Advances in Conceptualizing, Developing, and Assessing Computational Thinking

Presenters: Mary Webb[1] (chair), Charoula Angeli[2], Christine Bescherer[3], I. Kalaš[4], Amelie Labusch[5], Yaacov J Katz[6], Peter Micheuz[7], Maciej M. Sysło[8], Joyce Malyn-Smith[9].

[1]King's College London, mary.webb@kcl.ac.uk
[2]Department of Education, University of Cyprus, cangeli@ucy.ac.cy
[3]Ludwigsburg University of Education, Germany, bescherer@ph-ludwigsburg.de
[4]Comenius University, Bratislava, Slovakia, kalas@fmph.uniba.sk
[5]Paderborn University, Germany, amelie.labusch@upb.de
[6]Bar-Ilan University and Michlalah - Jerusalem Academic College, yaacov.katz@biu.ac.il
[7]Alpen-Adria-Universität, Klagenfurt, Austria, peter.micheuz@aau.at
[8]Nicolaus Copernicus University, Toruń, Poland; syslo@ii.uni.wroc.pl
[9] Education Development Center, Inc. (EDC), Waltham, USA:jmsmith@edc.org

Keywords: computational thinking, programming, problem solving, curriculum

Introduction

Computational Thinking (CT) has garnered much interest in recent years following Wing's [1] argument that CT is an essential analytical thinking process that all students should develop alongside basic numeracy and literacy. As a consequence of this renewed focus on CT, many countries have incorporated CT into their new curricula specifications, usually as a key component of their Computer Science curriculum. However, questions remain regarding the nature, detailed definition, purpose and ways of developing CT. Webb et al. [2] argued that CT is an essential element of a Computer Science curriculum because, as indicated by the Royal Society, CT is about recognising aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand, reason and solve problems in relation to both natural and artificial systems and processes [3 p. 29]. This definition emphasises the close relationship between Computer Science and CT. CT is not dependent on programming: the more recent definition from Wing's group emphasises the thought processes in getting to the stage of formulating solutions but not necessarily implementing them: "Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent." https://www.cs.cmu.edu/~CompThink/ . Nevertheless, being able to implement the results of a CT process through programming enables students to evaluate their thinking. Therefore, Webb et al. [2] argued that the Computer Science curriculum is the best place for introducing CT, which can be used and further developed across many curriculum areas.

At the same time, many have emphasised the ubiquity of CT and its value across many disciplines and have examined ways in which computational thinking in schools can be developed without a strong underpinning of Computer Science. This approach is fueled by the identification of core sub processes of CT such as: abstraction; decomposition; pattern recognition and writing algorithms, which can easily be examined and developed without reference to Computer Science or computers. The proliferation of developments of CT has led Denning [4] to argue that the fuzzy definitions and over-claiming in relation to the widespread benefits of CT are causing problems for teachers in teaching and assessing CT. Therefore, Denning proposed returning to previous definitions, grounded in Computer Science and emphasising models of computation, e.g.: Aho [5] stated: "We consider computational thinking to be the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms. An important part of this process is finding appropriate models of computation with which to formulate the problem and derive its solutions."

Therefore, CT and its position in the school curriculum is a subject of controversy. When considering the rationale for CT in school curricula, there are several possible positions, listed below. Perhaps some compromises between these three positions may be possible but at the same time, there remain untested claims such as the scope for transferability of skills, as well as controversy over the nature of CT.

CT is an essential prerequisite to programming and being able to design and write computer programs is an essential 21st-century skill. Many disciplines require elements of computation and programming. Furthermore, being able to write programs is a useful skill for managing various aspects of an individual's daily life. Such a rationale may go further in claiming the value and transferability of the problem-solving skills involved in programming to solving problems in other subject areas and daily life more generally.

CT involves the thinking skills required for solving problems. These skills are not only useful in relation to Computing but can be applied to solving problems in a range of situations in everyday life. Therefore CT is an essential and basic life skill which can be compared to literacy and numeracy.

CT is essential to Computer Science and therefore is an essential element in the Computer Science curriculum. Basic skills and understanding of Computer Science, including CT, are necessary for all, given the important developments in relation to Computer Science and new technologies.

In this symposium, we will explore the position of CT in schools from several different perspectives. First from the perspective of young children learning through programming a robot and how elements of CT can be developed and assessed. Second through examining the basic skills and concepts of CT that are comparable with basic skills and concepts in mathematics. Third through examining some of the most basic concepts of CT and programming and how they may be introduced to young children. Fourth through examining the theoretical relationship between constructs in CT and problem-solving generally. Finally three papers examine the analysis and decision-making in Israel, Austria and USA regarding designing CT into their curricula. These papers and associated discussions will consider the following main questions

1. What is the nature of CT in computer science learning?
2. How should CT be developed?
3. How can the acquisition of CT be observed among novice learners?
4. What do we know about the interface between CT in Computer Science learning/thinking and other subject areas?
5. How does CT relate to other forms of thinking especially in STEM subjects?
6. What are the key challenges for incorporating CT in curricula?
7. What are the key challenges in developing CT more widely?

References

1. Wing, J., *Computational thinking.* Communications of the ACM, 2006. **49**(3): p. 33-36.
2. Webb, M., et al., *Computer science in K-12 school curricula of the 2lst century: Why, what and when?* Education and Information Technologies, 2017. **22**(2): p. 445-468.
3. The Royal Society, *Shut down or restart? The way forward for computing in UK schools*. 2012, The Royal Society: London.
4. Denning, P.J., *Remaining trouble spots with computational thinking.* Commun. ACM, 2017. **60**(6): p. 33-39.
5. Aho, A.V., *Computation and Computational Thinking.* The Computer Journal, 2012. **55**(7): p. 832-835.
6. Voogt, J., et al., *Computational thinking in compulsory education: Towards an agenda for research and practice.* Education and Information Technologies, 2015. **20**(4): p. 715–728.

Scaffolding Pre-Primary Education Children's Computational Thinking during Learning with the Blue-Bot

Charoula Angeli

Department of Education, University of Cyprus, cangeli@ucy.ac.cy

Abstract. The study investigated the development of children's computational thinking during learning with a Blue-Bot, and two different scaffolding techniques that provided external memory support during the educational robotics activities. According to Grover and Pea (2013), and Selby and Woollard (2013), researchers have come to accept that computational thinking, as a thought process, utilizes the elements of abstraction, generalization, decomposition, algorithmic thinking (sequencing and flow of control), and debugging. Due to the young age of the children, the focus of the current investigation was on the development of those aspects of computational thinking that were more related to the development of algorithmic thinking and debugging. It was hypothesized that external memory support systems would play a significant role in the development of children's computational thinking, because the Blue-Bot does not provide a visual representation of the commands that are used to program it, weakening this way their ability to remember and reflect on their algorithm. Within this context, the study also sought to examine gender differences and whether the two types of scaffolding differentially affected boys' and girls' computational

thinking. Data were collected for each child individually in three research sessions. During the first session, the children were first engaged in free exploration with the Blue-Bot, and subsequently in various problem-solving tasks about directionality and sequencing. During the second phase, one group of children learned with Type A scaffolding and the other with Type B. Type A involved the use of laminated command cards that children used as materials to create a physical sequence of commands. The children then used the sequence of cards to program the Blue-Bot. The children who learned with Type B were encouraged to think out loud about an algorithm while the researcher jotted down the commands mentioned. Then, the children used the researcher's notes to program the Blue-Bot. During the third session, both types of scaffolding were removed, and children were again engaged with different problem-solving activities with the Blue-Bot. The statistical analyses showed that there were no statistically significant differences in the development of computational thinking skills between boys and girls. However, there was an interaction effect between Type A and B of scaffolding and gender indicating that girls learned better using the researcher's notes, while the boys learned better with the cards. Implications for teaching computational thinking in pre-primary education will be discussed.

Keywords. Computational thinking, pre-primary education, educational robotics, scaffolding, Blue-Bot.

References

1. Grover, S., Pea, R.: Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), 38-43 (2013)
2. Selby, C., Woollard, J.: Computational thinking: The developing definitions. In Proceedings of the 45th ACM Technical Symposium on Computer Science Education. Canterbury (2013)

Computational Thinking Basic Concepts

Christine Bescherer

Ludwigsburg University of Education, Germany, bescherer@ph-ludwigsburg.de

Abstract

What is the equivalent to the basic arithmetic skills or mathematic concepts in computer science?

Nearly everybody will agree that doing 'real mathematics' without having learned the basic skills and concepts of numbers, addition, subtraction, multiplication and division in primary school would not be possible. In addition, the knowledge of these basic arithmetic skills and concepts are considered necessary for everybody (i.e. NCTM, 2000) to take part in our world. There are studies, which show the relation between i.e. basic or advanced counting skills and the mathematics achievement further grade (i.e. Nguyen et al, 2016).

If computational thinking is really one in a set of reading, writing and arithmetic (Wing, 2006), what is its equivalent of knowing the letters/numbers, spelling/counting, composing sentences/operating with numbers? What are the basic skills and concepts - and therefore the related mental models/competencies - everybody should possess to understand computational thinking or even to become a computer scientist later on?

The paper proposes to identify some of the basic CT skills and concepts from descriptions of the elements of computational thinking (i.e. Grover & Pea, 2013; Voogt et al., 2015, Weintrop et al, 2016) in analogy with identification processes used in mathematics education.

1. Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, *42*(1), 38-43.
2. NCTM - National Council of Teachers of Mathematics (2000). *Principles and standards for school mathematics.* Reston, Virginia, USA.
3. Nguyen, T., Watts, T. W., Duncan, G. J., Clements, D. H., Sarama, J. S., Wolfe, C., & Spitler, M. E. (2016). Which preschool mathematics competencies are most predictive of fifth grade achievement?. *Early childhood research quarterly*, 36, 550-560.
4. Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: towards an agenda for research and practice. Education and Information Technologies, 1–14, doi:10.1007/s10639-015-9412-6.
5. Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, *25*(1), 127-147.

6. Wing JM (2006) Computational thinking. Commun ACM 49(3): 33–35

Identifying and supporting elementary computational concepts

I. Kalaš, A. Blaho, M. Moravčík
Department of Informatics Education, Comenius University, Bratislava, Slovakia
kalas@fmph.uniba.sk; andrej.blaho@gmail.com; milan.moravcik@gmail.com

Keywords: programming, primary school, computational thinking, early computing education

Abstract

Our recent design research projects (including ScratchMaths[i] and Thomas the Clown, see Benton et al., 2017; Moravčík, Kalaš, 2012) indicate that programming can be implemented as a *key instrument for developing computational thinking* (CT), especially in its elementary (i.e. pre-primary and lower primary) stages. However, it remains unclear and debated by different schools of researchers and educators, what exactly educational programming is, what are elementary aspects and constructs of programming and what are their cognitive difficulties. Actual computing (or informatics) curricula for primary stages often list *sequence*, *selection*, and *repetition* as basic and introductory computational constructs to be addressed in primary programmes of study (see National curriculum in England). However, ScratchMaths (SM) project with its extensive and systematic interventions for grades 5 and 6 confirms that it is advisable to design other complex interventions for even lower primary grades that would engage and address several constructs and practices to be experienced earlier than *sequences* or *repetitions*.

Therefore, in our current development we set out for identifying early and elementary concepts that would adopt similar pedagogical framework to SM and provide productive and systematic pre-SM experience for pupils of grades 3 and 4 (aged 7 to 9). We have been developing a series of gradated programming environments tied together by Emil, a virtual character to firstly be directly navigated by pupils through different situations, later being programmed to solve problems in contexts built of digits (and numeracy developing tasks), letters (and literacy developing tasks), coins (and financial literacy developing tasks) etc. The final intervention for Y3 consists of 12 lessons (with other 12 lessons currently being developed for Y4), three programming environments, a workbook for pupils and a PD material for general lower primary teachers. In Y3 we focus on exploiting several powerful ideas including (i) experiencing the sense of order (in objects and in actions), (ii) coping with static and dynamic constraints, (iii) exploring and mastering control, (iv) planning solutions, (v) working with plans as objects of pupils' thinking, and (vi) extending basic 'vocabulary of actions' of Emil.

We always work with the whole class, encouraging pupils to work and learn by exploring in pairs, with frequent common discussions on the carpet. Pairs of pupils solve short sequences of (sometimes open-ended, sometimes unsolvable, often with multiple solutions) small tasks, with no feedback from the software environment. They keep "records" of their strategies on the paper worksheets which later they bring into common discussions. Even before moving to explicit programming tasks (in the sense of constructing external representations of future behaviours, see Blackwell, 2002), we noticed that in their worksheets pupils started developing their own system to record the steps of their solutions – so that they could later present their strategy to other children. We use these situations as the first and authentic steps leading to planning sequences of future moves and actions of Emil, an object to control, and identify with.

References

1. Benton, L., Hoyles, C., Kalas, I., and Noss, R. (2017). *Bridging Primary Programming and Mathematics: Some Findings of Design Research in England*. Digital Experience in Mathematics Education, Vol. 3, Springer, doi: 10.1007/s40751-017-0028-x, pp. 115-138.
2. Moravčík, M., Kalaš, I. (2012). Developing software for early childhood education. IFIP Working Conference: *Addressing educational challenges: the role of ICT*. Manchester, MMU, 2012, 12 p.
3. National curriculum in England. Computing programmes of study: key stages 1 and 2. *https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/239033/PRIMARY_national_curriculum_-_Computing.pdf*.
4. Blackwell, A.F., 2002. What is Programming? In *14th Workshop of the Psychology of Programming Interest Group*. pp. 204–218.

[1] One of the authors of this paper is a member of the ScratchMaths team

Computational Thinking and Problem-Solving – Two Sides of the Same Coin?

Amelie Labusch[1], Birgit Eickelmann[1], Mario Vennemann[1]

[1]Paderborn University, Germany,
amelie.labusch@upb.de, birgit.eickelmann@upb.de, mario.vennemann@upb.de

Abstract. While computational thinking is gaining relevance as a 21st century problem-solving competence [1], little is known about the congruence between computational thinking and general problem-solving. Since knowledge about the theoretical and empirical relationship between the two is of immense importance for the development of computational thinking as a (cross-)curricular competence, this contribution considers a national extension to the IEA-study ICILS 2018 (International Computer and Information Literacy Study) which refers in the scope of an international option to computational thinking with computer-based test modules and questionnaires for Grade 8 students, teachers, school principals and IT-coordinators [2], [3]. The German national extension, referred to in this presentation, adds to the international study by applying additional instruments towards different aspects of problem-solving.

While it is often taken for granted that this congruence will be large [4] empirical studies on the relationship between computational thinking and general problem-solving are still pending. While some studies have indicated a correlation between single components of computational thinking such as debugging and general problem-solving abilities [5] or between programming abilities and general problem-solving abilities [6], the congruence between competences in computational thinking and problem-solving is virtually unexplored, especially when it comes to using a sound empirical data basis.

Yet it is frequently assumed that computational thinking is similar to programming, a misleading reduction Voogt and colleagues [1] attribute to the situation that a number of "studies or discussions of theory [use] programming as their context" (p. 716). If computational thinking is to be taught as a cross-curricular competence, we contend that it should be something that "everyone, not just computer scientists, should be eager to learn" [7] and that it therefore also has also to be considered from a cognitive perspective [8]. In other words, computational thinking could be seen as a way of thinking and solving problems [9] but the congruence of the constructs is still to be explored. The presentation therefore elaborates on this and addresses the following research questions:

Where do intersections between computational thinking and problem-solving in general arise and how could they be modeled on a theoretical level?

Which data basis and research approach are appropriate for an empirical examination of the theoretical understanding?

What kind of results will be obtained in such a research process?

With regard to the first research question, it follows from the above that problem-solving and cognitive abilities are two areas which overlap with computational thinking and where congruence exists in problem definition, problem decomposition, and pattern recognition and matching [10], [11]. In the case of the second research question, studies to explore the overlaps between computational thinking and problem-solving need to test both areas. This approach is taken in the German national extension to ICILS 2018, which uses computer-based tests to measure the computational thinking competences of Grade 8 students and applies paper-based tests to assess the same sample's problem-solving and cognitive abilities [12]. In addition, the national extension also collects data on students' self-perceived competences in both domains. As far as research question 3 is concerned, the ICILS 2018 study will provide new insights and a more detailed understanding of how students learn computational thinking by integrating analyses on students' backgrounds as well as information on students' learning and school variables. First results will be published in 2019.

In short, this presentation aims to establish a starting point for an in-depth understanding of computational thinking from both a theoretical and an empirical perspective by researching its overlaps with a general understanding of problem-solving.

Keywords. Computational Thinking, Problem-Solving, ICILS 2018

References

1. Voogt, J., Fisser, P., Good, J., Mishra, P., Yadav, A.: Computational thinking in compulsory education: Towards an agenda for research and practice. Education and Information Technologies. 20(4), 715-728 (2015)
2. Labusch, A., Eickelmann, B.: Computational Thinking as a Key Competence – a Research Concept. In Kong, S.C., Sheldon, J., Li, K.Y. (eds.) Conference Proceedings of International Conference on Computational Thinking Education 2017. The Education University of Hong Kong, Hong Kong (2017)
3. Fraillon, J., Schulz, W., Friedman, T., Duckworth, D.: Assessment Framework of ICILS 2018. IEA, Amsterdam (2018, in press)
4. Román-González, M., Pérez-González, J.C., Jiménez-Fernandez, C.: Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. Computers in Human Behavior. 72, 678-691 (2017)
5. Liu, Z., Zhi, R., Hicks, A., Barnes, T.: Understanding problem solving behavior of 6-8 graders in a debugging game. Computer Science Education. 27(1), 1-29 (2017)

6. Lishinski, A., Yadav, A., Enbody, R., Good, J.: The influence of problem solving abilities on students' Performance on Different Assessment Tasks in CS1. SIGCSE 2016 –Proceedings of the 47th ACM Technical Symposium on Computing Science Education (2016)
7. Wing, J.M.: Computational thinking. Communications in ACM. 49(3), 33-35 (2006)
8. Labusch, A., Eickelmann, B., Vennemann, M.: Computational Thinking Processes and Their Congruence with Problem-Solving and Information Processing. In Kong, S.C., Abelson, H. (eds.) Computational Thinking Education (2018, in press)
9. Curzon, P., McOwan, P.W.: The power of computational thinking: Games, magic and puzzles to help you become a computational thinker. World Scientific Publishing Europe Ltd., London (2017)
10. Grover, S., Pea, R.: Computational Thinking in K-12: A Review of the State of the Field. Educational Researcher. 42(1), 38-43 (2013)
11. Barr, V., Stephenson, C.: Bringing computational thinking to K-11: what is involved and what is the role of the computer science education community? ACM Inroads. 2(1), 48-54 (2011)
12. Heller, K.A., Perleth, C.: KFT 4-12 + R - Kognitiver Fähigkeits-Test für 4. bis 12. Klassen [Cognitive ability test for 4th to 12th grade], Revision. Beltz, Göttingen (2000)

Computational Thinking in Israeli Elementary and Junior High Schools: From Doubt to Determination

Yaacov J Katz, Bar-Ilan University and Michlalah - Jerusalem Academic College

Email: yaacov.katz@biu.ac.il

Abstract

Since the introduction of computer technology into elementary and junior high schools in the Israeli state school system in the 1970s, there has been an ongoing controversy in Israel about the specific aims of the use of technology in education [1]. There were those in the Ministry of Education who felt that the major goal regarding the use of technology in the elementary and junior high school system primarily necessitated the mandatory mastery of computer literacy. Computer literacy was purported to ensure that all students mastered basic computer applications that would facilitate word processing, presentation of assignments in different subjects to teachers, familiarity with email and web browsers so as to ensure communication skills and access to relevant databases and other techniques designed to replace traditional hand written assignments, chalk and talk presentations, regular snail mail correspondence, and use of library reference books in the process of acquiring knowledge.

According to this traditional perception, technology served solely as an important auxiliary learning accessory limited to mastery of technological techniques that facilitated students' efficient learning. Almost no attention was paid to the possibility that mandatory study of computer science could lead to a fundamental contribution to the improvement of cognitive processes and enhancement of learning in different subject domains. When it became clear that the use of technology as an educational accessory was not significantly contributing to fundamental cognitive processes, researchers began examining the potential contribution of technology to enhanced cognitive processes that would lead to a significant improvement of student learning. The disappointing results [2] achieved by Israeli students in the international TIMSS and PISA and national Meitzav assessments served as an additional catalyst for the educational authorities to find alternative innovative possibilities that would facilitate cognitive enhancement and improved student learning.

Results of research studies have increasingly indicated the possibility that the mandatory study of computer science, which includes exposure to computational thinking skills [3] such as abstraction, algorithmic coding, analysis of data and problem-solving design, in elementary and junior high schools could potentially lead to the enhancement of cognitive processes, resulting in improved learning standards in different subject domains and contexts. Thus, computer science, may well be an educational platform that contributes to the improved actualization of cognitive potential and to improved learning achievement of students in the different subjects

Thus, in 2016, the determination of the relevant authorities in the Israel Ministry of Education, to bring about improvement in the learning achievement of Israeli students, led to the introduction of a new computer science curriculum for elementary [4] and junior high schools [5] in the hope that the new curriculum will provide an

opportunity for students to benefit from the development of the cognitive processes and enhanced learning achievement.

References

1. Katz, Y.J. (2014) The historical relationship between affective variables and ICT based learning and instruction and achievement in the Israeli school system. In A. Tatnall & W. Davey (Eds.) *Reflections on the history of computers in education: early use of computers and teaching about computing in schools*. Heidelberg: Springer-Verlag, 324-338.
2. Katz, Y.J. & Rimon, O. (2006) Numeracy in the junior high school. Jerusalem: Ministry of Education Culture & Sport. (Hebrew)
3. Webb, M.E., Davis, N., Bell, T., Katz, Y.J., Reynolds, N., Chambers, D.P. & Syslo, M.M. (2016) Computer science in K-12 school curricula of the 2lst century: why, what and when? *Education and Information Technologies, (Springer Open Access).* DOI 10.1007/s10639-016-9493-x
4. Directorate for Science and Technology (2016) Computer science and robotics curriculum for elementary schools. Jerusalem: Ministry of Education (Hebrew). Retrieved from: http://moked.education.gov.il/MafmarFiles/CS_ElementaryVer02.pdf
5. Directorate for Science and Technology (2016) Introduction to computer science in junior high schools. Jerusalem: Ministry of Education (Hebrew). Retrieved from: http://cms.education.gov.il/NR/rdonlyres/4570CA91-6BAB-4D90-B8E5-4418CBCEBF43/210149/ver30.pdf

Computational Thinking in the New Austrian National Curriculum for Lower Secondary Education

Peter Micheuz

Alpen-Adria-Universität, Institute for Informtics Didactics,
Universitätsstraße 65-67, 9020 Klagenfurt, Austria
peter.micheuz@aau.at

Abstract

"Basic Digital Education" is the name for a new subject which will be introduced in all Austrian lower secondary schools beginning with the schoolyear 2018/2019. There is one curriculum covering four years of lower secondary education (age-groups 10-14 years) and encompassing eight main topics:

- Social Aspects of Media Change and Digitization
- Information-, Data- and Media Competence
- Operating Systems and Standard Applications
- Media Design
- Digital Communication and Social Media
- Security
- Technical Problem Solving
- Computational Thinking

Obviously these topics stand for a very broad curriculum wherein at first sight Informatics (computer science) does not play a prevalent and visible role. Media pedagogy and digital literacy are apparently better represented than core Informatics in the disguise of CT (Computational Thinking). This term has not been translated into the German synonym "Informatisches Denken" and appears in the curriculum originally as the "global trademark."

All topics of the curriculum are divided into two further subtopics and detailled competence descriptions. So does the main topic CT which is split into a basic and advanced level.

Computational Thinking	Basic Level	Advanced Level
Working with Algorithms	Pupils - name and describe everyday processes - use, build and reflect codes (e.g. secret writing, QR-Code) - reproduce distinct instructions (algorithms) and carry them out - formulate distinct instructions verbally and in written form	Pupils - discover similarities and rules (patterns) in instructions (algorithms) - discover the importance of algorithms in automatic digital processes (e.g. automated proposal of potentially interesting information)
Creative Use of Programming Languages	Pupils - produce simple programs or web applications with appropriate tools in order to solve a problem or to complete tasks - know different programming languages and production processes	Pupils - master basic programming structures (decision, loops, procedures)

It is not difficult to predict that this proprietary Austrian definition of CT formulated by the curriculum committee will lead to controversial, though fruitful (inter)national discussions among didacts and especially among teachers who have to interpret and to teach this content. Moreover, it is no secret that a majority of Austrian teachers who will or have to implement this part of the curriculum within lower secondary level does not have a theoretical and, what is even more important, a practical background of CT at all. Accordingly, there will be an extraordinary requirement for professional development in CT-related pedagogy for many Austrian teachers. Other big challenges for a sound and sustainable implementation of CT are due to organizational constraints and (limited) conditions at the schools which can decide autonomously between an independent subject, an integrative approach or particular mixed forms between both.

A way of developing computational thinking through all the school years

Maciej M. Sysło

Nicolaus Copernicus University, Toruń, Poland; syslo@ii.uni.wroc.pl

Abstract. We describe here how computational thinking is developed in some pilot implementations of the national CS curriculum in Poland.

Keywords. Computational thinking, programming, problem solving

In [1] we have described how using a combination of unplugged approach and computer applications we introduce children aged 5-12 to problems such as the Konigsberg Bridge problem, classical math themes such as the Tower of Hanoi and binary numbers and concluded that children were introduced "to some concepts in computing which are certainly abstract and demonstrate that they are capable to work with abstraction and to apply computational thinking (CT)".

Moreover in the presentation [2] we addressed the question: how to motivate and engage students to learn/study/use/develop computer science competencies through 12 years in school, e.g. learning programming which, as in the case of using any "language", requires constant practice? To this end, we demonstrated "How to introduce order through 12 years" – searching and sorting appears in all known to the author national CS curricula. Lesson plans were shortly described there by specifying: problem situation to be discussed and solved, lesson main goals, CS concepts introduced, teaching and learning methods (such as abstraction, decomposition), and supporting tools (such as robots, computer applications, educational software, programming environments).

We now very shortly describe how and in what situations CT is developed through all the 12 school years as an effect of realization of our national CS curriculum.

(1) In K-3: CT does not explicitly appear as a term (e.g. in lesson plans for teachers), however children/students and teachers develop its meaning and understanding through properly chosen problems/projects (e.g. Sudoku). It is the first stage when the main focus is on modeling situations via decomposition and abstraction. Simple computer applications in a virtual environment are used.

(2) In 4-6: Solutions to problem situations are built within a programming environment, however using not only a programming language (like Scratch or Logo), but also environments like the Hour of Code or code.org in which students learn how to formulate solutions for computers using particular programming constructs. Regardless of the environment, students learn that computer programs are abstract objects of situations which are modeled.

(3) In 7-8: Environments of programming languages are used – continuation of using Scratch, but then a smooth transition to textual language like Python or C++ (for those who participate in CS contests) comes. However: the main focus is on developing algorithms not just to writing programs – students learn that programming is a tool and programs are final products of the problem-solving process. They also learn all stages of "running a program": testing, debugging, analyzing programs' behaviour for various data.

(4) High School: Students solve more complex problem situations (coming from other school subjects, like mathematics, physics, science) and develop more advanced algorithms and data structures. At this stage of CS education, we find also an opportunity to explain to students what, for instance, dynamic programming (included in the CS curriculum for HS) has in common with computer programming. Originally, in 1940-50's the term "programming" was used to denote methods for finding optimal (the best) available solutions in the precisely defined situations. Today, although learning and teaching take place in technologically different situation, writing a computer program is the last stage of problem solving, and programming supported by CT concerns the whole process of solving a problem to get the best solution, where "best" usually depends on many conditions defined and accepted during this process.

References

1. Sysło M.M., Kwiatkowska A.B., Playing with Computing at a Children's University, *WiPSCE '14*, November 05-07 2014, Berlin, Germany, 104-107.
2. Sysło M.M., Implementing computer science curriculum in primary schools in Poland – a preliminary report, presented at SaITE 2016, IFIP TC3 Joint Conference "Stakeholders and Information Technology in Education", July 6-8 2016, Guimarães, Portugal.

Framework on Computational Thinking from a Disciplinary Perspective

Joyce Malyn-Smith – Education Development Center, Inc. (EDC)

Keywords: Computational thinking, computer science education, computational thinking integration, curriculum integration.

This paper describes progress towards the development of a Framework for Computational Thinking (CT) from a Disciplinary Perspective. It shares an initial set of 5 elements to guide instruction and assessment of CT within disciplinary learning once students have developed core CS skills needed to apply CT in various contexts. The work aims to build on these core skills connecting computational thinking developed in school with the tasks often performed in highly sophisticated STEM work environments, and raises new questions about the nature of CT instruction and assessment within disciplinary classrooms. The work focused on these questions: What do computational thinking-enabled students of science or mathematics, engineering need to know? What do they need to be able to do? Do our current assessments of CT adequately measure CT within disciplinary learning? If not, what is needed?

A select group of 54 US CT researchers and practitioners reviewed literature, analyzed research findings and CT classroom practices, and organized examples of CT integration activities across grade spans (K-2, 3-5, 6-8, 9-12) and disciplines (science, mathematics, engineering, social studies, humanities, arts, etc). Analysis yielded 10 themes that appeared to describe what students were able to do with CT that they were less able to do without CT. These were reviewed against CT practices found in sophisticated STEM workplaces. Five elements emerged bridging learning and working at the human-technology frontier - Using computational thinking for:

Understanding (complex) systems

Innovating with computational representations

Designing solutions that leverage computational power/resources

Engaging in collective sense making around data, and

Understanding potential consequences of actions.

Based on the review and analysis of the current state of CT assessments against the five elements grounding this new framework, new questions arose about assessment of CT within disciplinary learning and about the role of applied computational thinking and computational sciences in disciplinary curricula.

It is hoped that this paper will provoke dialog to extend the work of educators advocating for an underpinning of CS for all - to consider how students should be applying CT within disciplinary contexts at various grade spans to build their capacity to succeed in work at the human-technology frontier.