Computers in Human Behavior Reports 5 (2022) 100166

Contents lists available at ScienceDirect

Computers in Human Behavior Reports

journal homepage: www.sciencedirect.com/journal/computers-in-human-behavior-reports

The CT-cube: A framework for the design and the assessment of computational thinking activities

Alberto Piatti [a], Giorgia Adorni [b,*], Laila El-Hamamsy [c], Lucio Negrini [a], Dorit Assaf [d], Luca Gambardella [b], Francesco Mondada [c]

[a] Department of Education and Learning (DFA), University of Applied Sciences and Arts of Southern Switzerland (SUPSI), Locarno, Switzerland
[b] Dalle Molle Institute for Artificial Intelligence (IDSIA), USI-SUPSI, Lugano, Switzerland
[c] Mobile Robotic Systems Group (MOBOTS), Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
[d] School of Education, University of Applied Sciences and Arts Northwestern Switzerland, Windisch, Switzerland

ARTICLE INFO

Keywords:
Computational thinking
Algorithmic skills development
Assessment
K-12
Situated cognition

ABSTRACT

Computational thinking is a fundamental competence that is being introduced in K-12 and succeeding curricula worldwide. Despite this huge effort, many computational thinking models in the literature do not explicitly take into consideration the pupils' age and the developmental nature of computational thinking skills. Furthermore, many existing computational thinking models are focused on the internal thinking processes of the individuals, failing to explicitly consider the situated nature, in terms of social context and artefactual environment, that usually characterise tasks that require computational thinking to be solved. In this paper, we present a framework for the design, realisation, analysis, and assessment of computational thinking activities, called CT-cube. The CT-cube allows to extend existing computational thinking models to consider the life-long development of computational thinking skills in individuals, from childhood to adult age, and to take into consideration the situated nature of computational thinking activities. We use the CT-cube to design an unplugged task, called Cross Array Task (CAT), allowing to assess the algorithmic skills of K-12 pupils and we show how the CT-cube can be used in this case to illustrate the development of this competence along the entire compulsory school path, considering a sample of 109 pupil aged 3 to 16 in Switzerland.

1. Introduction

Jeannette Wing, in (Wing, 2006), has argued that *"computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability"*. In particular, he defines computational thinking as *"the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer – human or machine – can effectively carry out"* (Wing, 2014). In the last two decades, an impressive amount of research has been devoted to the study of the different facets of computational thinking. Surprisingly, this research has not followed the same direction as researchers that have investigated development, teaching and learning of other basic skills, like "reading, writing or arithmetic". For example, to the best of our knowledge, there is no complete account on how computational thinking is acquired and developed along the entire life span, starting from the very beginning of life, as the numerous and heterogeneous definitions of computational thinking that are present in the literature rarely take into account explicitly the age or the competence level of the individuals (Shute et al., 2017; Tikva & Tambouris, 2021). Furthermore, several components that are very often associated with computational thinking, like for example abstraction (Shute et al., 2017; Wing, 2014) are clearly beyond the potential of very young individuals. To investigate the development of computational thinking along the entire learning path of individuals, it is necessary to define computational thinking taking explicitly into consideration the developmental nature of thinking, the skills and the age of individuals, the social context in which computational thinking activities take place and the kind of artefacts that are at disposal of the individuals (artefactual environment).

In this paper, we argue that models of the development of mathematical thinking could also be used to model the development of computational thinking. In particular, we extend a model introduced by

* Corresponding author.
E-mail addresses: alberto.piatti@supsi.ch (A. Piatti), giorgia.adorni@idsia.ch, giorgia.adorni@usi.ch (G. Adorni), laila.elhamamsy@epfl.ch (L. El-Hamamsy), lucio.negrini@supsi.ch (L. Negrini), dorit.assaf@phsg.ch (D. Assaf), luca.gambardella@idsia.ch (L. Gambardella), francesco.mondada@epfl.ch (F. Mondada).

https://doi.org/10.1016/j.chbr.2021.100166
Received 28 July 2021; Received in revised form 10 December 2021; Accepted 27 December 2021
Available online 6 January 2022

David Tall for the modelling of the development of mathematical thinking (Tall, 2006, 2013, 2020) to computational thinking and propose a general definition of computational thinking based on the theoretical framework of situated cognition, allowing to take explicitly into consideration, theoretically and in practice, the developmental and situated nature of computational thinking. The resulting model, called *computational thinking cube* (CT-cube in short), can be applied for the design, the realisation, the analysis and the assessment of computational thinking activities. The CT-cube is then applied to define a particular task, the *Cross Array Task* (CAT), designed to assess the algorithmic skills of pupils and, consequently, investigate the development of these components of computational thinking along the entire school path, and to analyse the results collected in several compulsory school classes in Switzerland.

2. Theory

2.1. A definition of computational thinking based on situated cognition

Consider the two terms present in the expression *computational thinking*: computational thinking is *thinking* and has consequently to do with cognition; computational thinking is computational, and has therefore to do with computation, and consequently with *algorithms* (Rapaport, 2015). Roughly speaking, we could define *computational thinking as the cognitive activity required to solve problems through algorithms*. This intuitive definition is very similar to that of (Wing, 2014) quoted in the introduction: *"the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer – human or machine – can effectively carry out"*.

It is important to highlight that, in our view, consistently with the original definition in (Wing, 2014), computational thinking is not strictly related to the presence of artificial agents (e.g., computers), but to that of an algorithm. Actually, we consider a computational thinking problem to be an activity whose solution consists in an algorithm that should be performed by an artificial or human agent to realise a specific task and therefore computational thinking activities, encompassing thus also unplugged tasks which require the use of algorithms to be realised.

The difference in the definition that we present in this paper and the original one of Wing, is rooted in the particular formulation of cognition and on the developmental point of view that we adopt. While the most known definitions of computational thinking (Shute et al., 2017) are based on the components of thinking that should be activated by an individual to solve a computational task, and are therefore based on the classical definition of cognition as a process occurring in the head of a single individual, i.e., the "thought processes" in the definition of Wing, we base our formulation on the basic concepts of situated cognition (Roth & Jornet, 2013): cognitive activities are embodied, enacted and embedded in a situated cognitive system (Heersmink, 2013), consisting of a social context and/or an artefactual environment. In other words, cognitive activities do not occur in the head of a single individual, but are shared social practices, mainly based on representations and manipulation of knowledge and information through external cognitive artefacts. We argue that this theoretical setting corresponds better to the concrete settings in which computational thinking is activated (in education but also in general), that are usually characterised by the simultaneous presence of several persons (social setting) and a rich artefactual environment.

Consequently, we define computational thinking as a **situated cognitive activity** (individual or collective), **consisting of three (eventually iterative) steps**: (1) setting a **contextualised problem** in such a way that its solution can be computed (**problem setting**), (2) conceiving and representing an algorithm, that should be implemented by an agent (human, artificial and/or virtual agent) in order to solve the problem (**algorithm**), and (3) assessing the quality of the solution with respect to the original problem (**assessment**).

From this point of view, the cognitive skills of an individual can be inferred by observing her/him solving a contextualised (computational) task together with other individuals in a given artefactual environment. In particular, the way an individual interacts with other individuals and her/his contribution to the collective reasoning, respectively their choice of (cognitive) artefacts and his/her ability to use them correctly and efficiently in the different steps, can be observed directly and used to assess his/her competence level with respect to the given task.

This particular definition allows to take explicitly into consideration the situated nature of computational thinking. In the next section, we propose a theoretical framework allowing to take into consideration also its developmental nature.

2.2. Mathematical thinking and the three worlds of mathematics of David Tall

Mathematical thinking is a multifaceted set of skills and attitudes, that is widely recognised as a fundamental component of human thinking that is developed from the very beginning of life, throughout the entirety of education to the adult age (Tall, 2013). Consider for example the act of counting, i.e., enumerating the elements of a set to determine its cardinality and the development of the concept of number (Dehaene, 2011; Gelman & Gallistel, 1986). These fundamental mathematical skills require a combination of innate and acquired abilities that are developed starting in the first years of life and are further developed and abstracted along the entirety of the school life and beyond. Indeed, pupils continuously refine these skills through various experiences ranging from their first counting experiences with concrete objects in pre-primary school and even before (Benoit et al., 2004; Bruce & Threlfall, 2004; Sarnecka & Carey, 2008), through the development of counting strategies, symbolisation, automatisation, abstraction, and so on, up to axiomatisation of natural numbers and the development of very complex counting algorithms.

The model of the three worlds of mathematics (Tall, 2006, 2013, 2020) has been defined to describe the "overall growth of mathematical thinking in individuals" (Tall, 2006, p. 195). The model is based on the idea that mathematical concepts and theories are acquired on a long-term, starting from concrete, embodied experiences, subsequently internalised through the compression of procedures into symbolic concepts (procepts), and finally abstracted through axiomatisation. In particular, Tall (2006) argues that,

> "the long-term construction of mathematical knowledge uses the power of the biological brain, with input through perception, output through action and the internal power of reflection to re-assemble ideas into usable mental structures. I hypothesise that mathematical thinking evolves through three linked mental worlds of mathematics, each with its own particular way of developing greater sophistication:
> - *an object-based conceptual-embodied world reflecting on the senses to observe, describe, define and deduce properties developing from thought experiment to Euclidean proof;*
> - *an action-based proceptual-symbolic world that compresses action schemas into thinkable concepts operating dually as process and concept (procept);*
> - *a property-based formal-axiomatic world focused to build axiomatic systems based on formal definitions and set-theoretic proof*" (Tall, 2006, *p. 197*)".

In the first world, reasoning is based on perception and embodiment; in the second world, calculation and symbolisation are the most important components; while in the third world, axioms, properties and logical deduction play the most important role. It is crucial to remark that the model is defined as incremental: the reasoning in a given world is always based (implicitly) on the experiences that the individual has lived in the preceding worlds.

If we look at the development of counting and of the concept of number outlined above, we recognise immediately the phases

corresponding to the three worlds. These stages can also be correlated to the individuals age. In case of counting and of the concepts of number, during the pre-primary school pupils reason mainly in the first world, in the primary school both the first and the second world are equally present, in the secondary school reasoning occurs mainly in the second world, and the third world is introduced gradually, while reasoning occurs mainly in the third world at tertiary level.

2.3. The three worlds of computational thinking

The sound acquisition of complex computational concepts, like for example parallelisation and iteration, requires a long-term learning path, similar to that should be followed by individuals to internalise complex mathematical concepts, like the concept of counting. From this point of view, we argue that the model of the three worlds of mathematics of Tall can be directly extended to the field of computational thinking, in particular identifying the cognitive artefacts that are used in the three worlds in the case of computational thinking with respect to mathematical thinking, respectively the analogies and differences between mathematical and computational concepts.

Consequently, analogously to the model of (Tall, 2020) for mathematical thinking, we hypothesise that computational thinking occurs in three worlds of computation, characterised by different types of cognitive and representational artefacts:

An **embodied world**, based on embodiment and perception, in which computational thinking is mainly focused on the solution of contextualised problems through ecological and iconic representational cognitive artefacts.

A **symbolic world**, based on the conception and description of procedures and rules for solving contextualised problems through symbolic (both formal and natural) cognitive artefacts.

A **formal world**, based on the generalisation and representation of algorithms through formal languages, in order to define structures that can be applied for problem solving in different, even yet unknown, contexts. This third world corresponds to the concept of abstraction of Wing: *"the most important and high-level thought process in computational thinking is the abstraction process. Abstraction is used in defining patterns, generalising from specific instances, and parametrisation. It is used to let one object stand for many. It is used to capture essential properties common to a set of objects while hiding irrelevant distinctions among them. For example, an algorithm is an abstraction of a process that takes inputs, executes a sequence of steps, and produces outputs to satisfy a desired goal"* (Wing, 2014, p. 2).

The proposed framework can be used to conceive long-term educational paths for the development of the different components of computational thinking, respectively for assessing the competence level of an individual with respect to the different components. In general, to internalise and abstract a given concept or component, an individual should go through all the three worlds. Consequently, the competence level of an individual with respect to a given computational thinking component should take into account the age and the expertise of the individual; there are in fact cognitive artefacts that are too complex and/or abstract for young pupils and/or inexperienced people.

2.4. The computational thinking cube

The Computational Thinking cube (CT-cube) is a three-dimensional $3 \times 3 \times 3$ matrix, illustrated in Fig. 1, that can be used, in combination with any computational thinking model, for the design, the realisation, the analysis and the assessment of computational thinking. The CT-cube is characterised by three dimensions, referred to each single individual in the situated cognitive system at each moment of the activity: (1) the type of **activity** that is being performed or that is required (problem setting, algorithm, assessment), (2) the (computational) world in which the activities take place, characterised by the **artefactual environment** that is being used (embodied, symbolic, formal) and (3) the **autonomy** of the individual with respect to the other individuals in the situated cognitive system (inactive or passive role, active support role or supported individual activity, active leading role or autonomous individual activity). It is important to remark that the three dimensions are easily observable in practice, making this framework suitable for the assessment of computational thinking skills.

The CT-cube combines two framings of computational thinking that have been considered extensively in literature: cognitive computational thinking and situated cognitive thinking (Kafai et al., 2020), but while in situated computational thinking, the focus is often concentrated on social and creative skills, the CT-cube adopts a situated cognition view, that consider the whole situated cognitive system, consisting of both the social context and the artefactual environment. In particular, given one or more components of computational thinking, according to a given

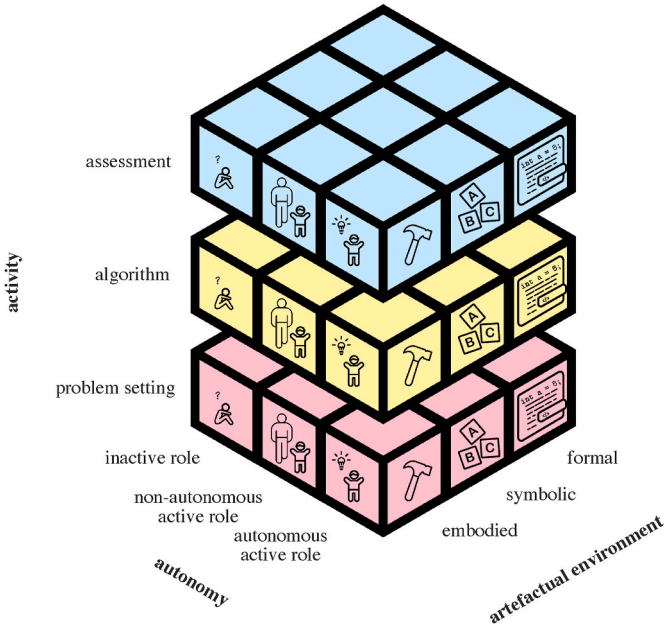


Fig. 1. The CT-cube.

computational thinking model, a task based on the selected components is designed and realised by explicitly structuring (i) the type of activity that is being performed, (ii) the artefactual environment and (iii) the social interactions and the level of autonomy, *a priori* and/or during the realisation of the activity.

More in detail, the three possible values of the dimension **activity** can be defined as follows:

1. **Problem setting**: this type of activity consists in recognising, understanding, contextualising, reformulating and/or modelling a problem such that its solution can be computed.
2. **Algorithm**: specifying a set of rules or instructions that should be adopted or followed by an executor (human, artificial and/or virtual agent) in order to realise a task that solve the problem.
3. **Assessment**: evaluate the quality and the suitability of the obtained solution.

The three possible values of the dimension **artefactual environment** correspond to the computational worlds defined above, i.e.,:

1. **Embodied**, based on embodiment and perception, artefactual environment based on ecological and iconic representational cognitive artefacts.
2. **Symbolic**, based on the conception and description of procedures through symbolic (both formal and natural) cognitive artefacts.
3. **Formal**, based on the generalisation and representation of algorithms through formal languages.

Finally, the three possible values of the dimension **autonomy** are the following:

1. **Inactive role**: the individual is not active because he is not able and/or motivated to realise the requested activity.
2. **Non-autonomous active role**: the individual is motivated for the activity and is able to realise the activity if helped, scaffolded or guided by other members of the situated cognitive system.
3. **Autonomous active role**: the individual is motivated for the activity and is able to realise autonomously the activity, respectively to guide and/or inspire the other members of the situated cognitive system in the realisation of their activities.

This model can be used to **design a computational thinking activity** (for example in an educational setting) structuring the situated cognitive system, i.e., setting the social context and the artefactual environment, and selecting the problem to be solved, in order to constrain the activities to a subset of the cells of the CT-cube.

It can be used for the **realisation of computational thinking activities**, in which the CT-cube is used as reference to actively shape the situated cognitive system in order to open or block the access to particular cells of the CT-cube, for example adding or eliminating a particular cognitive artefact from the artefactual environment.

It can be used to **analyse a computational thinking activity**, mapping the activity path of each single individual in a situated cognitive system with a given time frequency as a sequence of cells visited by the individual during the activity, for example through video analysis.

Finally, it can be used to **assess the computational thinking competence level of one or more individuals in the selected dimensions**, observing and comparing the subsets of the CT-cube visited by the individuals during the activity. Assessing several individuals with different age and school level at a given time with respect to the given task, or assessing the same individual with respect to the same (or very similar) task in different times, the model allows consequently to **assess the short- and long-time development of the given computational thinking dimensions**.

It is important to underline that the CT-cube is not a stand-alone computational thinking model, but an extension that can be applied to any computational thinking model, in order to take explicitly into account the developmental character and the situated nature of computational thinking activities and skills.

3. Materials and methods

In this section, we illustrate how the CT-cube can be used to design a task to assess the development of algorithmic skills in the compulsory school.

3.1. A practical application of the CT-cube: the CAT and the development of the algorithmic skills in compulsory school

The Cross Array Task (CAT) is a task designed to assess the development of algorithmic skills in the compulsory school, according to the definition of this component given by Shute et al. in (Shute et al., 2017). In particular, by *algorithm* we denote the skill of describing a complex procedure through a set of simpler instructions.

Because of the possible lack of technical skills and equipment of compulsory school pupils (in particular, the younger ones), we have decided to conceive an unplugged activity, without any artificial agent, in which the pupil should conceive and communicate an algorithm, using natural language and gestures, to another human being (a researcher in the case of this study) that is called to interpret and realise it.

By means of the CT-cube, we design the CAT, an unplugged task for assessing the development of algorithmic skills in compulsory school (K-12 pupils). In the computer science community, there is an open debate about/on whether computational thinking can be developed or assessed using unplugged activities. The main advantage of an unplugged assessment is that the use of technology is not required (Bell, 2018; Kalelioglu et al., 2016; Relkin et al., 2021), making it suitable for a huge number of schools around the world that do not have basic technology infrastructure (Brackmann et al., 2017; Unnikrishnan et al., 2016; Walletet al., 2015) — electronic devices, such as computers or tablet, or even Internet.

Moreover, it can be administered even to young children without any prior programming experience, making it adequate for young pupils, as shown in the Computational Thinking test (CTt) (Román-González et al., 2017, 2018), the CTt for Beginners (BCTt) (Zapata-Cáceres et al., 2020) and in TechCheck (Relkin et al., 2020).

Many studies have concluded that unplugged activities promote computational thinking (Hermans & Aivaloglou, 2017; Metin, 2020; Wohl et al., 2015). Relkin et al. (2021) and Brackmann et al. (2017) provided empirical evidence about the effectiveness of the unplugged approach to develop computational thinking skills, showing the improvement in students' computational thinking skills after participating in the unplugged computing instruction (Brackmann et al., 2016; Delal & Oner, 2020; Tsarava et al., 2017, 2018). proved its positive effect on motivation, engaging and consequently effectiveness in primary education (del Olmo-Muñoz et al., 2020; Saxena et al., 2020). have demonstrated that unplugged activities significantly improved the computational thinking skills of pupils in the second grade and middle school students.

Delal and Oner (2020) mentioned many other existing research that have confirmed the development of students' computational thinking skills through unplugged computing activities, at the same time simultaneously supporting them to learn computer science concepts and to improve their interests in a positive way (Bell, 2018; Cortina, 2015; Lu & Fletcher, 2009; Rodriguez et al., 2016). More specifically, Relkin et al. (2021) found that unplugged programming activities enable the highest level of understanding of algorithms, logic predictions, and debugging concepts.

To focus on the algorithm activity, we define a task where the problem setting activity and the assessment activity are straightforward. In other words, we focus on the central part of the CT-cube (see Fig. 2). Furthermore, because of the young age of the pupils, we focus only on two computational worlds, embodied and symbolic, and we exclude the formal world as it is too abstract for this age category. Finally, we consider all the three possible autonomy levels.

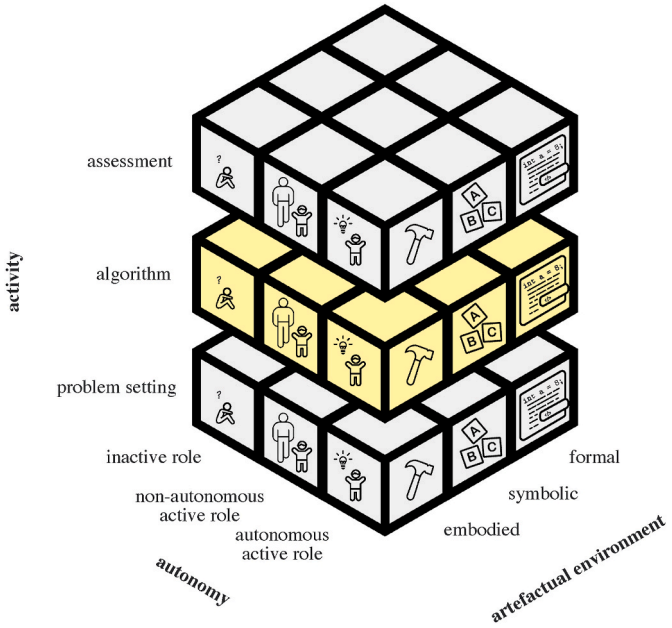


Fig. 2. The algorithm activity of the CT-cube.

We define a *cross array* as a cross shaped array consisting of five 2×2 square arrays of coloured dots, with colours selected from a set of k colours. Consider for example the cross arrays in Fig. 3. The first cross array displays a very irregular pattern, consisting of dots of four colours. The second cross array displays a very regular pattern, consisting of five equal square arrays with a regular pattern of dots of two colours. The task consists in observing a cross array and describe it to another person, that cannot look at the original array, in order to let him/her colour in the same way a blank cross array.

The experimental setting of the CAT is displayed in Fig. 4. Pupil and researcher are seated at two opposed sides of a table. In the middle of the table, there is a screen, that blocks the view of the task space but allows to see the other person. In front of the pupil, on the table, we have a coloured cross array. On the right of the pupil, there is a blank cross array, that the pupil can use as iconic artefact. In front of the researcher, there is a blank cross array, that the researcher must colour (using pencils) according to the instructions given by the pupil, and a protocol, that is compiled directly by hand by the researcher during the activity.

The initial instructions given by the researcher to the pupil are the following: *You have a coloured array in front of you. I have the same array, but uncoloured, in front of me. You should describe your array to me, so that I can colour mine the same way. You can try to describe it by voice. If it is too difficult, you can indicate the dots on the empty array on your right. If it is still too difficult, you can ask me to remove the screen, so that you can look at what I'm colouring.*

As highlighted in the beginning of this section, the CAT in

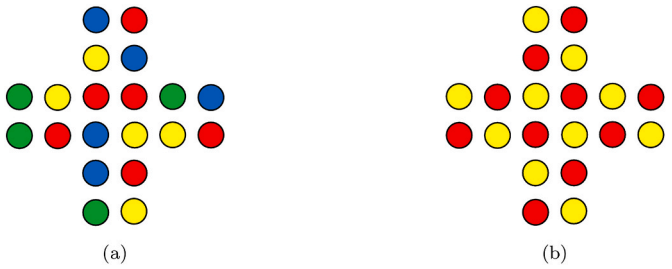


Fig. 3. Examples of cross arrays: (a) Cross array with an irregular pattern consisting of dots of four colours; (b) Cross array with a regular pattern consisting of dots of two colours.

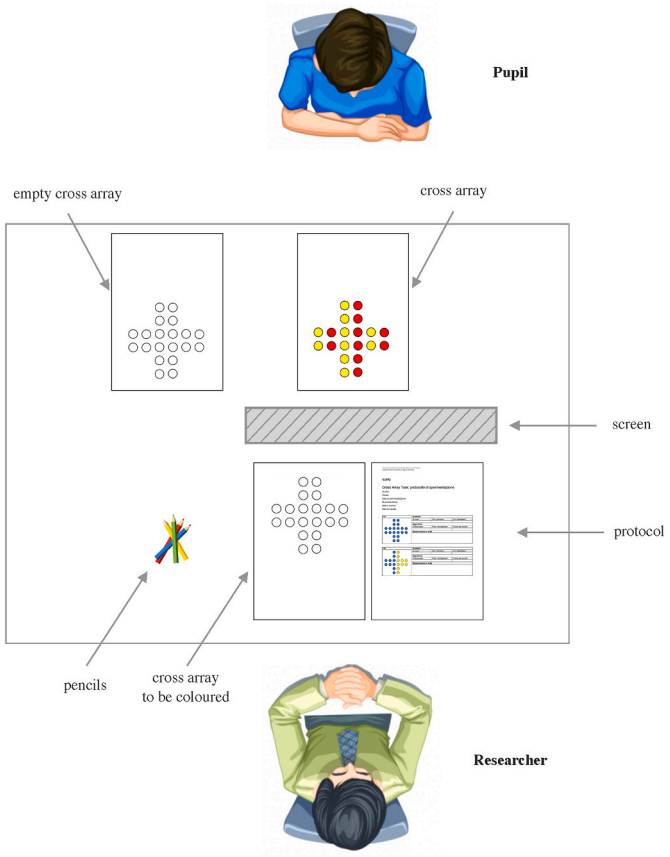


Fig. 4. The experimental setting of the CAT.

characterised by a straightforward problem setting phase (actually, during the experimentation, the task has been understood immediately also by very young pupils), and also the assessment phase is very simple, because it is sufficient to compare the final cross array coloured by the researcher with the original array. The focus is put on the algorithm phase (see Fig. 2).

The artefactual environment is characterised by two types of representational artefacts, the voice (symbolic artefact) and the arrays (iconic/embodied artefacts). The artefacts that can be used by the pupil to describe the array to the researcher are the voice and the empty array on his/her right; if the pupil describes the array only by voice, he/she is considered to act in the symbolic world; if he/she uses also the empty array, he/she is considered to act at the same time in the embodied and in the symbolic world.

The pupil is considered to be inactive if he/she does not try to solve the task and/or is unable to give intelligible instructions to the researcher; he/she is considered to have a non-autonomous role if the screen has been removed and he/she is giving intelligible instructions to the researcher, while he/she is considered to have an autonomous active role if he/she is giving intelligible instructions to the researcher with the screen.

The activity begins in the cell of the CT-cube corresponding to *algorithm - autonomous role - symbolic artefact* (see cell 1 in Fig. 5).

If the pupil is inactive or is giving false or incomplete instructions trying to solve the task only by voice (cell 5 in Fig. 5, corresponding to the cell of the CT-cube *algorithm - inactive role - symbolic artefact*), the researcher suggests to him/her to use the empty array on his/her right (cells 1 and 2 in Fig. 5, corresponding to the cells of the CT-cube *algorithm - autonomous role - symbolic artefact* and *algorithm - autonomous role - embodied artefact*). If also in this case the pupil is inactive or is giving false or incomplete instructions (cells 5 and 6 in Fig. 5, corresponding to the cells of the CT-cube *algorithm - inactive role - symbolic artefact* and

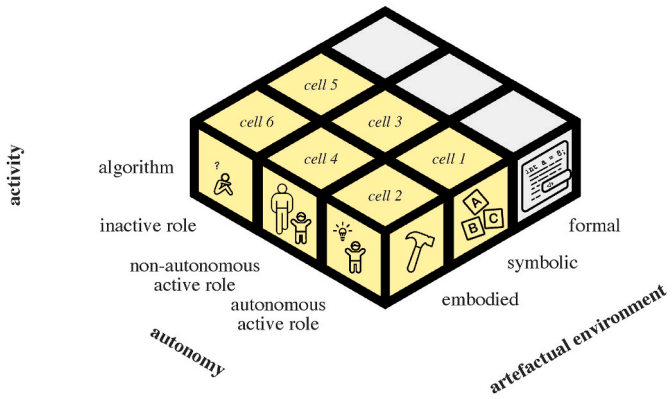


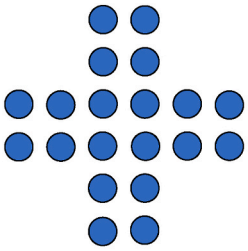
Fig. 5. The possible cells in the algorithm activity of the CT-cube.

algorithm - inactive role - embodied artefact), the researcher removes the screen and shows to the pupil what he/she is colouring. The possibility for the pupil to observe what the researcher is doing, is considered as a particular kind of (indirect) support; therefore, in this case, the role of the pupil is considered *non-autonomous*.

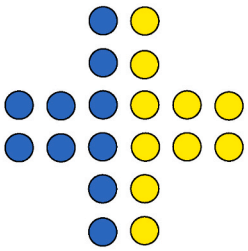
Immediately after having removed the screen, the pupil finds himself/herself in the cells of the CT-cube *algorithm - non-autonomous role - symbolic artefact* and *algorithm - non-autonomous role - embodied artefact* (see cells 3 and 4 in Fig. 5).

If also in the latter case the pupil is inactive or is giving false or incomplete instructions (cells 5 and 6 in Fig. 5), the task is finished and considered *unsuccessful*.

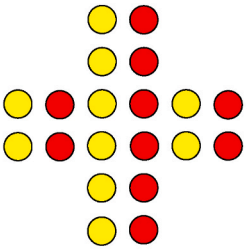
The pupil is in each case free to use the empty array on his/her right or to ask to remove the screen in each moment. The task is considered *successful*, if the pupil is able to give complete and correct instructions (eventually with corrections during the description, for example after



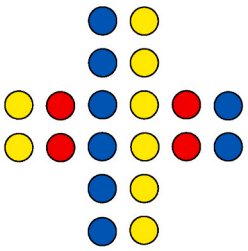
(a) S1 Monochrome scheme.



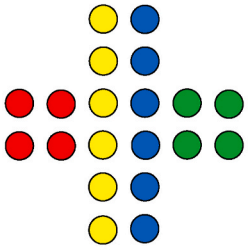
(b) S2 Two-colour scheme: two symmetric blocks with respect to the vertical axis.



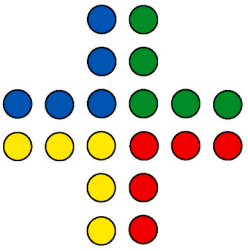
(c) S3 Two-colour scheme: alternating monochrome columns.



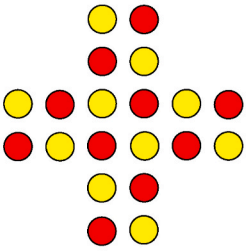
(d) S4 Three-colour scheme: alternating monochrome columns.



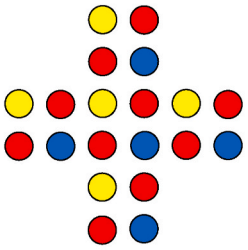
(e) S5 Four-colour scheme: two squares and two monochromatic columns.



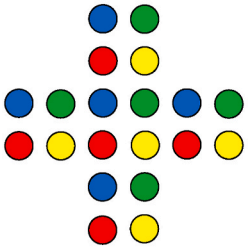
(f) S6 Four-colour scheme: double axis of symmetry.



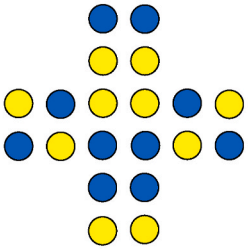
(g) S7 Scheme with a two-colour module repeated in each square.



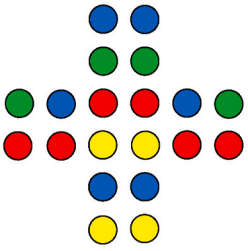
(h) S8 Scheme with a three-colour module repeated in each square.



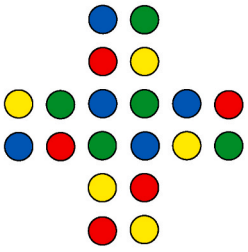
(i) S9 Scheme with a four-colour module repeated in each square.



(j) S10 Two-colour scheme with axial symmetry with respect to the vertical axis.



(k) S11 Four-colour scheme with axial symmetry with respect to the vertical axis.



(l) S12 Scheme with axial symmetry with respect to the vertical axis in four complementary colours (green-blue, yellow-red).

Fig. 6. Sequence of schemas.

having removed the screen) to the researcher, independently from the artefacts used by and the active role (non-autonomous or autonomous) of the pupil. In the rest of the paper, we call *algorithm* the entire set of correct instructions given by the pupil.

4. Classification of the algorithms

In our experimental study (see Section 4.1), we have considered the 12 cross arrays (which we refer to as S1 to S12), in Fig. 6, characterised by different levels of complexity and based on different types of regularities.

Each pupil has performed the CAT for each of the arrays and the produce algorithms have been registered by the researcher in the protocol. The algorithms used by the pupils to describe the cross arrays are classified in three categories, depending on the type of operations that have been used. We distinguish between:

1. *Zero-dimensional algorithms* (0D), in which the cross array is described point by point, colouring each single dot with a given colour (we call these operations `Colour-One-Dot`, in short COD);
2. *One-dimensional algorithms* (1D), in which the cross array is described using a sequence of operations consisting in colouring one dot with a given colour (COD) or colouring a series of dots in a given structure (row, column, diagonal, square, L, zig-zag, half-cross or entire cross) with a given colour (we call such operations Colour-Several-Dots, in short CSD), and
3. *Two-dimensional algorithms* (2D), in which one or more loops are performed on one or more CODs and/or CSDs.

A particular case of CSD is the *complement*, that consists in colouring all the dots that have not yet been described in the algorithm with a given colour. An algorithm is called *redundant*, if one or more dots are described more than once in the algorithm.

We define the *number of operations* used in an algorithm as the total number of CODs and CSDs used in the algorithm, where a COD and/or CSD used inside a loop is considered only once. The maximal number of operations in non-redundant algorithms is 20, the minimal number is 1.

Consider for example the cross array S3 in our sequence, in Fig. 7.

If the pupil describes the array point by point without redundancy, as displayed in Fig. 8a, he/she is using a zero-dimensional algorithm with 20 CODs (one for each dot), consequently the number of operations is equal to 20.

If the pupil describes the array column by column without any loop, as depicted in Fig. 8b, the algorithm consists of six CSDs (one for each column) and is one-dimensional. The number of operations corresponds to the number of CSDs (there is no COD) and is equal to 6.

Finally, consider an algorithm in which the left square is described through its two columns, and the square on the right is described as equal to the left one, while the two columns in the middle are described column by column, as displayed in Fig. 8c.

To generate the right square, it is sufficient to repeat the operations performed to generate the left square, and could therefore be described in a pseudo-code using a loop, as shown in Fig. 9.

In this case the algorithm contains a loop on two CSD as can be seen

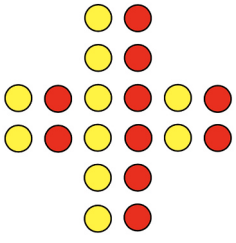


Fig. 7. Schema S3.

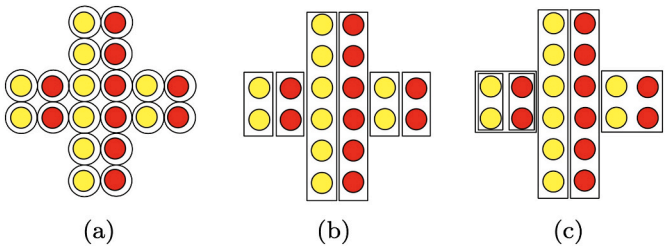


Fig. 8. Examples of algorithms for Schema 3: (a) S3 algorithm 1; (b) S3 algorithm 3; (c) S3 algorithm 5.

```
1:  // One cycle for each square, e.g., i=1 left square, i=2 right square
2:  for i ← 1, 2 do
3:      // Colour the left column of the square in yellow
4:      CSD(column, yellow)
5:      // Colour the right column of the square in red
6:      CSD(column, red)
7:  end for
8:  // Colour the left central column in yellow
9:  CSD(column, yellow)
10: // Colour the right central column in red
11: CSD(column, red)
```

Fig. 9. Schema 3 algorithm 5.

on line 2, therefore it is a two-dimensional algorithm. In this case, the number of operations is 4, corresponding to the number of CSDs, respectively on lines 4, 6, 9 and 11 (CSDs inside a loop are considered only once).

All the twenty cross arrays used in our experimental study could be described using zero-, one- or two-dimensional algorithms, with the exception of the last schema (S12), that could be described only using zero- or one-dimensional algorithms.

In the CAT, pupils are assessed considering, for each cross array, if the task has been successful:

- the *type of algorithm*: the higher the dimension, the better;
- the *artefacts used*: the exclusive use of the voice is considered to be more difficult and therefore valuable than the simultaneous use of voice and empty array;
- the *level of autonomy*: an autonomous active role is more valuable than a non-autonomous one.

We therefore consider that the most skilled pupils are the ones that are able to conceive autonomously, for each cross array, a non-redundant algorithm with the highest possible dimension (usually two dimensional-algorithms) and is able to describe them exclusively by voice. Because we are considering multiple assessment criteria, it is not always possible to rank pupils according to their performance; for example, it is not possible to say if it better a two-dimensional algorithm described using voice and empty array, or a one-dimensional algorithm described exclusively by voice. However, the dimensions observed are suitable to give a deep insight into the algorithmic skills of the pupils, as shown in the next section. We do not consider the number of operations as criterion for the assessment in this paper, because there is a clear correlation between the number of operations and the algorithm dimension, and we consider the latter criterion as more informative about the complexity of the algorithm conceived and described by the pupil.

4.1. Participants and data

The CAT was tested in an experimental study in eight classes from three public schools in Switzerland between March and April 2021.

In Southern Switzerland, compulsory schools are organised in three school orders: (1) two years of compulsory kindergarten, for pupils starting from the age of 4–6, with one optional year starting at age 3, (2) five years of primary school (elementary) for pupils aged 6 to 11, and (3) four years of low secondary school (middle school), for pupils aged 11 to 15. The language of instruction is Italian. In kindergarten, pupils of all ages are usually regrouped in mixed classes, while in primary school pupils are in classes corresponding to the different years of primary school. Exceptionally, in case of schools with a small number of pupils, they can be regrouped in classes covering several years. In the lower secondary school, pupils are regrouped in classes corresponding to the different school years with no exception. In our study, to cover the entire compulsory school, we have selected, in the same school district, one pre-school class (pupils aged 3 to 6), three primary school classes — a first, a third and a fifth (pupils aged 6 to 11) — and four low secondary school classes — a second, a third and two fourth (pupils aged 12 to 16). In each class, we have performed the CAT with all the pupils that were prone to participate and that were been explicitly allowed by their parents to take part to our study through a signed disclaimer.

A total of 109 pupils (51 girls and 58 boys) from all levels of compulsory school, i.e. between 3 and 16 years old, participated in the experiment (see Table 1) with the explicit permission of their parents and of the school authorities.

Because of the young age and consequent vulnerability of the involved participants, particular attention has been devoted to ethical integrity of the applied experimental protocol and to a transparent communication with pupils, parents, and schools to promote a conscious participation (Petousi & Sifaki, 2020).

In this context, national and international research ethics guidelines were followed (Aebi-Müller et al., 2021). In particular, the parents of all the pupils of the selected classes have received a document with a description of the research project and details about the involved institutions, the researchers, and the experimental activity to be performed in class. This document also contains a disclaimer with details about the type of data collected — research data was anonymous and did not monitor sensitive information —, the storage method — on protected servers —, the use of the collected data and the persons authorised to access them — only the researchers directly involved in the study. For further information, the contact details (mobile phone number and e-mail) of the principal investigator of the research have been made available. Parents could sign and return the disclaimer to authorise their children to participate in the experimentation. The school directors and the teachers have received a description of the research project and of the experimental protocol, together with the template of the disclaimer. They have been asked to provide explicit authorisation to access the school, respectively the classes, for the experimentation.

In each session, all the students present who were authorised by their parents to take part in the study were able to voluntarily decide whether to participate in the activity or not. Two pupils at a time were randomly selected from the class and taken to a separated room to avoid interference with the students remaining in the classroom.

Each pupil was randomly assigned to one of the available researchers who introduced the task, the goal, explained the rules (see detailed protocol in Section 3.1) and wrote down the strategy employed by the pupils for each schema of the CAT on paper. No time limit was imposed to complete the CAT. The strategies employed by the pupils were later transferred to a database which also included relevant demographic information (school, class, gender, age, ID number).

The time required to solve all the 12 schemas varied from a minimum of 10 min — in case of the older pupils — to a maximum time of 45 min — for the younger. Approximately, the total time required to administrate the CAT to all 109 has been 36 h.

As performance in the CAT is multi-factorial, we analyse for each schema of the CAT and each pupil the autonomy level, the artefacts used and the dimension of the algorithm used to describe the schema. These results are then put in relation with their age group and gender. The distinction according to age is essential to evaluate the evolution of student's algorithmic capabilities and assess the developmental adaptability of the CAT. Four age categories were established (see Table 1). The first category includes pupils aged 3 to 6 (referred to as pre-primary school pupils in the rest of the paper). The second category contains pupils from 7 to 9 years old (referred to as lower primary school pupils in the rest of the paper), corresponding to classes from first to third elementary school in Southern Switzerland. The third category is composed of kids aged 10 to 13, corresponding to classes from fifth primary school to second class of lower secondary school in Southern Switzerland (referred to as upper primary school pupils in the rest of the paper). Finally, the last category are pupils from 14 to 16 year old, who attend the secondary school classes from third to fourth (that in Southern Switzerland is the last class of the lower secondary school. The four categories encompass thus the complete compulsory school in Switzerland, from Kindergarten to lower secondary school.

Table 1
Number of pupils per age category.

Age categories	Age range	Female	Male	Total
Kindergarten	3–6 years old ($\mu = 5.1 \pm 0.6$)	6	8	14
I-III primary	7–9 years old ($\mu = 7.9 \pm 1.2$)	13	19	32
V primary-II secondary	10–13 years old ($\mu = 11.4 \pm 0.8$)	12	15	27
III-IV secondary	14–16 years old ($\mu = 14.3 \pm 0.9$)	19	15	34
Total		51	58	109

5. Results and discussion

The analysis is structured in two parts. We first present the analysis of algorithms proposed by the pupils (Section 5.1) before conducting a detailed performance analysis according to the CT-cube dimensions, considering the effect of both age and gender (Section 5.2).

5.1. Algorithm analysis

Table 2 summarises, for each schema of the CAT, the number of pupils who accomplished it successfully, the number of different algorithms observed and the distribution of algorithms according to their dimension.

All pupils accomplished the CAT successfully (according to the criteria defined in Section 3.1) for each schema they were confronted with. Note that while pupils managed to solve all schemes until S7, starting from S8 there are 5 cases where the experiment was interrupted due to time constraints (e.g., end of school hours, attention span for very young kindergarten pupils). This result implies that the task is suitable for pupils from kindergarten to secondary school, unlike most formal computational thinking assessments which are limited to a pre-defined age range. Indeed, the majority of studies focused on evaluating computational thinking skills for a specific educational level, more often elementary or middle school. For example, the Computational Thinking Test (CTt) by Román-González et al. (2017) is designed for 7th and 8th grade students (12–14 years old), although it can also be used to measure the computational thinking from 5th to 10th grade (10–16 years old). The Beginners Computational Thinking Test (BCTt) by Zapata-Cáceres et al. (2020), that can be considered as an extension of the previous work, is mainly aimed at first educational stages of primary school from the 1st to 4th grade (5–10 years old) and less adapted for middle school and beyond. Appropriate assessments of computational thinking for younger children are generally more challenging and

Table 2
Number of algorithms observed and corresponding distribution among dimensions. For the latter, cases in which the percentage is above 10% are highlighted in bold.

Schema	Num. pupils who solved the schema	Num. of different algorithms	0D (%)	1D (%)	2D (%)
1	109	5	1.83%	**94.50%**	3.67%
2	109	7	2.75%	**94.50%**	2.75%
3	109	10	1.83%	**84.40%**	**13.76%**
4	109	8	1.83%	**88.99%**	9.17%
5	109	5	0.00%	**100.00%**	0.00%
6	109	7	4.59%	**95.41%**	0.00%
7	109	19	**50.46%**	**18.35%**	**31.19%**
8	107	21	**51.40%**	**12.15%**	**36.45%**
9	105	9	**52.38%**	0.00%	**47.62%**
10	105	25	**17.14%**	**58.10%**	**24.76%**
11	105	15	**14.29%**	**60.95%**	**24.76%**
12	104	6	**79.81%**	**20.19%**	0.00%

require considering developmental appropriateness in both format and content (Relkin and BersTechCheck-, 2021; Relkin et al., 2020). Being an unplugged assessment, as shown in Section 3.1, the CAT does not require prior programming experience either, making it adequate for young pupils, as the Computational Thinking test (CTt) (Román-González et al., 2017, 2018), the CTt for Beginners (BCTt) (Zapata-Cáceres et al., 2020) and TechCheck (Relkin et al., 2020). Moreover, it is important to avoid confusing coding skills with computational thinking skills.

Overall, we have observed 137 different algorithms, distributed across the 12 schemas of the CAT (see Appendix A. for the visualization of the algorithms). For most schemas there is a clear prevalence of a certain algorithm dimension. In particular, from S1 to S6 and for S10 and S11, one-dimensional algorithms are largely preferred. From S7 to S9, slightly more than half of the pupils choose zero-dimensional algorithms followed by two-dimensional algorithms with a similar percentage. Finally, for S12 the majority of the pupils opted for a zero-dimensional algorithm. There are a few schemas (S5, S6 and S12) in which two-dimensional algorithms are never chosen, and one case (S9) in which one-dimensional algorithms are never applied. One important

Table 3
Frequency of algorithm usage per schema. Algorithm 1 refers to the zero-dimensional algorithm, with following algorithms referring to other solutions employed by students without any particular ranking. The cell colour refers to the algorithm dimension (white for zero-dimensional algorithms, blue for one-dimensional algorithms and yellow for two-dimensional algorithms). The most used algorithm for each schema is highlighted in bold.

Algorithms \ Schemas	1	2	3	4	5	6	7	8	9	10	11	12
1	2%	3%	2%	2%	0%	5%	**50%**	**51%**	**52%**	18%	14%	**78%**
2	**94%**	**83%**	13%	9%	**91%**	**67%**	11%	4%	11%	4%	1%	2%
3	2%	3%	**76%**	**82%**	5%	18%	3%	4%	2%	4%	5%	12%
4	2%	9%	1%	2%	3%	1%	2%	2%	2%	1%	6%	1%
5	1%	1%	1%	1%	2%	6%	2%	1%	3%	2%	9%	7%
6	–	1%	2%	2%	–	1%	1%	4%	7%	4%	2%	1%
7	–	1%	2%	2%	–	3%	1%	2%	15%	19%	1%	–
8	–	–	2%	1%	–	–	6%	4%	7%	2%	4%	–
9	–	–	1%	–	–	–	1%	1%	1%	2%	4%	–
10	–	–	1%	–	–	–	2%	6%	–	2%	1%	–
11	–	–	–	–	–	–	10%	6%	–	2%	**43%**	–
12	–	–	–	–	–	–	4%	4%	–	1%	2%	–
13	–	–	–	–	–	–	1%	2%	–	2%	1%	–
14	–	–	–	–	–	–	2%	1%	–	2%	1%	–
15	–	–	–	–	–	–	1%	1%	–	1%	8%	–
16	–	–	–	–	–	–	2%	3%	–	1%	–	–
17	–	–	–	–	–	–	1%	3%	–	1%	–	–
18	–	–	–	–	–	–	1%	1%	–	**20%**	–	–
19	–	–	–	–	–	–	1%	1%	–	1%	–	–
20	–	–	–	–	–	–	–	1%	–	9%	–	–
21	–	–	–	–	–	–	–	1%	–	1%	–	–
22	–	–	–	–	–	–	–	–	–	1%	–	–
23	–	–	–	–	–	–	–	–	–	1%	–	–
24	–	–	–	–	–	–	–	–	–	1%	–	–
25	–	–	–	–	–	–	–	–	–	1%	–	–

remark concerns S5 as the point-by-point algorithm (zero-dimensional) is never used, showing that the pupils preferred a more efficient algorithm in terms of algorithm dimension. Table 3 shows the usage frequency of the different algorithms. For each schema, we have one to three prevalent algorithms. As aforementioned, the most used algorithms are the one-dimensional algorithms followed by the zero-dimensional ones, although never the two-dimensional alternatives.

5.2. Pupils' performance according to the CT-cube

The main results with respect to the algorithmic skills of K-12 pupils of our investigation are the following:

1. The algorithmic skills of K-12 pupils increase with the age, mainly with respect to the autonomy and the capability of using symbolic artefacts to describe their algorithms. The most important increase is observed between lower and upper primary school pupils.
2. Very young learners, i.e., pre-primary and lower primary school pupils, are already able to conceive and describe complex algorithms (one- and two-dimensional algorithms).
3. There is no significant difference between genders with respect to algorithmic skills of K-12 pupils.

To explain how our data support these claims, we should firstly define what we mean with pupil's performance in the CAT. As described in Section 3.1, the most skilled pupils in the CAT are the ones who are able to conceive autonomously, for each cross array, an algorithm of the highest possible dimension (two dimensional algorithm, with the exception of S12 which can be described at best with a one-dimensional one) and is able to describe it exclusively by voice.

To establish how pupil performance evolves, we define a single metric to characterise performance in the CAT, referred to as the *CAT-score*. Given a pupil and a schema of the CAT, we assign a score between 0 and 2 to the performance with respect to the autonomy and the use of the artefacts of the pupil and a score between 0 and 2 for the performance with respect to the dimension of the algorithm conceived and described by the pupil. In particular, with respect to autonomy and use of artefacts (in the following referred to as *CT-cube dimensions*), the scores are assigned as follows: 0 corresponds to the strategy of using voice, the empty array and visual feedback (corresponding to cells 3 and 4 in Fig. 5, we denote this combination of CT-cube dimensions in the following figures with VSF), 1 to using voice and the empty array (corresponding to cells 1 and 2 in Fig. 5, we denote this combination of CT-cube dimensions in the following figures with VS), and 2 to using voice alone (corresponding to cell 1, we denote this combination of CT-cube dimensions in the following figures with V). With respect to the type of algorithm conceived and described by the pupil, we assign the score as follows: 0 corresponds to a zero-dimensional algorithm, 1 to a one-dimensional algorithm, and 2 to a two-dimensional algorithm. The CAT-score is defined as the sum of the two scores determined above, and goes from 0 (minimum) to 4 (maximum) for each schema solved per pupil.

As explained in Section 3.1, because of the multiple assessment criteria, it is not always possible to rank pupils according to their performance in a given schema. For example, it is not possible to say if is better to have a two-dimensional algorithm described using voice and empty array, or a one-dimensional algorithm described exclusively by voice. The CAT score is defined to assign the same score to strategies that cannot be ordered. For instance, the two approaches described in the example above both have a CAT-score equal to 3. Let us now describe in detail how the data support the claims stated at the beginning of this section.

The algorithmic skills of K-12 pupils increase with the age, mainly with respect to the autonomy and the capability of using symbolic artefacts to describe their algorithms. The most important increase is observed between lower and upper primary school pupils.

The distribution of CAT scores obtained can be seen in Fig. 10 and

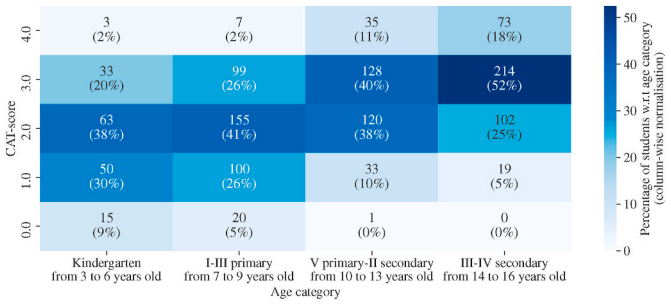


Fig. 10. Number of schemas solved with a given score w.r.t the age category. Percentages are provided as the proportion of responses w.r.t the total number obtained per category (column-wise normalisation), to facilitate the comparison between age groups.

appears to increase with age. This trend is consistent with prior works. For example, Dietz et al. (2019) show how children become more successful and more efficient at completing the task with age, according to several measures, confirming the relationship between age and success rate. Klahr and Robinson (1981) examine the problem-solving abilities and planning processes in preschool children, observing better performance among older children.

When considering all schemas, the χ^2 statistic indicates significant differences according to age ($\chi^2(12) = 270.2$, $p < .0001$).

The pairwise comparisons shown in Table 4 indicate that these differences are significant between all age categories, apart from the two youngest (i.e., Kindergarten and I-III primary). This leap that occurs after the two age groups is attributable to the fact that, the Switzerland school system is organised in cycles (Cantone, 2015) — the curriculum of each cycle is characterised by different educational objectives —, and one of the transitions that take place between one cycle and another, occurs precisely after these two youngest age categories.

Fig. 11 shows the strategy employed by the pupils to solve the CAT according to the CT-cube dimensions, with respect to the age categories. We observe that one-dimensional algorithms are the most used for each age category, and the percentage of two-dimensional algorithms increases with age. In other words, in all the age categories, the pupils are able to produce complex algorithms, and the algorithmic skills grow with age.

Furthermore, we observe that the increase in the CAT-score is mainly due to an improvement in the capability of using only the voice (symbolic artefact) and/or to the autonomy, rather than the algorithm dimension, although also the latter tends to increase with the age, notably between the second and the third age categories. More specifically, the most used combination of CT-cube dimensions in the first two age categories (ages 3–9) is the voice with empty array, while it is voice alone for the last two age categories (ages 10–16). It thus appears that older pupils are able to deal with more complex situations without the support of external artefacts. It is also interesting to remark that pupils aged 3–6 ask to remove the screen (non-autonomous role) in 17% of cases, while this falls to 0% for pupils aged 14–16.

The improvement in CT-cube dimensions is particularly evident for

Table 4
Pairwise comparison between age groups for the scores obtained over all schemas with Benjamani-Hochberg p-value correction for false detection rates (*pri.* stays for primary while *sec.* stays for secondary).

	Kindergarten	I-III pri.	V pri. - II sec.
I-III pr.	$\chi^2(4) = 5$, $p = .26$ (ns)		
V pr.-II sc.	$\chi^2(4) = 75$, $p < .0001$ (****)	$\chi^2(4) = 73$, $p < .0001$ (****)	
III-IV sc.	$\chi^2(4) = 160$, $p < .0001$ (****)	$\chi^2(4) = 182$, $p < .0001$ (****)	$\chi^2(4) = 30$, $p < .0001$ (****)

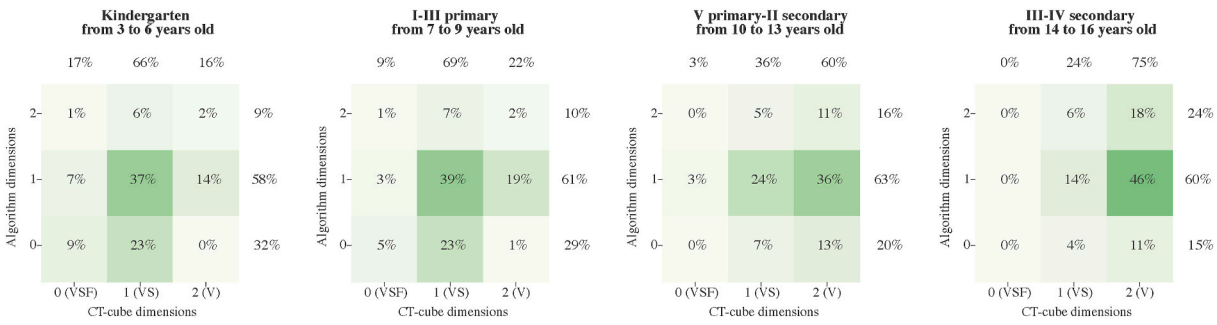


Fig. 11. Overall performance across age categories. The CT-cube dimensions are characterised in the case of the CAT by the three ways pupils can solve a schema: i) only by voice (V), ii) using voice and the empty array (VS), or iii) using voice, the empty array and the visual feedback (VSF).

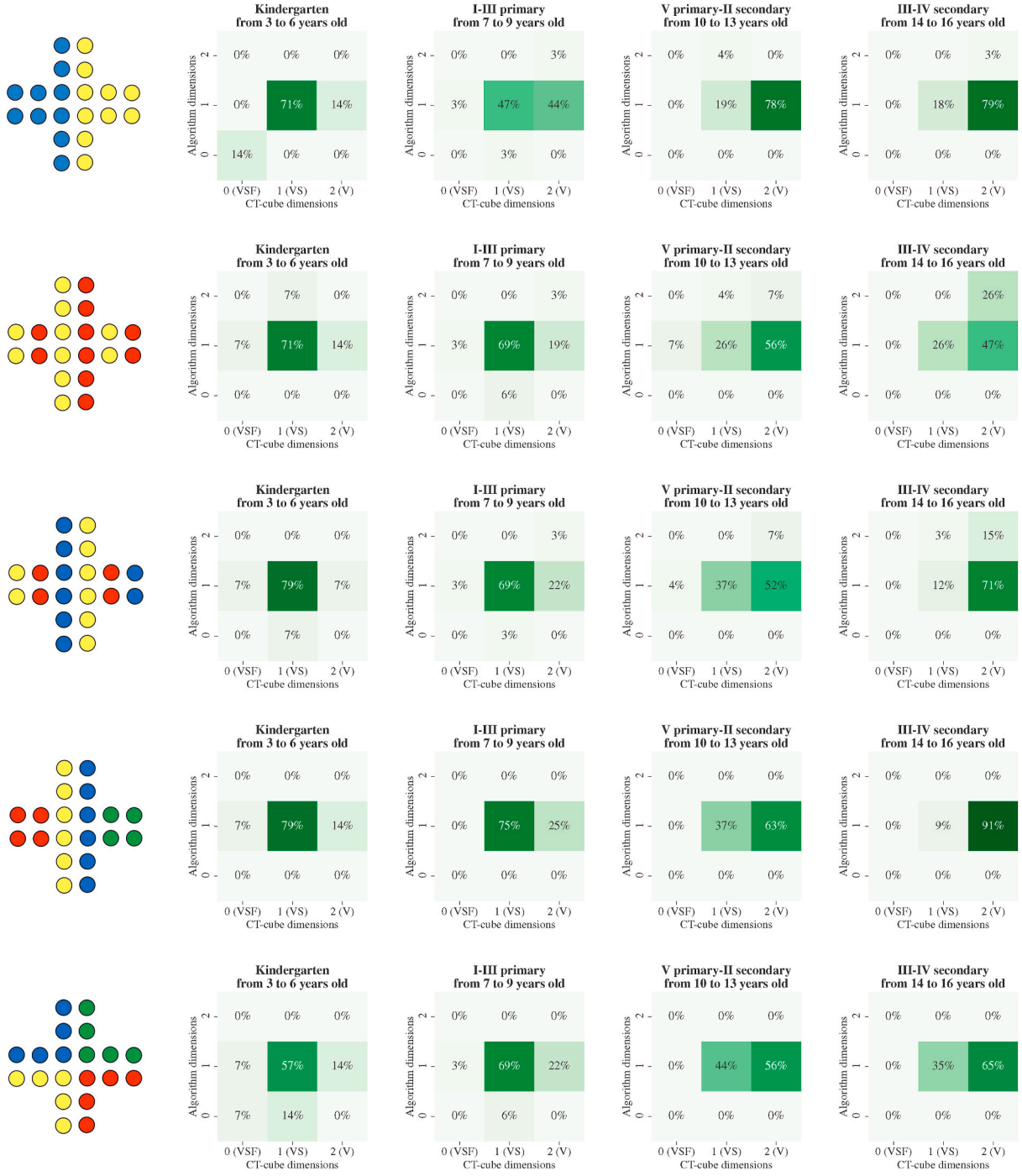


Fig. 12. Performance analysis across age categories of S2 to S6.

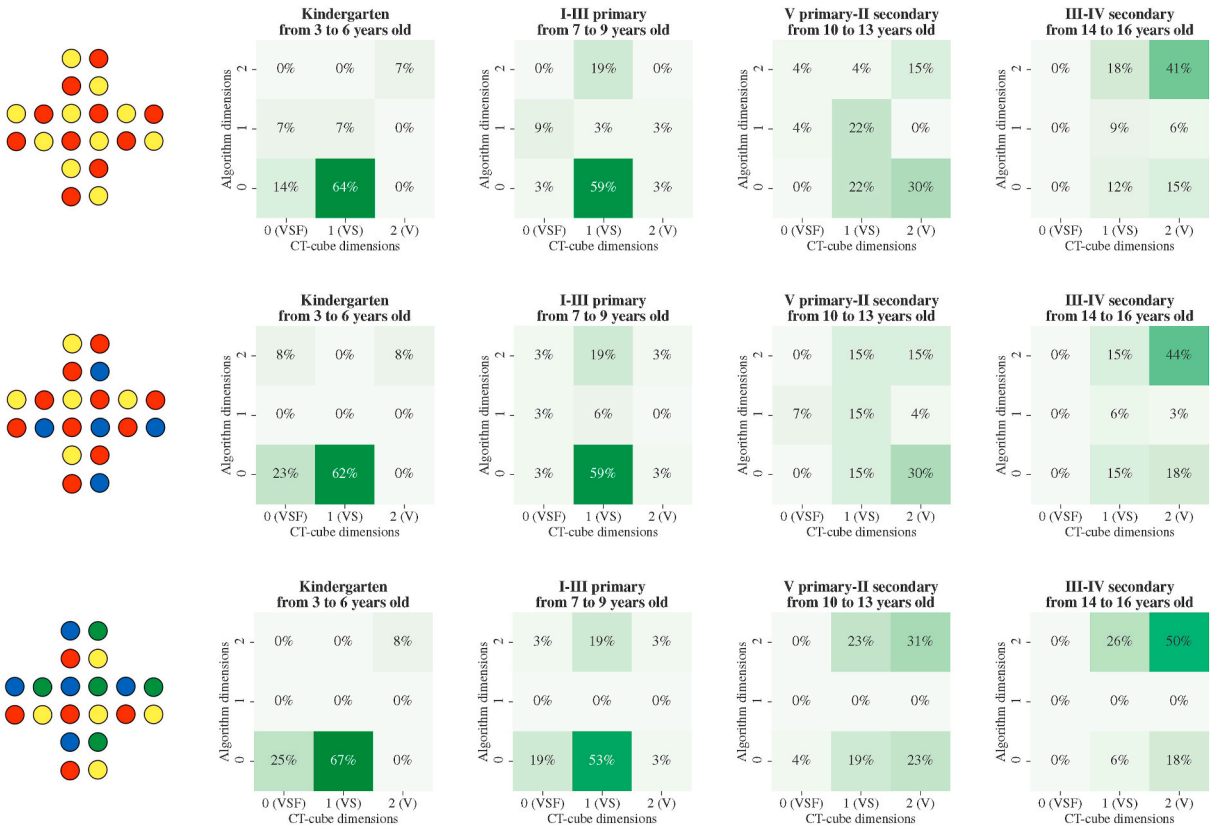


Fig. 13. Performance analysis across age categories of S7, S8 and S9.

example for schemas S2 to S6 (see Fig. 12). In Appendix B. is shown the performance analysis across age categories of all the schemas (see Figs. from B.30 to B.40).

The improvement in the complexity of the algorithms conceived and described by the pupils, on the other hand, is visible for example for schemas S7–S9 which have been generated according to the repetition of a square pattern (see Fig. 13).

Very young pupils, i.e., pre-primary and lower primary pupils, are already able to conceive and describe complex algorithms (one- and two-dimensional algorithms).

Pre-primary school pupils (aged 3–6) and lower primary pupils (aged 7–9) are able to conceive in many case one- and two-dimensional algorithms, and in some cases to describe them only by voice, showing good algorithmic skills already at very young age. This is particularly evident for example in the case of S10 where 43% of kindergarten pupils employed two-dimensional algorithms using the empty array as an artefact (see Fig. 14). These results show that kindergarten pupils already exhibit algorithmic skills, and that these may be elicited using suitable artefacts, in particular embodied/iconic.This evidence is a further support to the fact that it is possible to work on computational thinking skills already with very young pupils. Prior works have also shown that even in preschool-aged children, computational thinking skills can emerge and rapidly develop. Wohl et al. (2015) achieved satisfactory learning outcomes with students under the age of five, coming up with the conclusion that unplugged activities can be used effectively to introduce successfully computational concepts. Dietz et al. (2019, pp. 1647–1653) explore young children's developing capacities for problem decomposition and demonstrate that the skills necessary for this type of problem may be initiated to be fostered in preschool years.

There is no significant difference between genders with respect to algorithmic skills of K-12 pupils.

An analysis of the CAT-score according to gender and age categories (see distribution in Fig. 15) does not show significant differences between genders (χ^2 test of independence between genders for each age category $p > .05$). To determine whether this holds true for both the algorithm and CT-cube dimensions we look into more detail at the strategies employed by the pupils of the different age categories with respect to gender.

Fig. 16 confirms that the observations made for the full sample does not differ according to gender.

As reported by Tikva and Tambouris (2021), gender differences have been examined in many studies, often leading to contradictory results.

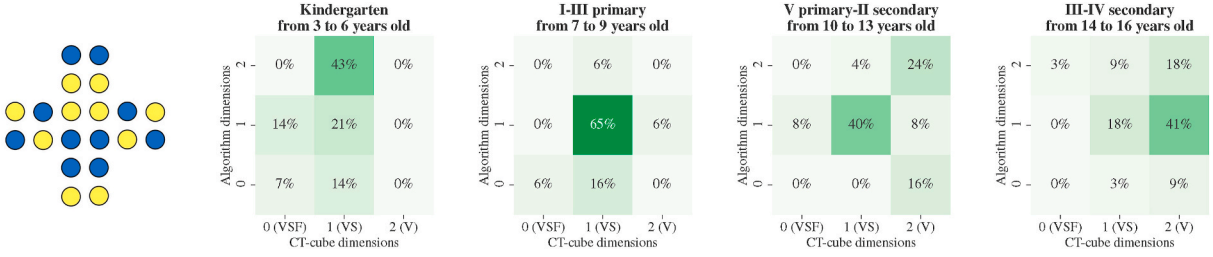


Fig. 14. Performance analysis across age categories of S10.

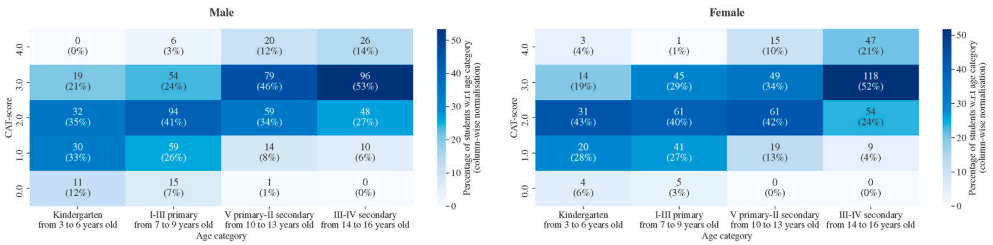


Fig. 15. Number of schemas solved with a given score w.r.t gender and age category. Percentages are provided as the proportion of responses w.r.t the total number obtained per category (column-wise normalisation), to facilitate the comparison between age groups.

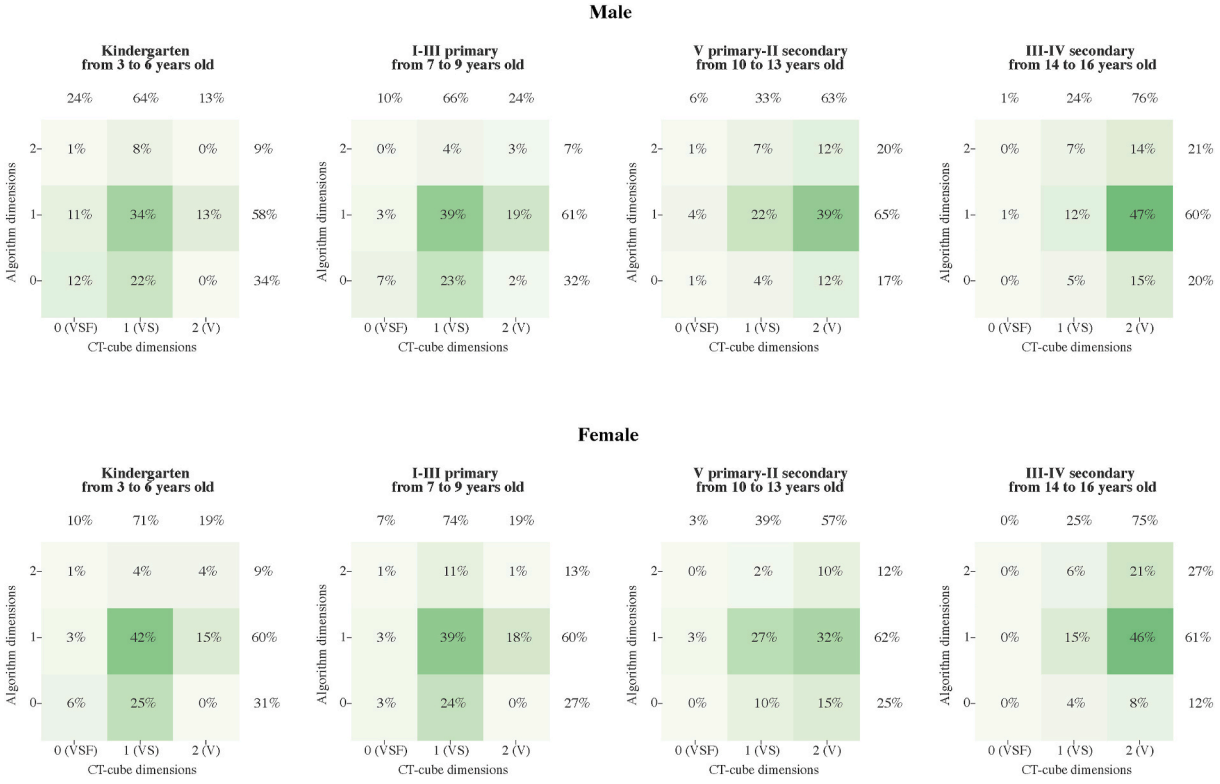


Fig. 16. Performance analysis across age categories and gender.

For example Román-González et al. (2017) reported a relationship between the role of gender and the development of computational thinking. Other studies, such as Relkin et al. (2020), Delal and Oner (2020), Atmatzidou and Demetriadis (2016) and Metin (2020), verified gender relationship in their tests and concluded that no significant difference between genders appears when evaluating the students.

6. Conclusion and further research

In this paper, we have proposed a framework, called CT-cube, allowing to extend existing computational thinking models to encompass the developmental and situated nature of computational thinking skills. To evaluate the suitability of the proposed framework, we have applied the CT-cube framework for the design and assessment of an unplugged computational thinking activity, the CAT, allowing to assess algorithmic skills of compulsory school pupils. The assessment performed in this paper shows that the CT-cube can actually be effectively and easily applied for the assessment of computational thinking skills in the context of classroom activities. This result delineates new possibilities for the assessment of computational thinking skills.

In the particular case of the CAT, the framework has allowed to produce an assessment of the algorithmic skills of a sample of 109 compulsory school pupils. The collected data show that (i) the algorithmic skills of the pupils increase with the age, in particular with respect to the autonomy and the capability of describing algorithms with symbolic artefacts, (ii) pre-primary and lower primary school pupils are already able to conceive and describe complex algorithms, and (iii) there is no significant difference between genders with respect to algorithmic skills.

Further research is needed to reinforce or confirm the results described in this paper and to validate in general the CT-cube as possible framework for the design and the assessment of computational thinking

activities. The CAT, an unplugged activity focusing only on a limited subset of the cells of the CT-cube and on a specific component of computational thinking (algorithms), should be considered as a first prototype of activity designed according to the CT-cube. Despite its suitability for the assessment of algorithmic skills in the compulsory schools, supported by the results obtained in this study, further research is needed to assess the effectiveness of the CT-cube as general framework for the design of more complex computational thinking activities, involving other components and definitions of computational thinking, other types of activities (for example coding and educational robotics) and exploring more extended regions or even the entirety of the CT-cube. Furthermore, the results of this paper should be interpreted in the light of the data collection that has been performed, that has been restricted to a relatively limited number of pupils in a single region of Southern Switzerland; further investigations on larger samples and covering several school systems would be necessary to confirm the development of algorithmic skills observed in this study.

From the latter point of view, the CAT has been designed as a bilateral (pupil-teacher or pupil-researcher), quite time consuming, assessment activity and in its actual form is not suitable for the realisation by one single teacher with an entire class at the same time. In particular, the data collection in the pre-primary school class has required indicatively the involvement of two researchers for an entire school day. The data collection in the primary school has required the same number of researchers, but for only half a school day. Finally, in the lower secondary school, depending on the size of the class, the data collection has required approximately two researchers during two teaching units of 45 min. To extend the activity to a larger number of pupils and to different regions, it would be useful to conceive and release an online version of the CAT to be performed on a tablet, substituting the researcher with an intelligent tutoring and assessment system.

To extend the activity to a larger number of pupils and to different regions, it would be useful to conceive and release an online version of the CAT to be performed on a tablet, substituting the researcher with an intelligent tutoring and assessment system.

In particular,

1. The CAT takes into account only one type of activity (algorithm description) and only a subset of the cells of the CT-cube. Further research is needed to assess the effectiveness of the CT-cube as framework for the design of more complex computational thinking activities, involving several types of activities and a larger number or all the cells of the CT-cube;
2. The data collection has been restricted to a limited number of pupils in a single region of Switzerland; further investigations on larger samples and covering several school systems would be necessary to confirm the particular type of development of algorithmic skills observed in this study.

From the latter point of view, it would be useful to conceive semi-automatic or automatic assessment procedures (for example through an intelligent tutoring system) allowing large-scale assessments, that would be very difficult to realise with the quite time-expensive procedure applied here.

CRediT authorship contribution statement

Alberto Piatti: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition. **Giorgia Adorni:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Laila El-Hamamsy:** Validation, Formal analysis, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Lucio Negrini:** Validation, Formal analysis, Investigation, Writing – review & editing. **Dorit Assaf:** Validation, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Luca Gambardella:** Conceptualization, Validation, Supervision, Project administration, Funding acquisition. **Francesco Mondada:** Validation, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

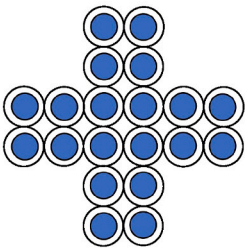
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

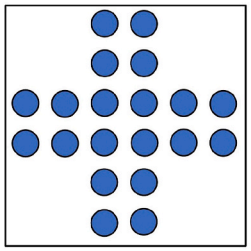
This research was funded by the Swiss National Science Foundation (SNSF) under the National Research Program 77 (NRP-77) Digital Transformation (project number 407 740_187246) and the National Centre of Competence in Research (NCCR) Robotics. The authors are very grateful to the teachers and pupils that have participated to the study. A special thanks to Martina Piatti for the first drawings of the cross arrays.

Appendix A. Algorithms classification

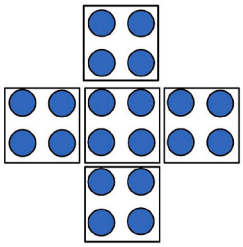
For each schema, the first algorithm is always the zero-dimensional algorithm which describes the array point-by-point, while all the others are assigned randomly and without necessarily having something in common.



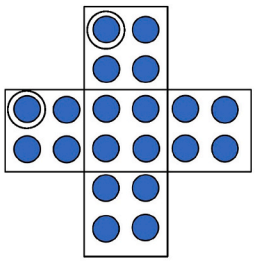
(a) Schema 1 Algorithm 1: Point by point



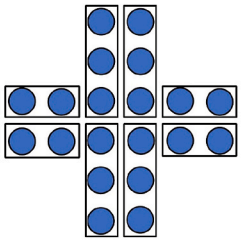
(b) Schema 1 Algorithm 2: All blue



(c) Schema 1 Algorithm 3: Five squares all blue

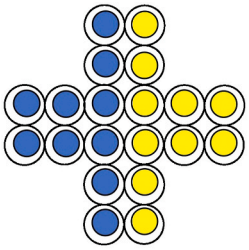


(d) Schema 1 Algorithm 4: Two columns and two rows

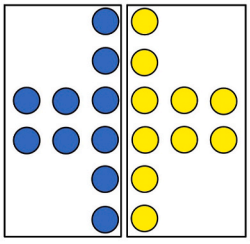


(e) Schema 1 Algorithm 5: Four L split in two

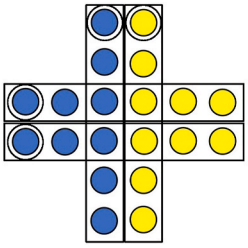
Fig. A.17. Algorithms that have been observed at least once for schema S1.



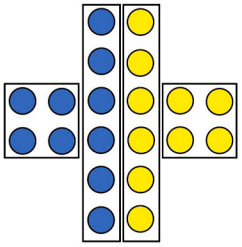
(a) Schema 2 Algorithm 1: Point by point



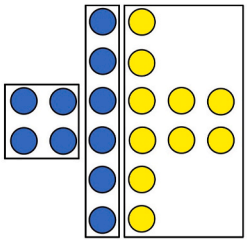
(b) Schema 2 Algorithm 2: Split vertically, blue on the left yellow on the right



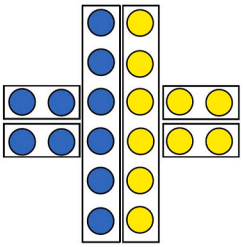
(c) Schema 2 Algorithm 3: Two columns and two rows



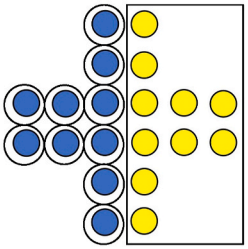
(d) Schema 2 Algorithm 4: Two columns and two squares on the sides



(e) Schema 2 Algorithm 5: Half of one colour, one square and one column with the other

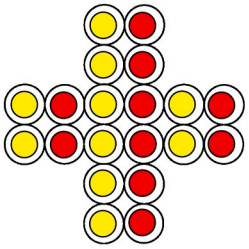


(f) Schema 2 Algorithm 6: Two columns and the sides row by row

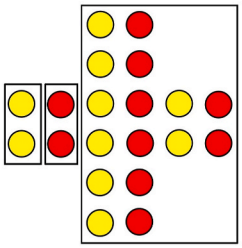


(g) Schema 2 Algorithm 7: Half point-by-point, the remaining of one colour

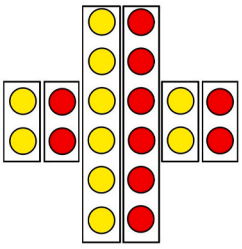
Fig. A.18. Algorithms that have been observed at least once for schema S2.



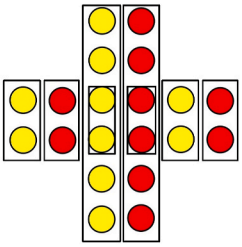
(a) Schema 3 Algorithm 1: Point by point



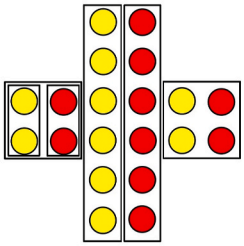
(b) Schema 3 Algorithm 2: Alternate columns



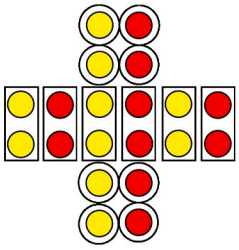
(c) Schema 3 Algorithm 3: Column by column



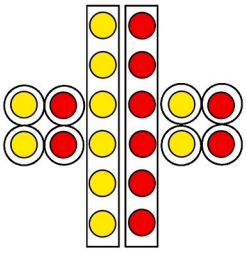
(d) Schema 3 Algorithm 4: Column by column (with redundancy)



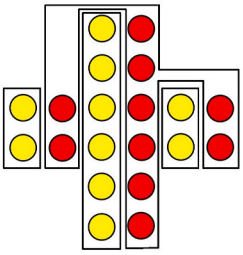
(e) Schema 3 Algorithm 5: Column by column symmetrical sides



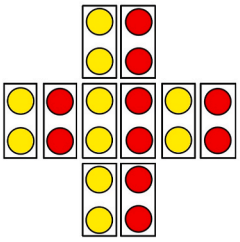
(f) Schema 3 Algorithm 6: Two rows column by column, point-by-point up and down



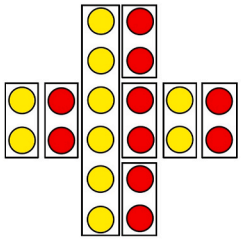
(g) Schema 3 Algorithm 7: Two columns, squares point-by-point



(h) Schema 3 Algorithm 8: One colour column by column, the remaining of the other colour

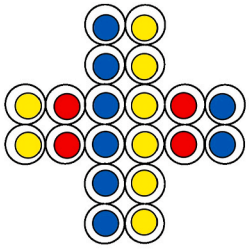


(i) Schema 3 Algorithm 9: Two rows column by column, the remaining column by column

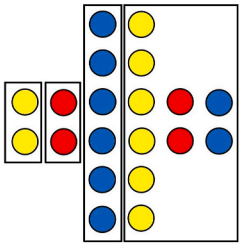


(j) Schema 3 Algorithm 10: One column with pairs of two, the remaining column by column

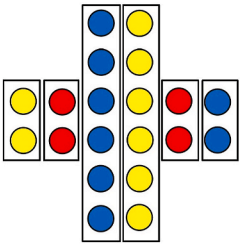
Fig. A.19. Algorithms that have been observed at least once for schema S3.



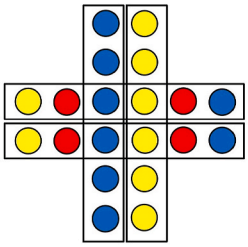
(a) Schema 4 Algorithm 1: Point by point



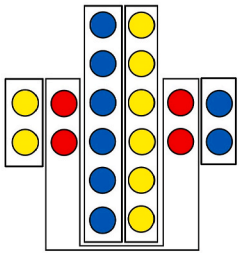
(b) Schema 4 Algorithm 2: Repeat three columns



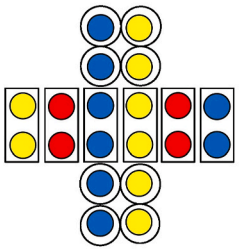
(c) Schema 4 Algorithm 3: Column by column



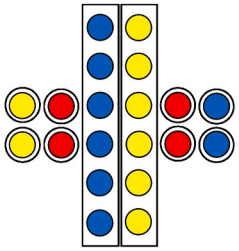
(d) Schema 4 Algorithm 4: Two columns, two rows (with redundancy)



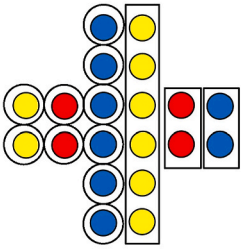
(e) Schema 4 Algorithm 5: Column by column, red all together



(f) Schema 4 Algorithm 6: Two rows column by column, point-by-point up and down

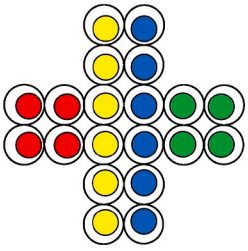


(g) Schema 4 Algorithm 7: Two columns, squares point-by-point

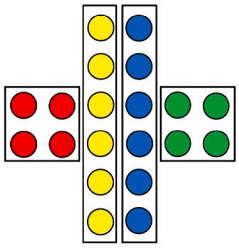


(h) Schema 4 Algorithm 8: Half point-by-point, the remaining column by column

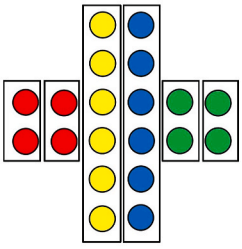
Fig. A.20. Algorithms that have been observed at least once for schema S4.



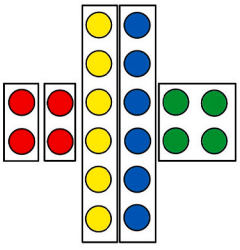
(a) Schema 5 Algorithm 1: Point by point



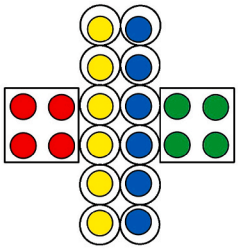
(b) Schema 5 Algorithm 2: Two squares in both sides, two columns



(c) Schema 5 Algorithm 3: Column by column

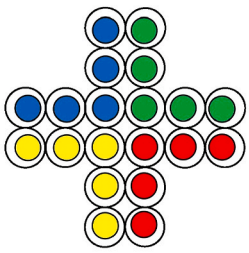


(d) Schema 5 Algorithm 4: Column by column, one square

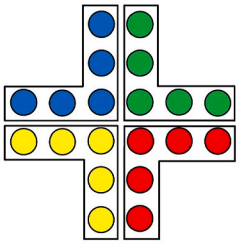


(e) Schema 5 Algorithm 5: Two squares in both sides, columns point-by-point

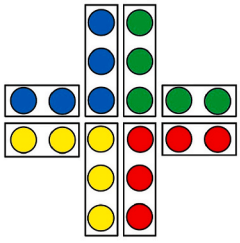
Fig. A.21. Algorithms that have been observed at least once for schema S5.



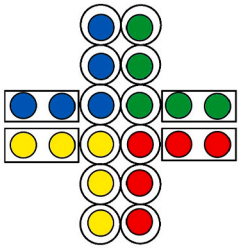
(a) Schema 6 Algorithm 1: Point by point



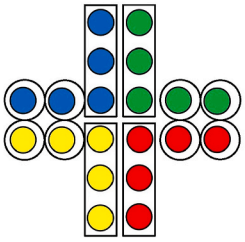
(b) Schema 6 Algorithm 2: Four L



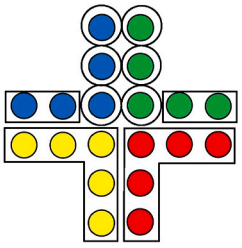
(c) Schema 6 Algorithm 3: Four L split in two



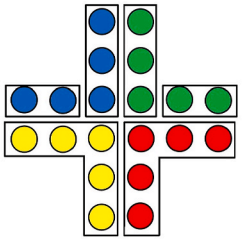
(d) Schema 6 Algorithm 4: Two columns row by row, on the sides row by row



(e) Schema 6 Algorithm 5: Column by column

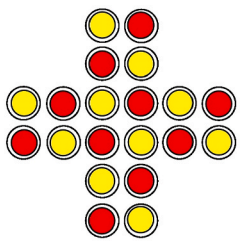


(f) Schema 6 Algorithm 6: Two L, two couples, six point-by-point

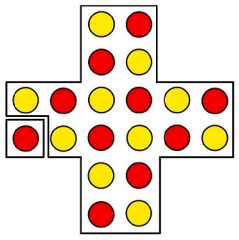


(g) Schema 6 Algorithm 7: Two L, two L split in two

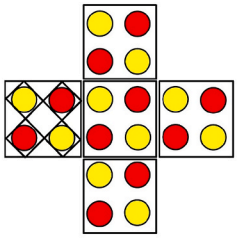
Fig. A.22. Algorithms that have been observed at least once for schema S6.



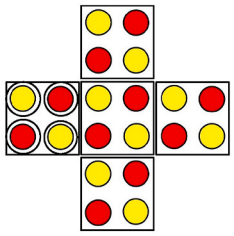
(a) Schema 7 Algorithm 1: Point by point



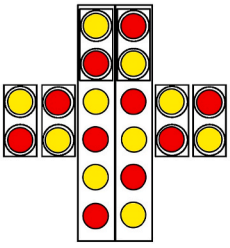
(b) Schema 7 Algorithm 2: Chessboard



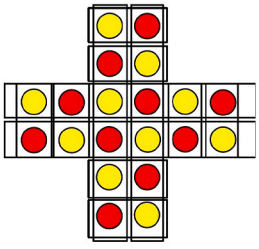
(c) Schema 7 Algorithm 3: Five squares composed of two diagonals



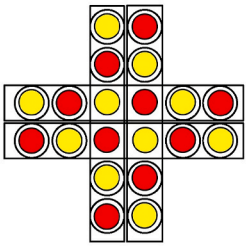
(d) Schema 7 Algorithm 4: Five squares point by point



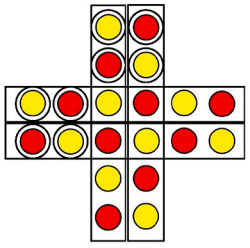
(e) Schema 7 Algorithm 5: Alternated columns



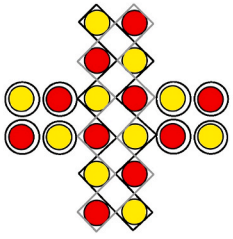
(f) Schema 7 Algorithm 6: Alternated columns and rows (with redundancy)



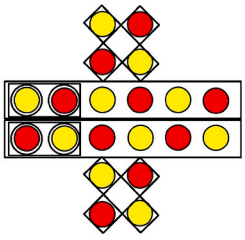
(g) Schema 7 Algorithm 7: Alternated starting from the edges



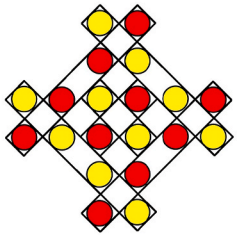
(h) Schema 7 Algorithm 8: Repeated and alternated columns and row



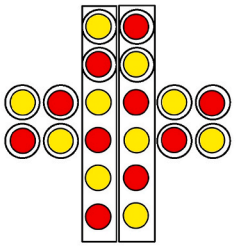
(i) Schema 7 Algorithm 9: Two columns alternated with zig zag, point-by-point on the sides



(j) Schema 7 Algorithm 10: Two rows alternated, diagonals in the squares up and down

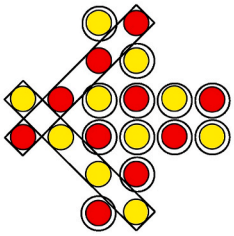


(k) Schema 7 Algorithm 11: Diagonal by diagonal

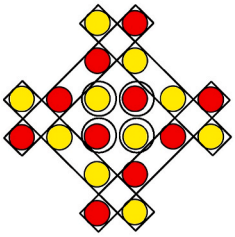


(l) Schema 7 Algorithm 12: Two columns alternated, two squares point-by-point

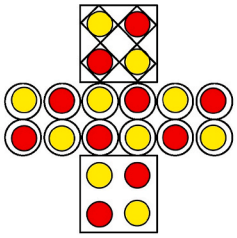
Fig. A.23. Algorithms that have been observed at least once for schema S7.



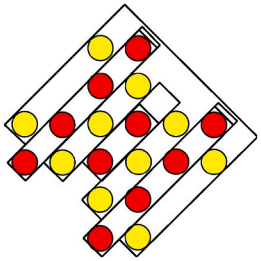
(m) Schema 7 Algorithm 13: Two diagonals, the remaining point-by-point



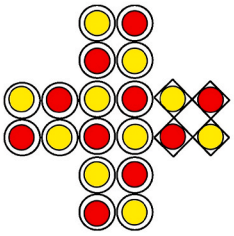
(n) Schema 7 Algorithm 14: Four diagonals, central square point-by-point



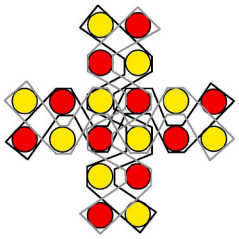
(o) Schema 7 Algorithm 15: Two rows point by point, two squares equals



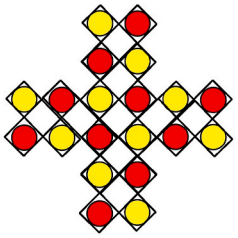
(p) Schema 7 Algorithm 16: Red diagonals, the remaining yellow



(q) Schema 7 Algorithm 17: Point by point, one square composed of diagonals

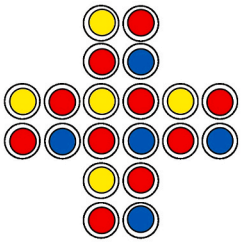


(r) Schema 7 Algorithm 18: Vertical and horizontal zig zag

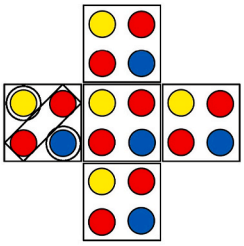


(s) Schema 7 Algorithm 19: Diagonals of two points with intersection (with redundancy)

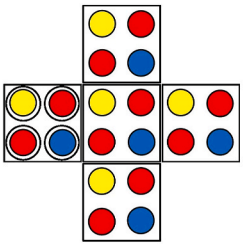
Fig. A.23. (*continued*).



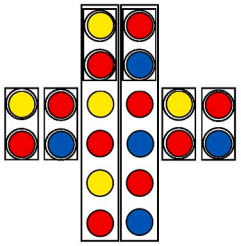
(a) Schema 8 Algorithm 1: Point by point



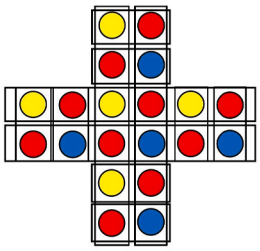
(b) Schema 8 Algorithm 2: Five squares with red diagonal and two points



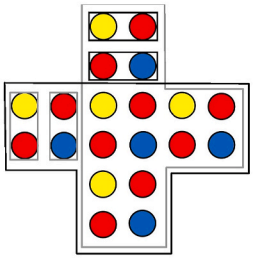
(c) Schema 8 Algorithm 3: Five squares point by point



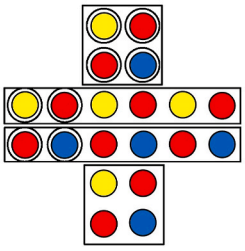
(d) Schema 8 Algorithm 4: Column by column alternated



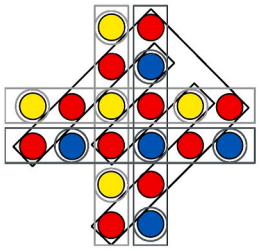
(e) Schema 8 Algorithm 5: Alternated columns and rows (with redundancy)



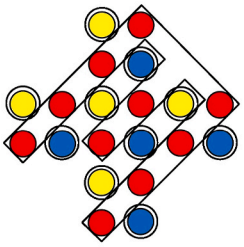
(f) Schema 8 Algorithm 6: Repeated and alternated column by column and row by row



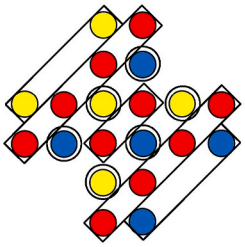
(g) Schema 8 Algorithm 7: Two rows alternated, two identical squares point-by-point



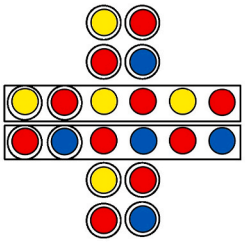
(h) Schema 8 Algorithm 8: Red alternated one yes and one no vertically and horizontally, yellow among reds in the first columns and in the first row, the others blue



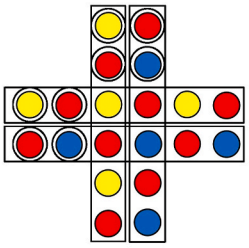
(i) Schema 8 Algorithm 9: Red alternated one yes and one no vertically and horizontally, the remaining point-by-point



(j) Schema 8 Algorithm 10: Diagonal by diagonal

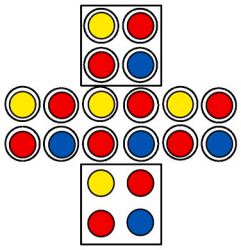


(k) Schema 8 Algorithm 11: Two rows alternated, two squares point-by-point

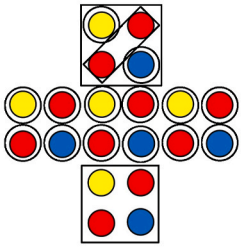


(l) Schema 8 Algorithm 12: Repeated and alternated columns and row

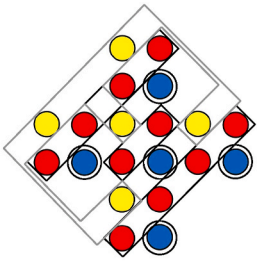
Fig. A.24. Algorithms that have been observed at least once for schema S8.



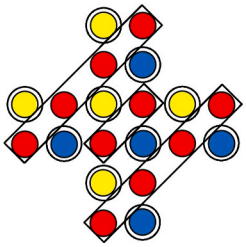
(m) Schema 8 Algorithm 13: Two rows point by point, two identical squares point-by-point



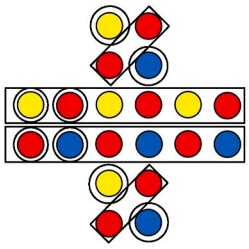
(n) Schema 8 Algorithm 14: Two rows point by point, two identical squares with diagonal



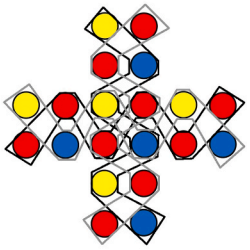
(o) Schema 8 Algorithm 15: Red diagonals, blue point-by-point, remaining yellow



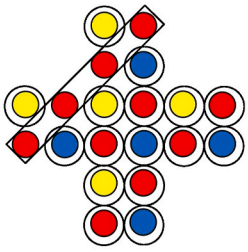
(p) Schema 8 Algorithm 16: Red diagonals, the remaining point-by-point



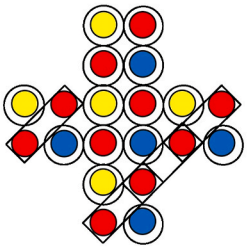
(q) Schema 8 Algorithm 17: Two rows alternated, two squares with diagonals



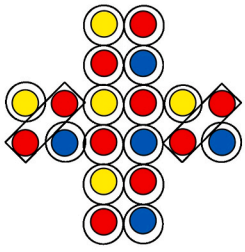
(r) Schema 8 Algorithm 18: Vertical and horizontal zig zag



(s) Schema 8 Algorithm 19: One red diagonal of four points, the remaining point-by-point

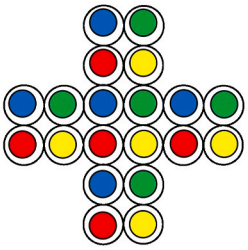


(t) Schema 8 Algorithm 20: Four pairs of red, the remaining point-by-point

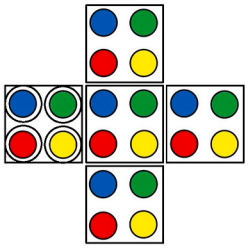


(u) Schema 8 Algorithm 21: Two squares with diagonal, the remaining point-by-point

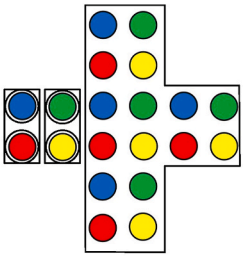
Fig. A.24. (*continued*).



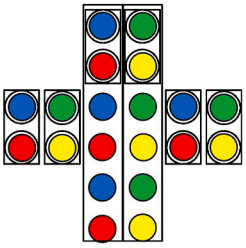
(a) Schema 9 Algorithm 1: Point by point



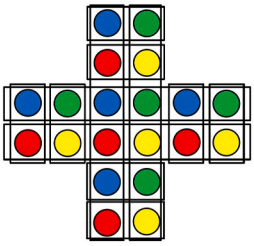
(b) Schema 9 Algorithm 2: Five squares point by point



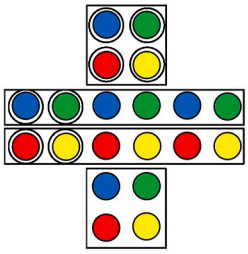
(c) Schema 9 Algorithm 3: Repeated and alternated column by column



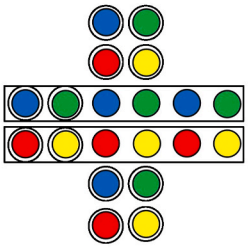
(d) Schema 9 Algorithm 4: Column by column alternated



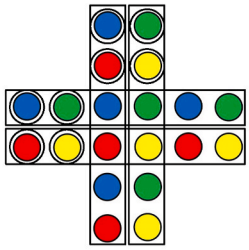
(e) Schema 9 Algorithm 5: Alternated columns and rows (with redundancy)



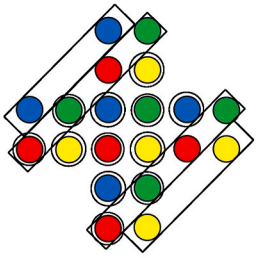
(f) Schema 9 Algorithm 6: Two rows alternated, two identical squares point-by-point



(g) Schema 9 Algorithm 7: Two rows alternated, two squares point-by-point

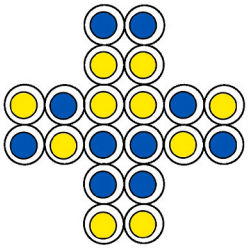


(h) Schema 9 Algorithm 8: Repeated and alternated columns and row

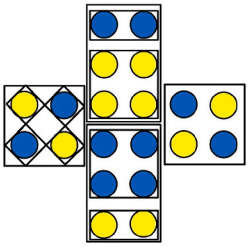


(i) Schema 9 Algorithm 9: Diagonal by diagonal with pairs

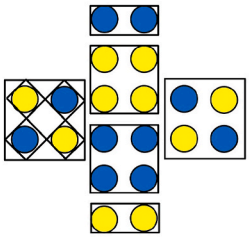
Fig. A.25. Algorithms that have been observed at least once for schema S9.



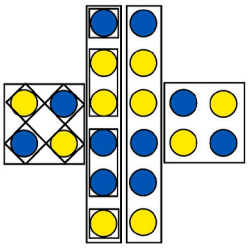
(a) Schema 10 Algorithm 1: Point by point



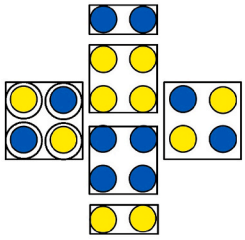
(b) Schema 10 Algorithm 2: Inverted symmetry for the two columns, two symmetric squares on the sides composed by diagonal



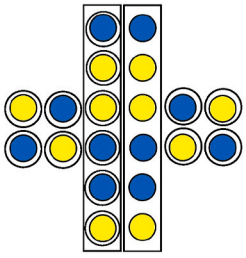
(c) Schema 10 Algorithm 3: Two columns with two rows and two squares, two symmetric squares on the sides composed by diagonal



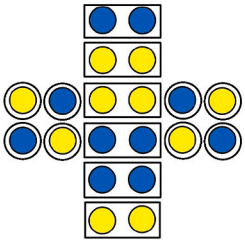
(d) Schema 10 Algorithm 4: Two identical columns with two pairs, two symmetric squares on the sides composed by diagonal



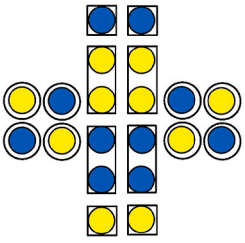
(e) Schema 10 Algorithm 5: Two columns with two rows and two squares, two symmetric squares on the sides point by point



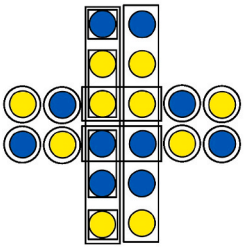
(f) Schema 10 Algorithm 6: Two identical columns point-by-point, two sides point-by-point



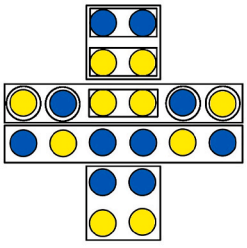
(g) Schema 10 Algorithm 7: Two columns row by row, two sides point-by-point



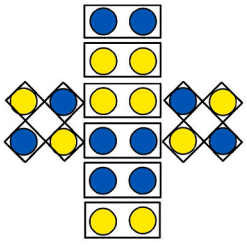
(h) Schema 10 Algorithm 8: Column by column



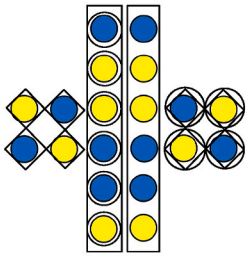
(i) Schema 10 Algorithm 9: Two identical columns with two pairs, two rows with pairs (with redundancy)



(j) Schema 10 Algorithm 10: Two identical squares up and down with pairs, two symmetric rows with pair of points

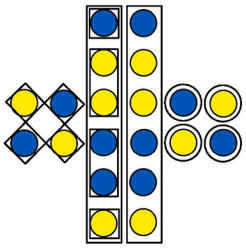


(k) Schema 10 Algorithm 11: Two columns row by row, double diagonal on both sides

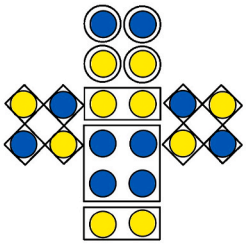


(l) Schema 10 Algorithm 12: Two identical columns point-by-point, double diagonal on both sides (one side redundant point by point)

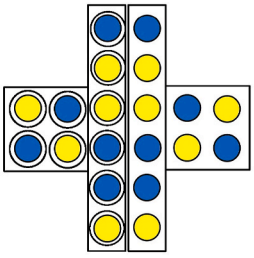
Fig. A.26. Algorithms that have been observed at least once for schema S10.



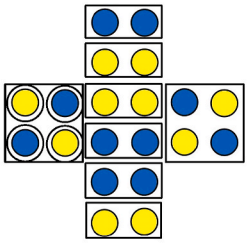
(m) Schema 10 Algorithm 13: Two columns with two rows and two squares, one side diagonals, other side point-by-point



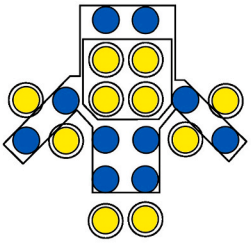
(n) Schema 10 Algorithm 14: In the central column a square point-by-point, then a row then square and row, double diagonal on both sides



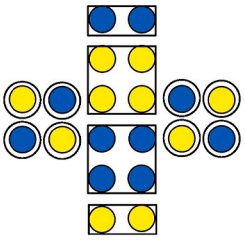
(o) Schema 10 Algorithm 15: Two identical columns point-by-point, two symmetric squares point-by-point



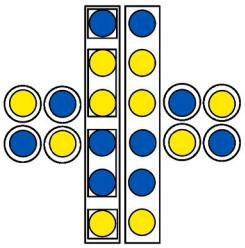
(p) Schema 10 Algorithm 16: Two columns row by row, two symmetric squares point-by-point



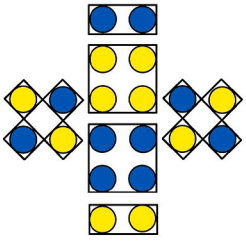
(q) Schema 10 Algorithm 17: One colour point by point, the remaining with the other colour



(r) Schema 10 Algorithm 18: Two columns with two rows and two squares, two squares on the sides point-by-point

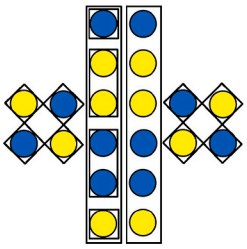


(s) Schema 10 Algorithm 19: Two identical columns with two pairs, two squares on the sides point-by-point

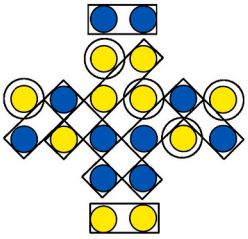


(t) Schema 10 Algorithm 20: Two columns with two rows and two squares, double diagonal on both sides

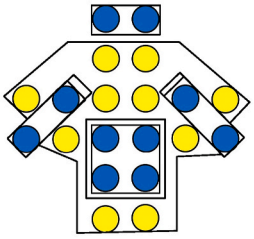
Fig. A.26. (*continued*).



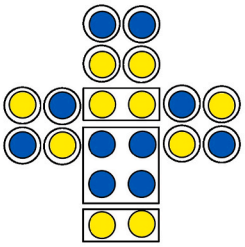
(u) Schema 10 Algorithm 21: Two identical columns with two pairs, double diagonal on both sides



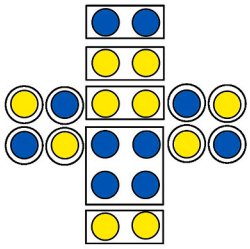
(v) Schema 10 Algorithm 22: Blue in pairs and



(w) Schema 10 Algorithm 23: One square blue, other blue in pairs, the remaining yellow

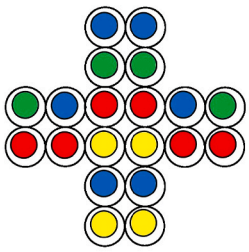


(x) Schema 10 Algorithm 24: In the central column a square point-by-point, two rows and a square, two sides point-by-point

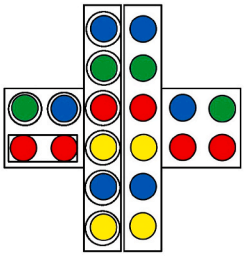


(y) Schema 10 Algorithm 25: In the central column a square and remaining by rows, two sides point-by-point

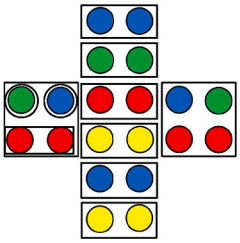
Fig. A.26. (*continued*).



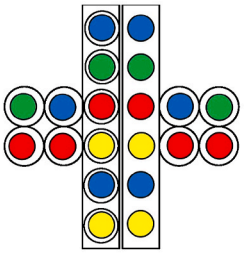
(a) Schema 11 Algorithm 1: Point by point



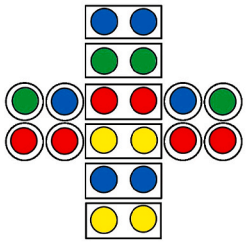
(b) Schema 11 Algorithm 2: Two identical columns point-by-point, on one side a red line and two points, mirrored on the other side



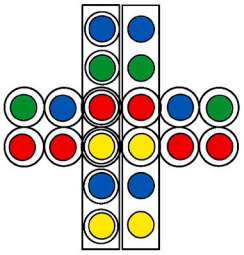
(c) Schema 11 Algorithm 3: Two columns row by row, on one side a red line and two points, mirrored on the other side



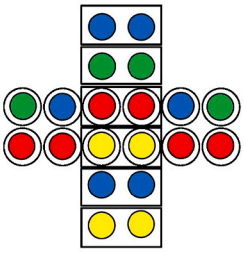
(d) Schema 11 Algorithm 4: Two identical columns point-by-point, on the sided point-by-point



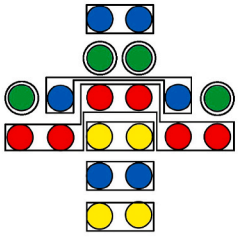
(e) Schema 11 Algorithm 5: Two columns row by row, on the sided point-by-point



(f) Schema 11 Algorithm 6: Two identical columns point-by-point, two rows point-by-point (with redundancy)

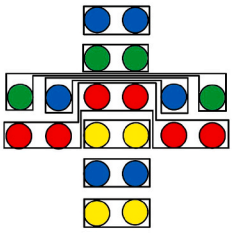


(g) Schema 11 Algorithm 7: Two columns row by row, two rows point-by-point (with redundancy)

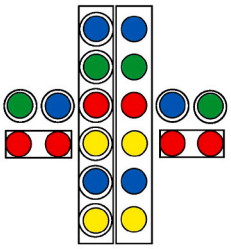


(h) Schema 11 Algorithm 8: Blue and yellow in pairs, green point-by-point, remaining in red

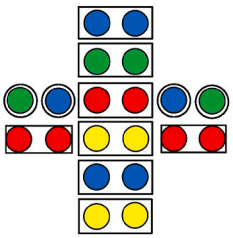
Fig. A.27. Algorithms that have been observed at least once for schema S11.



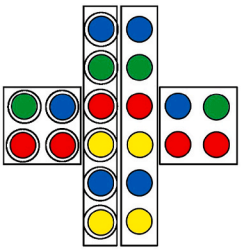
(i) Schema 11 Algorithm 9: Blue, yellow and green in pairs, the remaining red



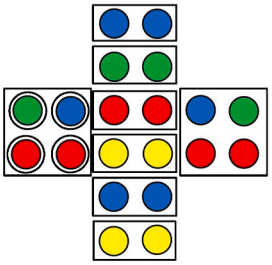
(j) Schema 11 Algorithm 10: Two identical columns point-by-point, two pairs of red, the remaining point by point



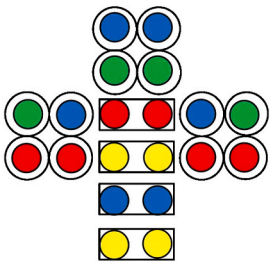
(k) Schema 11 Algorithm 11: Two columns row by row, two pairs of red, the remaining point-by-point



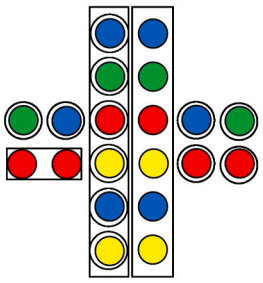
(l) Schema 11 Algorithm 12: Two identical columns point-by-point, on one side point-by-point, mirrored on the other side



(m) Schema 11 Algorithm 13: Two columns with 4 pairs, the remaining point-by-point

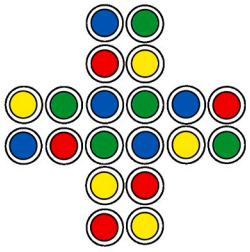


(n) Schema 11 Algorithm 14: Two identical columns point-by-point, one pair of red, the remaining point by point

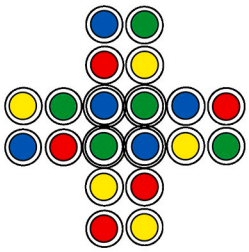


(o) Schema 11 Algorithm 15: Two columns row by row, one pair of red, the remaining point-by-point

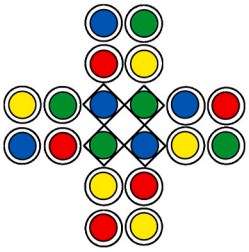
Fig. A.27. (*continued*).



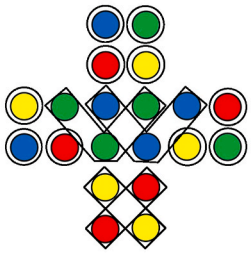
(a) Schema 12 Algorithm 1: Point by point



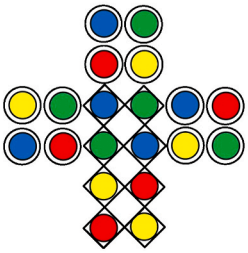
(b) Schema 12 Algorithm 2: Point by point (with redundancy)



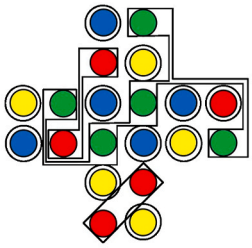
(c) Schema 12 Algorithm 3: Point by point except one square with diagonals



(d) Schema 12 Algorithm 4: L of three green, L of three blue, a square with diagonals, the remaining point by point



(e) Schema 12 Algorithm 5: Point by point except two squares with diagonals



(f) Schema 12 Algorithm 6: Blue and yellow point-by-point, red in pairs, the remaining green

Fig. A.28. Algorithms that have been observed at least once for schema S12.

Appendix B. Schemas performance

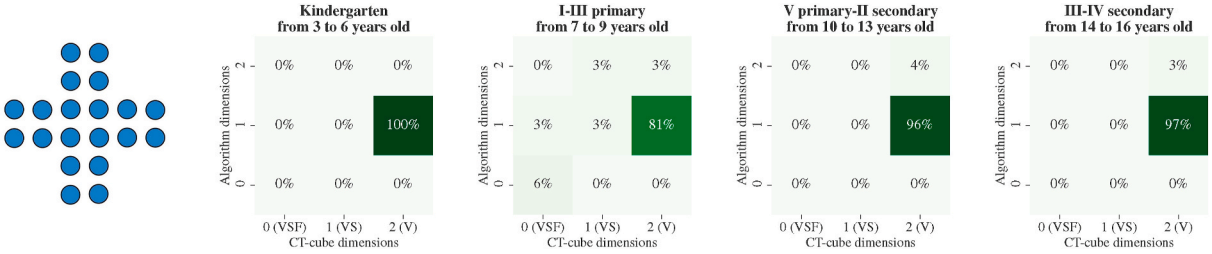


Fig. B.29. Performance analysis across age categories of S1.

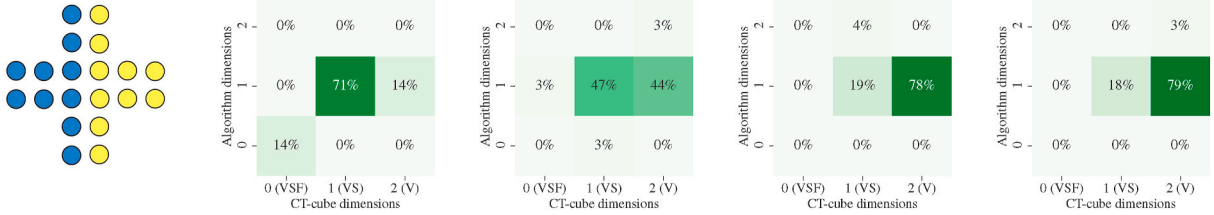


Fig. B.30. Performance analysis across age categories of S2.

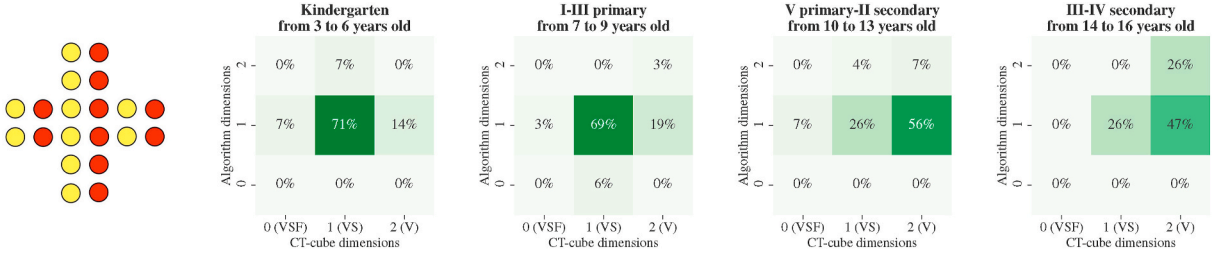


Fig. B.31. Performance analysis across age categories of S3.

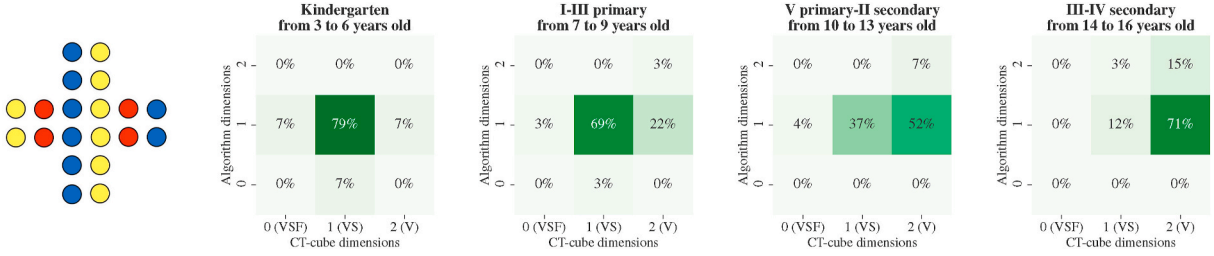


Fig. B.32. Performance analysis across age categories of S4.

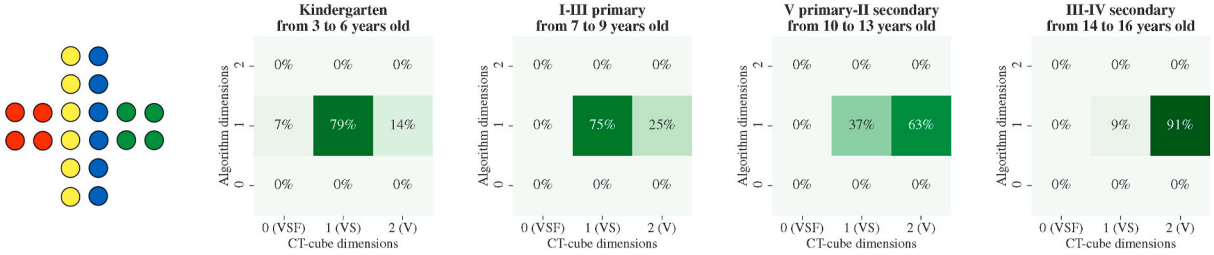


Fig. B.33. Performance analysis across age categories of S5.

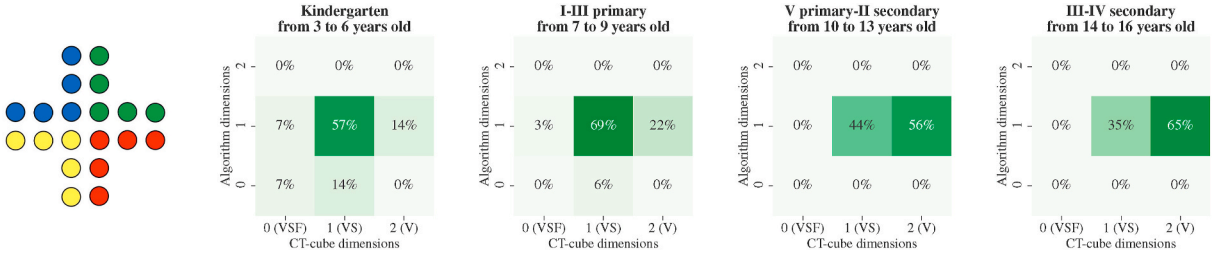


Fig. B.34. Performance analysis across age categories of S6.

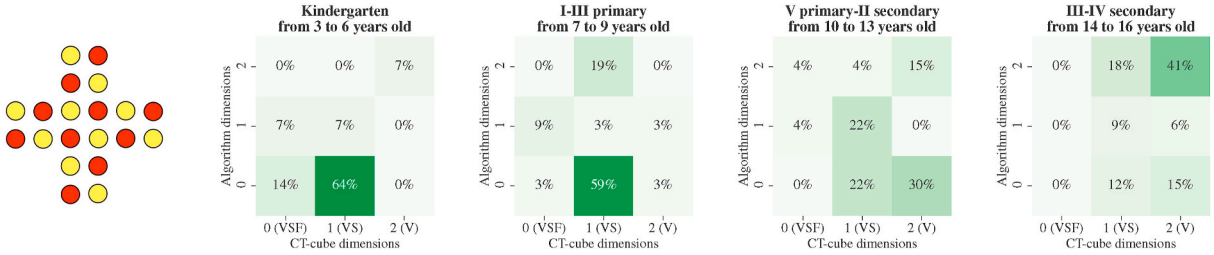


Fig. B.35. Performance analysis across age categories of S7.

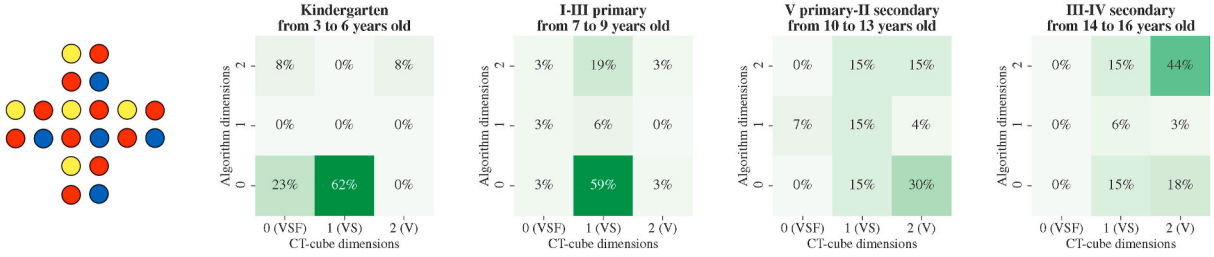


Fig. B.36. Performance analysis across age categories of S8.

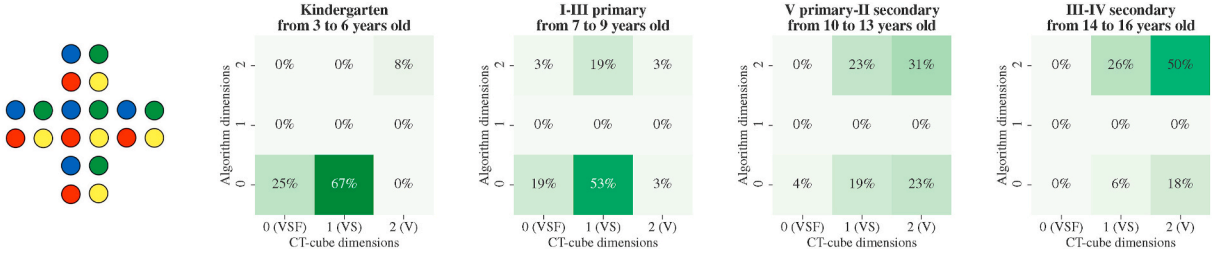


Fig. B.37. Performance analysis across age categories of S9.

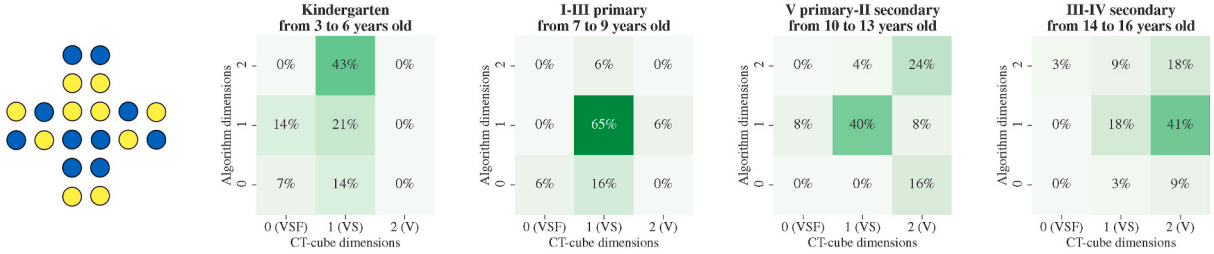


Fig. B.38. Performance analysis across age categories of S10.

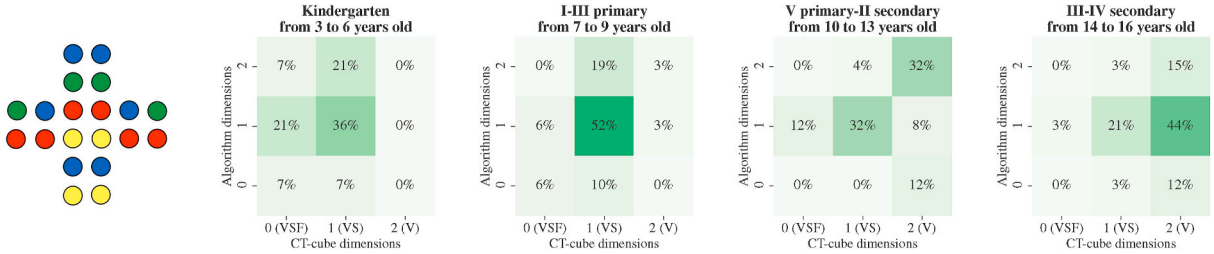


Fig. B.39. Performance analysis across age categories of S11.

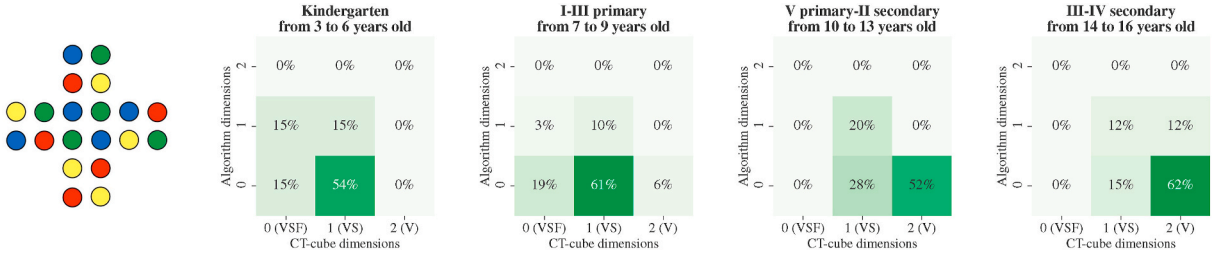


Fig. B.40. Performance analysis across age categories of S12.

References

Aebi-Müller, R. E., Blatter, I., Brigger, J., Constable, E. C., Eglin, N., Hoffmeyer, P., Lautenschütz, C. C., Lienhard, A., Pirinoli, C., Röthlisberger, M., et al. (2021). *Code of conduct for scientific integrity*.

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems, 75*, 661–670. https://doi.org/10.1016/j.robot.2015.10.008

Bell, T. (2018). CS unplugged—how is it used, and does it work?. In *Adventures between lower bounds and higher altitudes* (pp. 497–521). Springer International Publishing. https://doi.org/10.1007/978-3-319-98355-4_29.

Benoit, L., Lehalle, H., & Jouen, F. (2004). Do young children acquire number words through subitizing or counting? *Cognitive Development, 19*, 291–307. https://doi.org/10.1016/j.cogdev.2004.03.005

Brackmann, C., Barone, D., Casali, A., Boucinha, R., & Muñoz-Hernandez, S. (2016). Computational thinking: Panorama of the Americas. In *2016 International symposium on computers in education (SIIE)* (pp. 1–6). IEEE. https://doi.org/10.1109/SIIE.2016.7751839.

Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., & Barone, D. (2017). Development of computational thinking skills through unplugged activities in primary school. In *Proceedings of the 12th workshop on primary and secondary computing education* (pp. 65–72). ACM. https://doi.org/10.1145/3137065.3137069.

Bruce, R., & Threlfall, J. (2004). One, two, three and counting. *Educational Studies in Mathematics, 55*, 3–26. https://doi.org/10.1023/B:EDUC.0000017676.79430.dc

Cantone, T. (2015). *Piano di studio della scuola dell'obbligo ticinese*. Lugano: Società d'arti grafiche Veladini.

Cortina, T. J. (2015). Reaching a broader population of students through "unplugged" activities. *Communications of the ACM, 58*, 25–27. https://doi.org/10.1145/2723671

Dehaene, S. (2011). *The number sense: How the mind creates mathematics* (Revised and Updated Edition). USA: Oxford University Press.

Delal, H., & Oner, D. (2020). Developing middle school students' computational thinking skills using unplugged computing activities. *Informatics in Education, 19*, 1–13. https://doi.org/10.15388/infedu.2020.01

Dietz, G., Landay, J. A., & Gweon, H. (2019). *Building blocks of computational thinking: Young children's developing capacities for problem decomposition*. CogSci.

Gelman, R., & Gallistel, C. R. (1986). *The child's understanding of number*. Harvard University Press.

Heersmink, R. (2013). A taxonomy of cognitive artifacts: Function, information, and categories. *Rev. Philos. Psychol., 4*, 465–481. https://doi.org/10.1007/s13164-013-0148-1

Hermans, F., & Aivaloglou, E. (2017). To scratch or not to scratch?: A controlled experiment comparing plugged first and unplugged first programming lessons. In *Proceedings of the 12th workshop on primary and secondary computing education* (pp. 49–56). ACM. https://doi.org/10.1145/3137065.3137072.

Kafai, Y., Proctor, C., & Lui, D. (2020). From theory bias to theory dialogue: Embracing cognitive, situated, and critical framings of computational thinking in K-12 CS education. *ACM Inroads, 11*, 44–53. https://doi.org/10.1145/3381887

Kalelioglu, F., Gulbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic J. Mod. Comput., 4*, 583–596.

Klahr, D., & Robinson, M. (1981). Formal assessment of problem-solving and planning processes in preschool children. *Cognitive Psychology, 13*, 113–148. https://doi.org/10.1016/0010-0285(81)90006-2

Lu, J. J., & Fletcher, G. H. L. (2009). Thinking about computational thinking. In *Proceedings of the 40th ACM technical symposium on Computer science education* (pp. 260–264). SIGCSE '09. https://doi.org/10.1145/1508865.1508959.

Metin, S. (2020). Activity-based unplugged coding during the preschool period. *International Journal of Technology and Design Education*, 1–17. https://doi.org/10.1007/s10798-020-09616-8

del Olmo-Muñoz, J., Cózar-Gutiérrez, R., & González-Calero, J. A. (2020). Computational thinking through unplugged activities in early years of Primary Education. *Computers & Education, 150*, Article 103832. https://doi.org/10.1016/j.compedu.2020.103832

Petousi, V., & Sifaki, E. (2020). Contextualising harm in the framework of research misconduct. findings from discourse analysis of scientific publications. *International Journal of Sustainable Development, 23*, 149–174. https://doi.org/10.1504/IJSD.2020.115206

Rapaport, W. J. (2015). Philosophy of computer science, Teaching philosophy. *Current draft in progress at*. https://cse.buffalo.edu/rapaport/Papers/phics.pdf.

Relkin, E., Bers, M., & TechCheck-, K. (2021). A measure of computational thinking for kindergarten children. In *2021 IEEE global engineering education conference* (pp. 1696–1702). EDUCON). https://doi.org/10.1109/EDUCON46332.2021.9453926.

Relkin, E., de Ruiter, L., & Bers, M. U. (2020). TechCheck: Development and validation of an unplugged assessment of computational thinking in early childhood education. *Journal of Science Education and Technology, 29*, 482–498. https://doi.org/10.1007/s10956-020-09831-x

Relkin, E., de Ruiter, L., & Bers, M. (2021). Learning to code and the acquisition of computational thinking by young children. *Computers & Education, 169*, 104222. https://doi.org/10.1016/j.compedu.2021.104222

Rodriguez, B., Rader, C., & Camp, T. (2016). Using student performance to assess CS unplugged activities in a classroom environment. In *Proceedings of the 2016 ACM conference on innovation and technology in computer science education* (pp. 95–100). ACM. https://doi.org/10.1145/2899415.2899465.

Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test. *Computers in Human Behavior, 72*, 678–691. https://doi.org/10.1016/j.chb.2016.08.047

Román-González, M., Pérez-González, J.-C., Moreno-León, J., & Robles, G. (2018). Can computational talent be detected? Predictive validity of the computational thinking test. *Int. J. Child-Comput. Interact., 18*, 47–58. https://doi.org/10.1016/j.ijcci.2018.06.004

Roth, W.-M., & Jornet, A. (2013). *Situated cognition, Wiley Interdisciplinary Reviews: Cognitive Science, 4*, 463–478. https://doi.org/10.1002/wcs.1242

Sarnecka, B. W., & Carey, S. (2008). How counting represents number: What children must learn and when they learn it. *Cognition, 108*, 662–674. https://doi.org/10.1016/j.cognition.2008.05.007

Saxena, A., Lo, C. K., Hew, K. F., & Wong, G. K. W. (2020). Designing unplugged and plugged activities to cultivate computational thinking: An exploratory study in early childhood education. *Asia-Pacific Educ. Res., 29*, 55–66. https://doi.org/10.1007/s40299-019-00478-w

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

Tall, D. (2006). A theory of mathematical growth through embodiment, symbolism and proof. *ANNALES de DIDACTIQUE et de SCIENCES COGNITIVES, IREM de STRASBOURG, 11*, 195–215. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.377.5017&rep=rep1&type=pdf.

Tall, D. (2013). *How humans learn to think mathematically: Exploring the three worlds of mathematics*. Cambridge University Press. https://doi.org/10.1017/CBO9781139565202

Tall, D. (2020). *Making sense of mathematical thinking over the long term: The framework of three worlds of mathematics and new developments, MINTUS: Beiträge zur mathematischen, naturwissenschaftlichen und technischen Bildung*. Wiesbaden: Springer. https://homepages.warwick.ac.uk/staff/David.Tall/pdfs/dot2020a-3worlds-extension.pdf.

Tikva, C., & Tambouris, E. (2021). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature review. *Computers & Education, 162*, 104083. https://doi.org/10.1016/j.compedu.2020.104083

Tsarava, K., Moeller, K., & Ninaus, M. (2018). Training computational thinking through board games: The case of crabs & turtles. *Int. J. Serious Games, 5*, 25–44. https://doi.org/10.17083/ijsg.v5i2.248

Tsarava, K., Moeller, K., Pinkwart, N., Butz, M., Trautwein, U., & Ninaus, M. (2017). Training computational thinking: Game-based unplugged and plugged-in activities in primary school. In *European conference on games based learning* (pp. 687–695). Academic Conferences International Limited.

Unnikrishnan, R., Amrita, N., Muir, A., & Rao, B. (2016). Of elephants and nested loops: How to introduce computing to youth in rural India. In *Proceedings of the the 15th International conference on Interaction design and children* (pp. 137–146). ACM. https://doi.org/10.1145/2930674.2930678.

Wallet, P., et al. (2015). *Information and communication technology (ICT) in education in sub-Saharan Africa: A comparative analysis of basic e-readiness in schools*. Information Paper No. 25. UNESCO Institute for Statistics. https://doi.org/10.15220/978-92-9189-178-8-en.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*, 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2014). *Computational thinking benefits society, 40th anniversary blog of social issues in computing*, 2014 2014 http://socialissues.cs.toronto.edu/index.html%3Fp=279.html.

Wohl, B., Porter, B., & Clinch, S. (2015). Teaching computer science to 5-7 year-olds: An initial study with scratch, cubelets and unplugged computing. In *Proceedings of the workshop in primary and secondary computing education* (pp. 55–60). ACM. https://doi.org/10.1145/2818314.2818340.

Zapata-Cáceres, M., Martín-Barroso, E., & Román-González, M. (2020). Computational thinking test for Beginners: Design and content validation. In *2020 IEEE global engineering education conference* (pp. 1905–1914). EDUCON). https://doi.org/10.1109/EDUCON45650.2020.9125368.