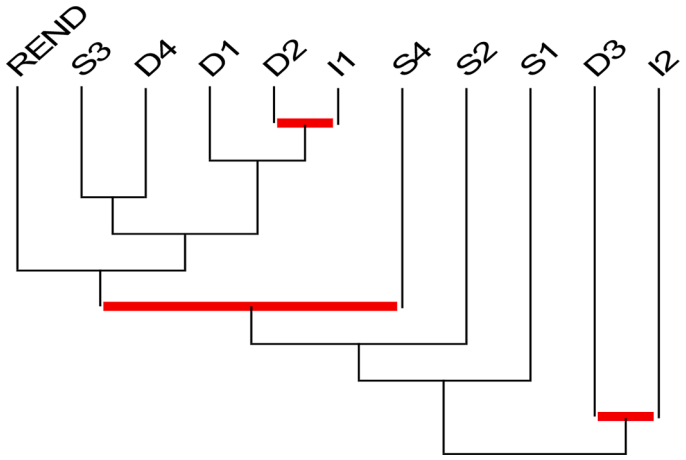


Fig. 11. Similarity tree task 3, test B.

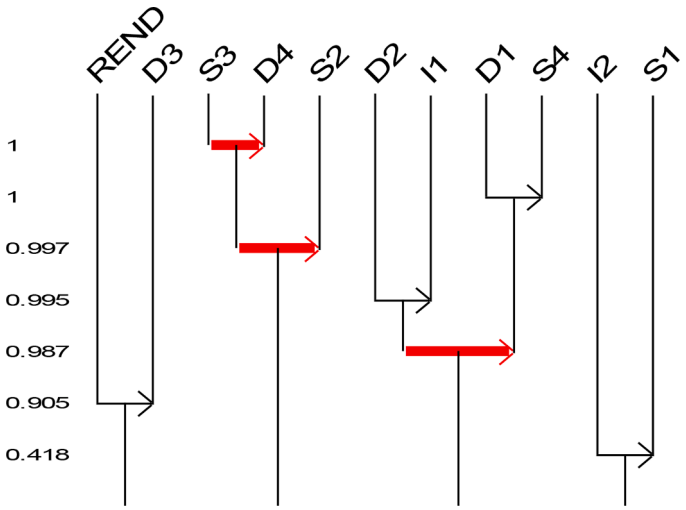


Fig. 12. Cohesive tree task 3 test B.

4.2. Semiotic, discursive and instrumental dimensions and achievement

We make a comparison of the three Tasks considering the implication graph (Fig. 13) of each one. This will allow us to go deeper into the different dimensions of thinking and their relation to achievement (Research Question 2). The implication graphs show those rules whose intensity is greater than 75% and are distributed in such a way that: the red arrow encompasses the implications with an intensity greater than 90%, the blue arrow encompasses the rules with an intensity between 85% and 90%, the green arrow encompasses those between 80% and 75%, and the grey arrow encompasses those between 50% and 75%. Some interesting properties can be observed in the graphs that allow us to identify relationships with achievement.

The hierarchical tree of the three Tasks indicates a causal relationship with a reliability rate of at least 0.75%.

For Task 1 of Test A the relationships can be synthesised into three chains that have some common variables:

Achievement (REND) -(0.75)→ Programming structure (I1)-(0.80)→Division into simpler problems (D3) -(.) 80→Structuring prior thinking (D4).

For Task 2 of Test A, we can identify two strings:

Achievement (REND) -(0.80)→Division into simpler problems (D3) -(0.85) →Structured (prior thinking) (D4) -(0.80) → Use of loops and alternative (S1).

In Task 3 of Test B, the relationships between achievement and variables are as follows:

Achievement (REND) -(0.85) → Explanation of steps (I2) -(0.80) → Use of loops and alternative (S1) -(0.85) → Structured (prior thinking) (D4)-(0.85) →Division into simpler problems (D3)-(0.85) →Correct use of representations (S4) -(0.85) →Computational variables (S2).

According to the implicational analysis, implication chains show category convergence independently of tasks, so it could be

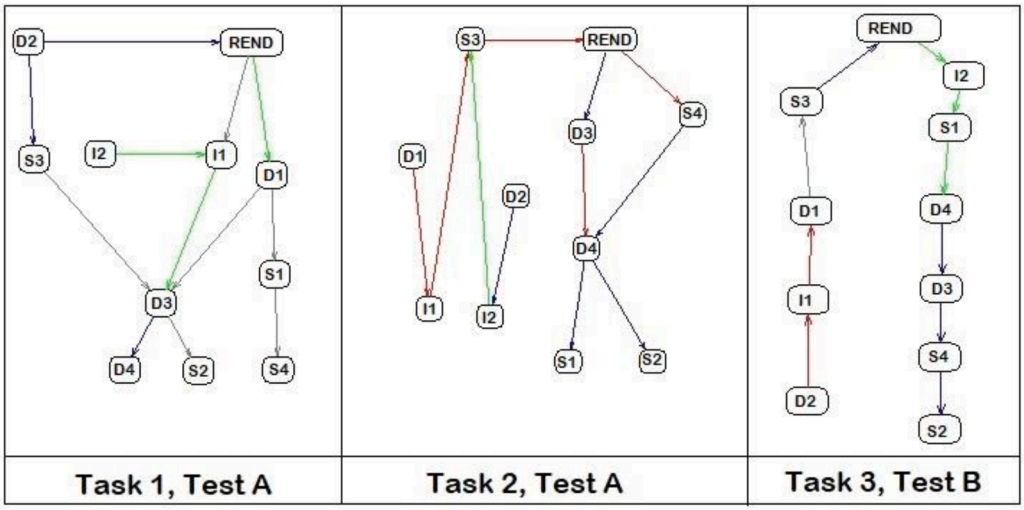


Fig. 13. Implication graphs of task 1, test A, task 2, test A and task 3, test B.

argued that they are prerequisites for task success.

5. Discussion

The main objective of this study was to characterise AT from a mathematical orientation in unplugged activities. Several questions were established, focusing on three dimensions and a set of categories that allow us to describe them and give consistency to the definition of AT: semiotic, instrumental and discursive dimensions. The main conclusions were the following: (1) the need to articulate these three dimensions in order to characterise AT, the discursive dimension plays an essential role, supported by the semiotic and instrumental dimensions, (2) the circularity or association between the categories of these dimensions as prescriptive variables in achievement (3) the effectiveness of the Theory of *Mathematical Working Space* and the breakdown of categories for the proposed model of algorithmic analysis (or the establishment of the *Algorithmic Working Space*), (4) the effectiveness of the use of unplugged tasks for the development of AT.

In relation to the first result, the similarity analysis of the different tasks shows that in the nodes with the highest similarity index of all the tasks, category D2 (Efficiency) is always involved. This indicates that the fact of making an optimal algorithm with as few instructions as possible is fundamental for AT, an element already highlighted by Knuth (1985). This also shows us that, depending on the problem, this category which belongs to the discursive dimension is related to other categories of both this dimension and semiotics.

It is noteworthy in the results of Task 3 that the four categories of the discursive dimension are related with a similarity of more than 80%, giving rise to a rule that expresses a form of student behaviour with respect to this dimension: 'usually people who pay attention to efficiency and divide the problem into simpler problems tend to structure prior thinking'. This rule seems to contain relevant elements to make sense of the categories associated with AT, and it seems to point quite accurately to a way forward for classroom work with students.

The results of the similarity graphs in Tasks 2 and 3 show an interaction of achievement with different categories belonging to the three dimensions, with a similarity of more than 63% and 93% respectively. Therefore, this relationship associates the other categories and indicates that when the similarity relationship between the categories is high and achievement is involved, all three dimensions are necessary for the student's achievement to be good. It is noteworthy that in these two associations, categories from all three dimensions are shared: S3 (Mathematical knowledge), D2 (Efficient), D3 (Division into small problems) and I1 (Structured programming). This is in line with Hiebert and Lefevre (1986), where both procedure and concepts are necessary for adequate learning of mathematics.

Likewise, there is an association between the discursive and instrumental dimensions in Task 3 with a similarity of more than 69%, where the fact of dividing the problem into smaller problems is related to the explanation of the steps that are being carried out. This result reinforces the need to articulate the three dimensions to operate not only intuitively but also to consolidate the formal sequence. The execution of an algorithm is closely linked to the need to decompose the problem into simpler ones that allow us to get to the desired solution and, therefore, if there is an explanation of these, the student can check whether what he/she is doing is what he really wants to do.

The different results we obtain from the tasks are consistent with the characterisation proposed. The fact that there is only one category, efficiency, which is transversal to the three tasks can have different interpretations: the time span between the tasks, the different mathematical approach between them or the difficulty involved for the student. However, the fact that efficiency is the category they share presents a significant scenario about the importance of this category in AT (Knuth, 1997; Peña Marí, 2006).

All of the above is reinforced in the cohesive analysis, where here we do have directional relationships, and we see which category influences another. Again, category D2 (efficiency) of the discursive dimension appears in all of them, being influenced by prior structuring (D4), the use of computational variables (S2) and the correct use of assignments (D1) and representations (S4). Thus, the efficiency of the algorithm is fundamentally affected by categories of the discursive and semiotic dimension. And not only the aforementioned categories appear, but also other categories related to efficiency appear in the cohesive analysis, which shows that the efficiency of the algorithm is related in one way or another to almost all the remaining categories (Knuth, 1985).

Regarding the second result, the circulation between dimensions and possible prescriptions on achievement, the implicative analysis of the tasks has allowed us to establish rules prescribing the influences of the dimensions (their categories) on achievement with a reliability rate of at least 75%. In Task 3 we obtain an interesting chain that indicates that who uses computational variables (S2) makes a correct use of representations (S4), which in turn divides the problem into smaller problems (D3), which makes a structuring of the previous thinking (D4), which indicates that he/she makes use of loops and alternatives (S1), then makes explanation of the steps (I2) and therefore has a high achievement (REND). This result points to a quite clear scenario about the categories that play a more fundamental role in the student's achievement, as in the implicative analysis of the rest of the tasks all these categories also appear. The relationship between the three dimensions and their different categories and the need to work on each of them in order for AT to be fully developed is once again evident. This confirms our assumption that AT can be developed independently of learning programming (Futschek, 2006).

Moreover, the implication graphs of the problems emphasise categories linked to structuring and language and their meaning. In other words, the algorithm, from a mathematical point of view, could be correct and well-focused, but when translated into formal language, errors appear. These errors must be differentiated because they do noy have the same importance. For example, some students use commands that do not exist or use functions that have nothing to do with what is expected. The study shows a lack of knowledge associated with the symbolic mapping of processes and the translation of natural language into formal language. This result coincides with Couderette (2016), where the coexistence of mathematics and computer science in algorithmic thinking can lead to problems.

In addition, the associated information structures fail so that AT does not occur. Another example of the same nature is when the learner makes an error in naming an element. In these cases, the algorithm is correct, but does not work because of these typing (syntax) errors. Implication graphs highlight the assignment symbolised as a causal relationship to achievement therefore, training in certain aspects of algorithmic thinking is necessary to improve it (Cachero et al., 2020).

With regard to the third result, which refers to the methodological theoretical proposal that supports the breakdown of categories for an analysis of thinking in the classroom, we consider that it has been effective both to have the theory of *MWS* as a theoretical framework, as well as the inductive procedure for obtaining the descriptive categories of the AT. The initial analysis carried out on the different solutions given to Task 1 of Test A has provided consistency for an operational formulation of the categories. The validity of this formulation was consolidated with the application to the other tasks.

Finally, it is worth remarkable in the results what role the fact of carrying out unplugged activities may have played. As we have indicated, writing the algorithm on paper rather than on a computer has its limitations as it means that the student cannot check in situ what he/she is doing, seeing if the steps make sense and where the possible failure lies. However, doing the task on paper allows the student to effectively develop abstract thinking and the need to put the solution in writing (Battal et al., 2021). The latter is interesting for the development of CT and in particular for AT, where the prior structure of the thinking is related to the underlying mathematical thinking and requires conceptual clarification of the content and the sequence in order to correctly perform the algorithm.

Similarly, the unplugged activities have allowed us to observe repetition patterns and behaviours in students that allow us to identify possible weaknesses in order to improve AT. In addition, we have been able to work on the different patterns that appear in the tasks to conclude thinking structures and problems that transcend the mathematical field and to be able to make a relationship between the mathematical and computational context. All this has been fundamental for the correct characterisation of the AT that has been offered, the aspects to work on to improve achievement.

6. Conclusions

The study of AT is a relatively recent topic and much remains to be explored. Throughout the work, the need and usefulness of a characterisation to improve our knowledge of this field was highlighted. The results have shown that the categories provided are relevant to the proposed objective and that they can also help to detect weaknesses and strengths in students' learning.

Similarly, the unplugged activities have provided a very useful tool for the study, allowing the proposed exercises to observe skills and behaviours in the students, as well as pointing out patterns that have supported the characterisation of AT.

All of this has allowed us to make progress on this topic, providing tools for its study and work with university students.

It could be considered a limitation of the study that we focused on first-year Mathematics students and unplugged activities. It would be interesting to analyse these categories in plugged activities, to assess how the use of a computer can affect a particular dimension, and if, as we have indicated here, efficiency continues to play an essential role. Similarly, extending these analyses to other courses and other degrees in order to see if the proposed characterisation is extensible as expected and which elements stand out according to the student profile.

Data availability

The datasets analysed during the current study are not publicly available due to individual privacy.

Author agreement

The authors declare that the paper is original and non-published.

CRediT authorship contribution statement

Adrián Bacelo: Conceptualization, Methodology, Validation, Investigation, Data curation, Visualization, Writing – original draft, Writing – review & editing. **Inés M. Gómez-Chacón:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing.

Declaration of Competing Interest

The authors declare that there is no conflict of interest.

Data availability

The data that has been used is confidential.

Acknowledgments

This study was partially support by the Science and Innovation Minister under project PID2021-122752NB-I00 and by the European Commission under European Project SUPERA (European Commission, Call Research H2020 (Contract No.: 787829)) We would also like to thank J. Carmona from the Department of Computer Systems and Computing of the Faculty of Mathematical Sciences of the UCM.

References

Adell, J. S., Llopis, M. A. N., Esteve, M. F. M., & Valdeolivas, N. M. G. (2019). El debate sobre el pensamiento computacional en educación. *Revista Iberoamericana de Educación a Distancia, 22*(1), 171–186. https://doi.org/10.5944/ried.22.1.22303

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832–835. https://doi.org/10.1093/comjnl/bxs074

Battal, A., Afacan Adanır, G., & Gülbahar, Y. (2021). Computer science unplugged: A systematic literature review. *Journal of Educational Technology Systems, 50*(1), 24–47. https://doi.org/10.1177/00472395211018801

Burton, B. (2010). Encouraging algorithmic thinking without a computer. *Olympiads in Informatics, 4*, 3–14.

Cachero, C., Barra, P., Meliá, S., & López, O. (2020). Impact of programming exposure on the development of computational thinking capabilities: An empirical study. *IEEE Access, 8*, 72316–72325. 10.1109/ACCESS.2020.2987254.

Cobb, P., Confrey, J., diSessa, A., Lehrer, R., & Schauble, L. (2003). Design experiments in educational research. *Educational Researcher, 32*(1), 9–13. https://doi.org/10.3102/0013189X032001009

Cohors-Fresenborg, E. (1993). Register machine as a mental model for understanding computer programming. In E Lemut, B du Boulay, & G Dettori (Eds.), *Cognitive models and intelligent environments for learning programming* (pp. 235–248). Berlin: Springer.

Cooper, S., Dann, W. P., & Pausch, R. F. (2000). Alice: A 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges, 15*, 107–116.

Couderette, M. (2016). Enseignement de l'algorithmique en classe de seconde : Une introduction curriculaire problématique. *Annales de Didactique et de Sciences Cognitives, 21*, 267–296.

Couturier, R., Bodin, A. & Gras R. (2000). *Classification hiérarchique implicative et cohésitive*. Rennes: Association pour le Recherche en didactique des mathématiques.

Drijvers, P., Kodde-Buitenhuis, H., & Doorman, M. (2019). Assessing mathematical thinking as part of curriculum reform in the Netherlands. *Educational Studies in Mathematics, 102*, 435–456. https://doi.org/10.1007/s10649-019-09905-7

Erümit, A. K., & Sahin, G. (2020). Plugged or unplugged teaching: A case study of students' preferences for the teaching programming. *International Journal of Computer Science Education in Schools, 4*(1), 3–32.

Ferreira-Szpiniak, A., Luna, C. D., & Medel, R. H. (1997). Una propuesta de integración de nociones lógico-matemáticas en la enseñanza de la Programación. In *, 2. Proceedings del III Congreso Argentino de ciencias de la computación, CACIC'97* (pp. 881–892). La Plata, Argentina.

Figueiredo, M. P., Amante, S., Gomes, H., Gomes, M. C., Rego, B., Alves, V., & Duarte, R. P. (2021). Algorithmic thinking in early childhood education: Opportunities and supports in the Portuguese context. In L. Gómez Chova, A. López Martínez, & I. Candel Torres (Eds.), *EduLearn 2021 proceedings* (pp. 9339–9348). IATED. https://doi.org/10.21125/edulearn.2021.1885.

Fraillon, J., Ainley, J., Schulz, W., Friedman, T., & Duckworth, D. (2020). *Preparing for life in a digital world*. Springer. https://doi.org/10.1007/978-3-030-38781-5

Futschek, G. (2006). Algorithmic thinking: The key for understanding computer science. Mittermeir, R.T. (eds) Informatics education – The bridge between using and understanding computers, ISSEP 2006. Lecture notes in computer science, 4226 (pp. 159–168), Springer, Berlin, Heidelberg. https://doi.org/10.1007/11915355_15 AUTHORS.

Gal-Ezer, J., & Lichtenstein, O. (1996). A mathematical-algorithmic approach to sets: A case study. *Mathematics and Computer Education, 31*(1), 33–42.

Georgiou, K., & Angeli, C. (2021). Developing computational thinking in early childhood education: A focus on algorithmic thinking and the role of cognitive differences and scaffolding. In D. Ifenthaler, D. G. Sampson, & P. Isaías (Eds.), *Cognition and exploratory learning in the digital age* (pp. 33–49). Cham: Springer. https://doi.org/10.1007/978-3-030-65657-7_3.

Gómez-Chacón, I. M., & Kuzniak, A. (2015). Geometric Work Spaces: figural, instrumental and discursive geneses of reasoning in a technological environment. *International Journal of Science and Mathematics Education, Singapore, 13*(1), 201–226.

Gras, R., Suzuki, E., Guillet, F., & Spagnolo, F. (2008). *Statistical implicative analysis, theory and applications, 127*. Berlin-Heidelberg: Springer-Verlag. Studies in Computational Intelligence.

Gubo, S., & Végh, L. (2021). Assessment of algorithmic thinking of Slovak and Hungarian secondary school students: results of a pilot study. In *ICERI2021 proceedings* (pp. 2924–2933).

Hiebert, J., & Lefevre, P. (1986). Conceptual and procedural knowledge in mathematics: An introductory analysis. In J. Hiebert (Ed.), *Conceptual and procedural knowledge: the case of mathematics* (pp. 1–27). Lawrence Erlbaum Associates, Inc.

Hill, R. K. (2016). What an algorithm is. *Philosophy & Technology, 29*, 35–59. https://doi.org/10.1007/s13347-014-0184-5

Hoyles, C., & Noss, R. (2015). A computational lens on design research. *ZDM Mathematics Education, 47*, 1039–1045. https://doi.org/10.1007/s11858-015-0731-2

Hromkovič, J., Kohn, T., Komm, D., & Serafini, G. (2016). Examples of algorithmic thinking in programming education. *Olympiads in Informatics, 10*, 111–124. 10.15388/ioi.2016.08.

Kanaki, K., & Kalogiannakis, M. (2022). Assessing algorithmic thinking skills in relation to age in early childhood STEM education. *Education Sciences, 12*(6), 380. https://doi.org/10.3390/educsci12060380

Kim, J. (2018). A study on systematic review of unplugged activity. *Journal of The Korean Association of Information Education, 22*(1), 103–111. https://doi.org/10.14352/jkaie.2018.22.1.103

Kirçali, A.Ç., & Özdener, N. (2022). A comparison of plugged and unplugged tools in teaching algorithms at the K-12 level for computational thinking skills. *echnology, Knowledge and Learning*. https://doi.org/10.1007/s10758-021-09585-4

Knuth, D. E. (1985). Algorithmic thinking and mathematical thinking. *The American Mathematical Monthly, 92*(3), 170–181.

Knuth, D. E. (1997). *The art of computer programming volume 1 (3rd ed.): Fundamental algorithms*. Redwood City: Addison Wesley Longman Publishing Co. Inc. ISBN 0-201-89683-4.

Korkmaz, Ö., Çakır, R., Özden, M., Oluk, A., & Sarıoğlu, S. (2015). Investigation of individuals' computational thinking skills in terms of different variables. *Journal of Ondokuz Mayıs University Education Faculty, 34*(2), 68–87.

Kuzniak, A., Tanguay, D., & Elia, I. (2016). Mathematical working spaces in schooling: An introduction. *ZDM Mathematics Education, 48*(6), 721–737. https://doi.org/10.1007/s11858-016-0812-x. AUTHORS.

Lafuente Martínez, M., Lévêque, O., Benítez, I., Hardebolle, C., & Zufferey, J. D. (2022). Assessing computational thinking: Development and validation of the algorithmic thinking test for adults. *Journal of Educational Computing Research, 60*(6), 1436–1463. https://doi.org/10.1177/07356331211057819

Lagrange, J. B. (2020). Algorithmics. In S. Lerman (Ed.), *Encyclopedia of mathematics education* (pp. 32–36). Cham: Springer. https://doi.org/10.1007/978-3-030-15789-0_100044.

Lagrange, J. B., & Laval, D. (2019). Connected Working spaces: the case of computer programming in mathematics education. *Eleventh congress of the European society for research in mathematics education*. Utrecht, Netherlands: Utrecht University. hal-02418181f.

Lagrange, JB., & Laval, D. (2023). Connecting algorithmics to mathematics learning: a design study of the intermediate value theorem and the bisection algorithm. *Educational Studies in Mathematics, 112*, 225–245. https://doi.org/10.1007/s10649-022-10192-y

Liu, J. & Wang, L. (2010). Computational thinking in discrete mathematics. In Education technology and computer science, International Workshop on Wuhan, Hubei, China, 413–416, doi: 10.1109/ETCS.2010.200.

Lockwood, E., Delarnette, A. F., Asay, A., & Thomas, M. (2016). Algorithmic thinking: An initial characterization of computational thinking in mathematics. In *Proceedings of the 38th annual meeting of the North American chapter of the international group for the psychology of mathematics education* (pp. 1588–1595). Tucson: AZ: The University of Arizona.

Lockwood, E., & De Chenne, A. (2020). Enriching students' combinatorial reasoning through the use of loops and conditional statements in Python. *International Journal of Research in Undergraduate Mathematics Education, 6*(3), 303–346. https://doi.org/10.1007/S40753-019-00108-2

Montes-León, H., Hijón-Neira, R., Pérez-Marín, D., & Montes-León, R. (2020). Mejora del pensamiento computacional en estudiantes de secundaria con tareas unplugged. *Education in the Knowledge Society, 21*, 24, 10.14201/eks.23002.

Montoya Delgadillo, E., & Vivier, L. (2014). Les changements de domaine dans le cadre des espaces de travail mathématique. *Annales de Didactique et de Sciences Cognitives, 19*, 73–101.

Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED-Revista de Educación a Distancia, 46*, 1–23. AUTHORS.

Nechache, A., Gómez-Chacón, I.M. (2022). Methodological Aspects in the Theory of Mathematical Working Spaces. In: Kuzniak, A., Montoya-Delgadillo, E., Richard, P.R. (eds) *Mathematical Work in Educational Context. Mathematics Education in the Digital Era*, vol 18. (pp.33-56). Springer, Cham. https://doi.org/10.1007/978-3-030-90850-8_2.

Peña Marí, R. (2006). *De euclides a Java. Historia de los algoritmos y de los lenguajes de programación*. Madrid: Nivola. AUTHORS.

Posso, M.E. (2022) *Las "actividades desconectadas" y el desarrollo del pensamiento algorítmico* [Trabajo de Grado, Universidad Católica de Pereira]. http://hdl.handle.net/10785/9635.

Pérez-Marín, D., Hijón-Neira, R., Bacelo, A., & Pizarro, C. (2020). Can computational thinking be improved by using a methodology based on metaphors and Scratch to teach computer programming to children? *Computers in Human Behavior, 105*, 105849. https://doi.org/10.1016/j.chb.2018.12.027.

Prediger, S., Gravemeijer, K., & Confrey, J. (2015). Design research with a focus on learning processes: an overview on achievements and challenges. *ZDM, 47*(6), 877–891. https://doi.org/10.1007/s11858-015-0722-3

Shin, N., Bowers, J., Krajcik, J., et al. (2021). Promoting computational thinking through project-based learning. *Disciplinary and Interdisciplinary Science Education Research, 3*, 7. https://doi.org/10.1186/s43031-021-00033-y

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003. ISSN 1747-938X.

Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Computational thinking in high school science classrooms: Exploring the science 'Framework' and 'NGSS'. *Science Teacher, 81*(5), 53–59.

Stephens, M. (2018). Embedding algorithmic thinking more clearly in the mathematics curriculum. In Y. Shimizu, & R. Withal (Eds.), *Proceedings of ICMI study 24 school mathematics curriculum reforms: challenges, changes and opportunities* (pp. 483–490). University of Tsukuba.

Stephens, M., & Kadijevich, D. M. (2020). Computational/algorithmic thinking. In S. Lerman (Ed.), *Encyclopedia of mathematics education* (pp. 117–123). Cham: Springer. https://doi.org/10.1007/978-3-030-15789-0_100044.

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education, 148*, Article 103798. https://doi.org/10.1016/j.compedu.2019.103798. ISSN 0360-1315.

van Borkulo, S., Chytas, C., Drijvers, P., Barendsen, E. & Tolboom, J. (2021). Computational thinking in the mathematics classroom: Fostering algorithmic thinking and generalization skills using dynamic mathematics software. *In The 16th Workshop in Primary and Secondary Computing Education (WiPSCE '21). Association for Computing Machinery*, New York, NY, USA, Article 19, 1–9. https://doi.org/10.1145/3481312.3481319.