# Non-Deterministic Computational Thinking: Challenges and Opportunities

**Leandro Ferreira Paz**
MSc. Student, Federal Institute of Education, Science and Technology Farroupilha,
Alegrete-RS, Brazil.
Email: leandro.paz@iffarroupilha.edu.br

**Cristiano Gomes Carvalho**
MSc. Student, Federal Institute of Education, Science and Technology Farroupilha,
Alegrete-RS, Brazil.
Email: cristiano.carvalho@iffarroupilha.edu.br

**Andréia dos Santos Sachete**
Professor, Federal Institute of Education, Science and Technology Farroupilha,
Alegrete-RS, Brazil.
ORCID: https://orcid.org/0000-0003-2226-3322
Email: andreia.sachete@iffarroupilha.edu.br

**Marcele Teixeira Homrich Ravasio**
Professor, Federal Institute of Education, Science and Technology Farroupilha,
Santo Ângelo-RS, Brazil.
ORCID: https://orcid.org/0000-0003-0162-1426
Email: marcele.ravasio@iffarroupilha.edu.br

**Ricardo Antonio Rodrigues**
Professor, Federal Institute of Education, Science and Technology Farroupilha,
Jaguari-RS, Brazil.
ORCID: https://orcid.org/0000-0002-5292-3646
Email: ricardo.rodrigues@iffarroupilha.edu.br

**Fábio Diniz Rossi (Corresponding author)**
Professor, Federal Institute of Education, Science and Technology Farroupilha,
Alegrete-RS, Brazil.
ORCID: https://orcid.org/0000-0002-2450-1024
Email: fabio.rossi@iffarroupilha.edu.br

## Abstract

*An educational paradigm that has improved problem-solving capacity is computational thinking, which uses characteristics such as decomposition, abstraction, pattern recognition, and algorithmic thinking. However, most of the resources developed under this paradigm are deterministic. However, the current world is not linear. Non-deterministic dynamics play a vital role in today's world. Decisions about the same fact can cause different events, and students must be prepared to live with such uncertainties. This article discusses challenges and possibilities in the development of non-deterministic computational thinking resources. This work shows a large field of research yet to be worked on, with new possibilities and a great potential to connect new resources with the students' daily lives.*

**Keywords:** automata; computational thinking; non-deterministic.

## 1. Introduction

Computational thinking [4] is the process of understanding aspects of computing in our world and applying tools and techniques to facilitate systems and processes. At school, it can be exemplified when students solve problems, dividing them in part and using logic. This concept comprises critical, strategic, and creative skills, using the fundamentals of computing in different areas of life. Thus, either individually or in groups, the student can reason and resolve issues. Unlike what the name suggests, computational thinking is not just related to technology or programming, much less requires using a computer. This competence proposes that individuals can identify problems and find solutions creatively and use other types of knowledge, and can be divided in some steps [3].

- Decomposition. This ability refers to the ability to divide a more significant problem into smaller parts. Thus, when working on each piece at once, it is easier for the student to understand the problem and solve it. The advantage is that it reduces anxiety and fear of facing challenges. In elaborating a large school project, the decomposition is applied when the student is organized and proposes to do a part each week.
- Abstraction. Abstraction proposes to focus on essential processes instead of prioritizing details. In this way, the solution can be valid for several different problems. When applying it in the educational area, the student understands how to perform exercises more quickly and can use these techniques in other subjects, for example.
- Pattern recognition. Pattern recognition is another fundamental principle of computational thinking. By doing so, the child can identify trends in behavior and similarities. From this, it is possible to think of new solutions using innovation and creativity.
- Algorithmic thinking. The word algorithm refers to its computational context, but it can also symbolize the creation of steps and solutions until reaching an objective. It is the use of logic and rationality to solve problems.

Computational thinking has several advantages for children and helps in the individual's socio-cognitive development [1] [2]. This concept prepares young people to identify information and produce something important from these concepts and apply them in their daily lives. Also, it contributes to the adaptation of classes to a social context. Individuals will be prepared to develop diverse skills that will be important in the labor market demands.

Computational thinking stimulates the construction of logical thinking [4]. Therefore, from childhood, the child must engage in activities that encourage the perception of patterns and specific actions. After a while, logically, she will be able to solve problems herself through rationality. Abstraction and algorithmic thinking directly influence the child's creativity and autonomy. When your child has contact with these stimuli, he/she ceases to be a consumer of information and starts to create and produce as well. This is a necessary process, as it prepares the young person for the world and makes him/her stand out from the rest.

One of the expressions of computational thinking is Computer Science (CS) Unplugged [5]. It consists of free and accessible activities that bring concepts and problems from the world of computing, but without using any computer or electronic equipment. The activities use games, challenges, and puzzles that use simple materials such as pencils, paper, pens, and lots of movement. With that, this type of reasoning focused on problem-solving can be applied to any school environment, even those that do not have computational resources.

Based on the above, computational thinking through CS Unplugged is not only related to programming and technology [6]. When the child builds logical thinking and uses the right tools to solve problems, he/she can repeat the process and apply it in any discipline. However, most of the resources produced using computational thinking are based on deterministic solutions. A deterministic solution behaves the same way in different executions, given the same inputs and the same internal state of the machine. It does not always represent or reproduce the problems we encounter daily. The same actions in the same situations can cause a multitude of responses. And that is the concept behind non-determinism.

Perhaps this imbalance between the production of deterministic and non-deterministic CS Unplugged resources comes from computing itself, where the vast majority of problems are deterministic. In this article, we desire to explore non-determinism as an option in developing computational thinking (especially when produced for unplugged CS), presenting some challenges and opportunities for applying non-determinism.

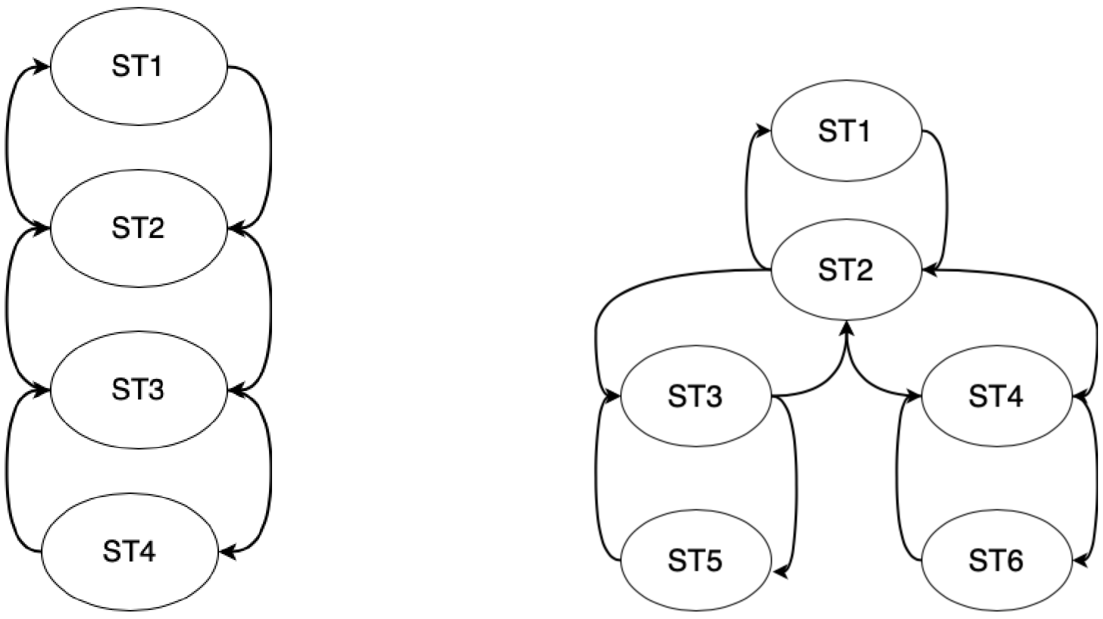
Section 2 presents the difference between determinism and computational non-determinism; Section 3 presents resources based on CS Unplugged from most cited repository by scientific works in the area, and classifies them between deterministic and non-deterministic, showing the imbalance between them; Section 4 presents the challenges imposed on the development of non-deterministic resources, followed by new opportunities in Section 5; Section 6 presents some conclusions.

## 2. Automata Theory: A Brief Review

In computer science, automata theory is the study of mathematical objects called abstract machines, or automata, and the computational problems that can be solved using these objects [12]. Among the automata, we have the finite state machine that consists of states (usually represented by circles) and transitions (represented by arrows). When the automaton receives an input symbol, it transitions (or jumps) to another state, according to its transition function (whose inputs are the current state and the recent symbol).

Therefore, an automaton consists of a digital computer model that can be used to represent languages. An automaton has some essential characteristics. It has a mechanism for reading entries. Assuming that these inputs are strings over a given alphabet, written on an input tape, the automaton can read but not change. The input mechanism can read the input tape from left to right, one symbol at a time. The input mechanism can detect the end of the input chain. There are two basic types of finite automata: deterministic and non-deterministic.

A deterministic finite automaton (Figure 1(a)) is defined by a set of internal states, a set of symbols (called the input alphabet), a function called the transition function, an initial state, and a set of final states. Graphs of transitions are used to visualize and represent automata in which the vertices represent states, and the edges represent transitions. The vertex labels are the state names, while the edge labels are the current values of the input symbols.



(a) deterministic finite automata (b) non-deterministic finite automata

Figure 1: Basic types of automata

The second type is the non-deterministic finite automata (Figure 1(b)). Its main characteristic is the possibility of reaching a set of states from a given state by consuming a specific input symbol or without consuming any symbol. The expression "non-deterministic" is because it is impossible to predict the next state to be reached from a symbol at a given time. Deterministic finite automata do not have this characteristic since, starting from each state, consuming itself some input symbol will reach some state of the automaton. Besides, non-determinism means a choice of movements for an automaton. Instead of predicting a single direction in each situation, we accept a set of possible paths.

In this section, we presented a quick review of the theory of automata in order to explain the difference between determinism and non-determinism. Computer theory paves the way for the development of computational thinking. However, as we will see below, most of the resources produced for computational thinking activities are based on deterministic computational thinking. It means that there are several opportunities in the development of non-deterministic resources [11] [13].

## 3. Computational Thinking Resources

Nowadays, the importance of learning to program is becoming increasingly evident, a suggestion even from official documents that include logical thinking based on computing in elementary education in several countries [8]. The programmer's logical thinking allows the creation of algorithms to solve problems. There are several processes present in this creation: many times, the programmer needs to divide the problem into smaller parts, he needs to create strategies to correct his programming, he needs creativity. The programmer creates paths, scripts for the computer to understand what it expects.

These processes are also present in our daily lives, and they are essential for the formation of the individual who will act in society [7]. In this sense, it is vital to learn how to solve problems, develop creativity and logical thinking by creating scripts, paths, and strategies. Despite the need for programming, not all schools have computer labs or an internet connection. So, how to enable students to learn to program and develop technological fluency without a computer?

The most apparent possibility of bringing computational thinking to all teaching environments, formal or non-formal, is unplugged computing, which allows the performance of programming activities without the need for the use of computers, for example, through the use of boards, description of scripts, use of the creation of algorithms in written form, reorganization of infographics, use of blocks for programming in puzzles, among others.

The proposal addresses activities that involve concepts related to programming in a playful way and without the use of screens since early childhood education [9]. Therefore, unplugged computing emerges as an option to democratize technological fluency since students from smaller schools can also have access to activities developed in renowned schools, which already deal with robotics and programming in the computer lab. Still, it is noteworthy that schools with a range of digital resources can also opt for this type of activity, which awakens creativity, helps develop skills, and contributes to a reduction in excess exposure to screens [10].

One of the most reputable repositories of CS Unplugged resources is based at the University of Canterbury[1], NZ. The resources available in this repository present unplugged solutions to various everyday problems and allow the development of multiple capacities addressed by computational thinking. Table 1 presents the resources available in the repository, describes the applicability of each, and indicates whether the proposed solution is deterministic (D) or non-deterministic (ND).

[1] https://csunplugged.org/en/

Table 1. Classification of Resources between deterministic and non-deterministic

| Resource | Description | Solution Type |
|---|---|---|
| Arrows | This resource places a source and a destination on a square board, and the path must be plotted using a set of direction arrows. For each new round, there is always a fixed origin and destination. | D |
| Barcode Checksum | This resource is used to teach the calculation of a check digit based on a 12-digit checksum. The same input value will always generate the same check digit. | D |
| Binary Cards | This resource allows the teacher to develop conversion and understanding of binary numbers, using the sum of the decimal representation of each position of the binary number. | D |
| Binary to Alphabet | This resource allows the teacher to teach the student to convert based on a fixed table of representations, binary values in letters. | D |
| Binary Windows | This resource allows the teacher to develop conversion and understanding of binary numbers, using the sum of the decimal representation of each position of the binary number, from left to right. | D |
| Grid | This resource places a source and a destination on a square board, and the path must be plotted using a set of direction arrows. For each new round, there is always a fixed origin and destination. | D |
| Job Badges | This resource allows teachers to use software development roles (such as programmer, tester, and bots) to separate activities between students with a common goal. | ND |
| Left and Right Cards | This resource allows teachers to teach students notions of left and right through various activities. | D |
| Modulo Clock | This resource allows teachers to teach notions of time, ranging from months of the year, days of the week to counting time in minutes. | D |
| Number Hunt | This resource features a table of symbols related to the numbers. Two students must play, trying to guess which | D |

| | | |
|---|---|---|
| | number the other chose. Whoever makes the least attempts to find out the opponent's number wins. | |
| Parity Cards | This resource presents concepts of parity bits used in the verification and correction of data through a grid and markings in the lines of that grid. | D |
| Piano Keys | This resource features a keyboard that references piano notes in a digital format called MIDI and allows you to apply mathematical concepts to students. | D |
| Pixel Painter | This resource allows the representation of images bit by bit, which helps the teacher to teach how an image is formed in the digital number. | D |
| Searching Cards | This resource allows the teacher to teach algorithmic concepts (such as division and conquest) through values and images. | ND |
| Sorting Network | This resource shows how computers sort random numbers into order using a thing called a sorting network. | ND |
| Sorting Network Cards | These cards are to be used with the Sorting Network. | ND |
| Train Stations | This resource presents a circular or twisted route to be followed, passing through several stopping stations. | D |
| Treasure Island | This resource contains printouts for the Treasure Island activity for learning about Finite State Automata. | D |

When we see the classification of resources presented in Table 1, we can see that most of the resources developed are based on deterministic solutions, showing an imbalance between the two aforementioned behaviors. It opens up a promising research opportunity for the coming years. Although data from only one repository is presented (understood as the most important repository of unplugged computational thinking), several other unplugged repositories give the same amount between objects with deterministic and non-deterministic proposals[2][3][4][5].

[2] http://ispython.com/repository-1/
[3] https://www.pdcunplugged.org
[4] https://repositorio.grial.eu
[5] https://www.cde.state.co.us/computerscience/computer-science-resource-bank

## 4. Challenges

In a society that lives with constant technological advances and connectivity as something natural, the education sector needs to be connected with the news that technology offers. Computational thinking then assumes an essential role in the learning and training of children and young people, as it allows them to explore the elements that these resources offer. Cognitive development is linked to how human beings perceive the world: how they acquire knowledge, represent, live with other people, use language, imagination, and memory. Computational logic contributes to growth in this way of thinking and sees the world, facilitating learning and problem-solving [7].

The construction of logical reasoning is also one of the primary skills acquired. In this way, it is possible to understand some patterns that help to define specific actions. With this understanding, they start to solve problems logically, always considering a rational thought to determine the actions for the different daily challenges. And most of these challenges are not deterministic and one-dimensional because the world is not linear. Perhaps this is the starting point for all the issues addressed in this work.

Something that needs to be reinforced is the applicability of non-deterministic computational thinking. When working with problem-solving [23] [24], we can apply determinism or non-determinism to solve the same problem. Solutions can be deterministic or non-deterministic, not the problem. But then, why use non-deterministic solutions? Because our daily lives are full of non-determinism, like chance and probabilities. Therefore, non-deterministic computational thinking can bring us closer to reality.

But then why is there an imbalance in creation between deterministic and non-deterministic resources in computational thinking? Perhaps the answer lies in computer science itself. Deterministic programming languages are much more common than non-deterministic programming languages. However, it can be a much deeper discussion. The big issue here is uncertainty. Life is, by definition, the space of uncertainty and the security we feel is just imaginary. Despite this knowledge, when the unexpected happens, we are always disoriented and perplexed. Heraclitus of Ephesus already stated that the world and we change all the time. According to the statement, “No man ever steps in the same river twice, for it's not the same river and he's not the same man.”

For Heraclitus, nothing remains the same, and everything is continually changing. And despite this knowledge, we continue to pursue certainties. And why do we fear uncertainty? First, because biology is sovereign. Certainty brings the security claimed by our survival instinct. The uncertain is synonymous with the unknown, which brings danger and, at its limit, death. We love the comfort zone because it's the space we believe we have under our control. Immediately the following biology comes to the culture that we are rational beings and must be in charge. For Nietzsche, those who inhabit a world of certainties, of “everything under control”, “everything according to plan”, live in a world that does not exist. He calls these people nihilists, those who deny life. And life is a succession of uncertainties.

When the uncertainty becomes present in the classroom, the teacher's difficulty in managing these issues can be apparent. Teachers tend to resist what they perceive as a threat to the established way of doing things. The more intense the change, the lower the resistance tends to be. In a way, this is a positive thing, as it offers stability and predictability to behavior. On the other hand, resistance can be a source of functional conflicts, making adaptation and progress challenging. Many of the behaviors of resistance to change are based on the inexistence of emotional or motivational factors that can change the current situation and the lack of psychological security to recognize the real problems or even the advantages of alternative conditions. The fact is that educational institutions, more than ever, need to prepare and have great power to adapt to changes. If historically we are always in transition, we need to update ourselves, as educational institutions expect modern teachers to deal with conflicts, know how to work under pressure, relate, and the main thing, know-how to interact with the environment by adapting to the changes.

## 5. Opportunities

The modern world requires changes in thinking regarding education and decision-making [22]. Being a more critical, logical citizen who knows how to work in a team and solve daily problems is essential to survive in a competitive world full of challenges in personal and professional life. Given this scenario and technology in practically everything around us, computational thinking emerges as a methodology created to help children learn. They learn to develop their creative, logical, and strategic capacity to solve problems in different areas of knowledge using computational bases [23]. All this is combined with socio-emotional skills acquired through experiences, culture, and other social factors.

Programming, also currently used as a learning tool, involves contact with new languages and a new way of thinking, such as abstraction and analysis skills. When children learn to program, a new, more participative, and innovative look begins to form, and technological devices take on a new meaning in their lives [24]. Programming consists of solving problems using the computer and its data and information processing capacity as allies. In computing, programming means creating solutions – such as games, animations, and digital applications – to solve different problems. When using programming as a pedagogical practice, one of the results is the development of computational thinking. However, this practice can be unplugged in offering learning objects that consider the fundamental pillars of computational thinking.

We see it as an opportunity to develop new learning objects, a paradigm shift from "sequential" to "parallel" computational thinking [15]. Parallel computational thinking provides the student with more direct and precise development, analyzing problem situations with more abstraction than sequential thinking. However, curricula teach beginning students to think sequentially, that is, one instruction at a time. New projects for practical solutions that depend on knowledge of computer science support the need to include parallel computational thinking in the teaching and learning process in schools [14]. Therefore, breaking a big problem into small sub-problems (and using divide-and-conquer methodologies or pipelines) is part of computational thinking that is still little explored [16].

Another point consists of objects such as storytelling with different trajectories, which can change the student's choice [18]. However, typically, storytelling with continuous, linear strokes is more straightforward. The beginning is well defined, and in the middle, there is the most sublime moment, perhaps a problem or an outcome in the narrative, and finally, everything is clarified [17]. On the other hand, a more exciting story, with a certain dynamism, where the end is undetermined, the future depends on choices, paths that the author provides to the reader throughout the story. This type of narrative is called non-linear. In education, non-linear stories play a challenging role: teaching the art of writing stories builds a path to preserve memories, share knowledge, and act out emotion, action, and reaction moments. However, for the student, developing narratives is sometimes tedious in the linear aspect, unlike non-linear stories that do not follow a chronological sequence [19]. For this purpose, digital tools have been developed that help in the construction of non-linear narratives.

Finally, learning objects, including those supported by non-linear narratives, which present a probabilistic behavior in the scripts or paths to be followed, is a great challenge and a great opportunity [21]. Depending on the student's behavior, attitudes, and actions, the different paths to follow can probabilistically present which way is most appropriate to follow. Some concepts related to probability theory are used - sometimes unconsciously as to probability theory - directly in analyzing solutions present in our daily lives. Perhaps for most, concepts such as an event, the complementarity of an event, independence between events, and random variables go unnoticed or are abstracted [20]. But that is not to say that its application does not occur. And this application in new learning objects should be encouraged, as it opens up a range of possibilities where students can be active subjects in the learning process.

## 6. Conclusions

According to our certainty, our mind always works to take control of situations and plan all the steps from childhood to adulthood. The mistake lies precisely in not preparing to overcome inevitable challenges along the way. This uncertainty is something natural and must be passed on in the form of computational thinking. This new proposal brings the student closer to the uncertain daily life and allows the development of various skills driven by computational thinking.

Logical reasoning, which goes far beyond mathematics, helps to organize thinking and, in this way, conduct actions in an organized and coherent manner to solve everyday problems more efficiently. Non-deterministic computational thinking enhances the development of logical reasoning, one of the essential skills of the future. The student educated within the bases of computational thinking works with different skills, such as analyzing, interpreting, understanding relevant points of a question, and allowing the connection of other knowledge and skills. Therefore, computational thinking will allow exercising the ability to learn and apply different knowledge constantly.

Non-deterministic computational thinking also involves planning, executing, and managing tasks, something extremely relevant to school, academic, personal, and professional life. Thus, when facing a problem or developing an action, the student will be able to think beyond, taking into account its objectives, effects, and possible forms of control. All of this will guarantee you planning and management skills.

Also, the student will decompose problems, identify patterns, analyze what is or is not relevant, and establish the steps to obtain a solution, just as the software does. By using this way of thinking, children and young people can organize their thinking and solve problems that arise more efficiently and strategically. Autonomy is another skill developed with the help of non-deterministic computational thinking because this methodology allows children to stop only receiving knowledge and consuming technology and generate content and produce digital resources.

In summary, this work is the first step towards provoking the development of non-deterministic computational thinking. It seems to be this thought that allows students to approach the reality of their daily lives, allowing the solution of more realistic problems.

## 7. References

[1] Yeting Bao and Hadi Hosseini. 2021. Computational Thinking, Perception, and Confidence in Distance Learning. Proceedings of the 52nd ACM Technical Symposium on Computer Science Education. Association for Computing Machinery, New York, NY, USA, 1253.

[2] Imke de Jong and Johan Jeuring. 2020. Computational Thinking Interventions in Higher Education: A Scoping Literature Review of Interventions Used to Teach Computational Thinking. In Koli Calling '20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research (Koli Calling '20). Association for Computing Machinery, New York, NY, USA, Article 35, 1–10.

[3] Bernadette Spieler, Ferenc Kemény, Karin Landerl, Bernd Binder, and Wolfgang Slany. 2020. The learning value of game design activities: association between computational thinking and cognitive skills. In Proceedings of the 15th Workshop on Primary and Secondary Computing Education (WiPSCE '20). Association for Computing Machinery, New York, NY, USA, Article 19, 1–4.

[4] Jiang Li, Paulette Shockey, Jennifer Cuddapah, Christy Graybeal, and Anthony Williams. 2020. Introducing computational thinking to pre-service teachers. J. Comput. Sci. Coll. 36, 3 (October 2020), 174.

[5] Edmund A. Lamagna. 2015. Algorithmic thinking unplugged. J. Comput. Sci. Coll. 30, 6 (June 2015), 45–52.

[6] Peter Henderson. 2008. Computer Science unplugged. J. Comput. Sci. Coll. 23, 3 (January 2008), 168.

[7] Julian C. Wallis and Ayse Buyuktur. 2010. Information challenges in collaborative science. In Proceedings of the 73rd ASIS&T Annual Meeting on Navigating Streams in an Information Ecosystem - Volume 47 (ASIS&T '10). American Society for Information Science, USA, Article 13, 1–3.

[8] Nick Rahimi and Nancy L. Martin. 2020. Challenges and Strategies for Online Teaching in Information Technology and Other Computing Programs. In Proceedings of the 21st Annual Conference on Information Technology Education (SIGITE '20). Association for Computing Machinery, New York, NY, USA, 218–222.

[9] Djoko Sigit Sayogo, Theresa A. Pardo, and Donna Canestraro. 2011. Understanding the impact of computing and information technology on critical challenges facing 21st century financial market

regulators. In Proceedings of the 12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times (dg.o '11). Association for Computing Machinery, New York, NY, USA, 345–346.

[10] Ruhua Huang, Dan Wu, Jeonghyun Kim, and Billy Tak Hoi Leung. 2020. Data and Information Literacy Education: Methods, Models, and Challenges. In Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020 (JCDL '20). Association for Computing Machinery, New York, NY, USA, 443–444.

[11] David P. Bunde. 2019. Real examples to motivate automata theory. J. Comput. Sci. Coll. 35, 5 (October 2019), 28–36.

[12] M. Armoni, S. Rodger, M. Vardi, and R. Verma. 2006. Automata theory: its relevance to computer science students and course contents. In Proceedings of the 37th SIGCSE technical symposium on Computer science education (SIGCSE '06). Association for Computing Machinery, New York, NY, USA, 197–198.

[13] Daiki Isayama, Masaki Ishiyama, Raissa Relator, and Koichi Yamazaki. 2016. Computer Science Education for Primary and Lower Secondary School Students: Teaching the Concept of Automata. ACM Trans. Comput. Educ. 17, 1, Article 2 (January 2017), 28 pages.

[14] Santiago Ontañón, Jichen Zhu, Brian K. Smith, Bruce Char, Evan Freed, Anushay Furqan, Michael Howard, Anna Nguyen, Justin Patterson, and Josep Valls-Vargas. 2017. Designing Visual Metaphors for an Educational Game for Parallel Programming. In Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '17). Association for Computing Machinery, New York, NY, USA, 2818–2824.

[15] Keshav Pingali. 2010. Towards a science of parallel programming. In Proceedings of the 19th international conference on Parallel architectures and compilation techniques (PACT '10). Association for Computing Machinery, New York, NY, USA, 3–4.

[16] Kathy M. Lyons and Ryan T. Sharpe. 2012. Divide and conquer. ACM SIGUCCS plugged in 1, 1 (Summer 2012), 9.

[17] Dragan Mileski, Florian Schneider, and Bernd Bruegge. 2015. Visual storytelling. In Proceedings of the 18th European Conference on Pattern Languages of Program (EuroPLoP '13). Association for Computing Machinery, New York, NY, USA, Article 12, 1–7.

[18] Víctor Socas, Carina González, and Sara Caratelli. 2014. Emotional Navigation in nonlinear narratives: a case study on the influence of color. In Proceedings of the XV International Conference on

Human Computer Interaction (Interacción '14). Association for Computing Machinery, New York, NY, USA, Article 18, 1–2.

[19] Julie Porteous, João F. Ferreira, Alan Lindsay, and Marc Cavazza. 2020. Extending Narrative Planning Domains with Linguistic Resources. In Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '20). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1081–1089.

[20] Lawrence M. Leemis. 2014. Computational probability applications. In Proceedings of the 2014 Winter Simulation Conference (WSC '14). IEEE Press, 51–65.

[21] Aravind Srinivasan. 2018. Probability and Computing. SIGACT News 49, 3 (September 2018), 20–22.

[22] Min Hae Song, Jung Ae Park, and Jooyong Park. 2020. Measuring Collaborative Problem Solving Capability in Creative Problem Solving Situation. In Companion of the 2020 ACM International Conference on Supporting Group Work (GROUP '20). Association for Computing Machinery, New York, NY, USA, 103–106.

[23] Jaime L'Heureux, Deborah Boisvert, Robert Cohen, and Kamaljeet Sanghera. 2012. IT problem solving: an implementation of computational thinking in information technology. In Proceedings of the 13th annual conference on Information technology education (SIGITE '12). Association for Computing Machinery, New York, NY, USA, 183–188.

[24] Benjamin A. Ring, John Giordan, and J. Scot Ransbottom. 2008. Problem solving through programming: motivating the non-programmer. J. Comput. Sci. Coll. 23, 3 (January 2008), 61–67.