Chapter 1
Introduction to Computational Thinking Education

Siu-Cheung Kong, Harold Abelson and Ming Lai

Abstract This chapter provides an overview of this edited volume on Computational Thinking Education (CTE). It starts with a historical review of CTE, beginning from the pioneering ideas of Seymour Papert on promoting the need to think computationally, the seminal work by Jeanette Wing, who argued that computational thinking (CT) should be an essential skill for everyone, and efforts to incorporate CT into K-12 education, such as those made by the National Research Council (Report of a workshop on the scope and nature of computational thinking, National Academies Press, 2010). With this background, the chapter introduces its conceptual framework of CTE and identifies six sub-themes. The section on the 'Computational Thinking and Tool Development' sub-theme includes an in-depth discussion of abstraction, a key concept of CT, and the development of a programming environment to facilitate CT development. 'Student Competency and Assessment' contains chapters that identify the key components, methods and tools for assessing CT. 'Computational Thinking and Programming Education in K-12' focuses on how CT can be taught and cultivated in K-12. 'Computational Thinking in K-12 STEM Education and Non-Formal Learning' discusses the combination of STEM and game activities with CT development. 'Teacher and Mentor Development in K-12 Education' sheds light on the capacity building of teachers and teaching assistants in implementing CT education. 'Computational Thinking in Educational Policy and Implementation' discusses the educational policy related to CT and a 10-year project with thinking skills embedded

S.-C. Kong (✉)
Department of Mathematics and Information Technology,
The Education University of Hong Kong, 10 Lo Ping Road, Tai Po,
N.T. Hong Kong, China
e-mail: sckong@eduhk.hk

H. Abelson
Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA
e-mail: hal@mit.edu

M. Lai
Centre for Learning, Teaching and Technology, The Education University of Hong Kong, Hong Kong, China
e-mail: mlai@eduhk.hk

S.-C. Kong and H. Abelson (eds.), *Computational Thinking Education*,
https://doi.org/10.1007/978-981-13-6528-7_1

in computer studies. Among the issues discussed in these chapters, the key focus of CTE is the importance of learning to think computationally.

Keywords Computation thinking · Computational thinking education · Computer science education · Programming education · STEM education

1.1 Introduction

This collection presents the latest research on and implementations of Computational Thinking Education (CTE). Our book includes contributions from educators and leading researchers around the world aimed at deepening the understanding of CTE theories and pedagogies, with an emphasis on education at the K-12 level. The term 'Computational Thinking Education' emphasizes the role of computing and computational ideas in facilitating learning, a perspective that is the legacy of Seymour Papert (discussed below). The term 'computational thinking' appeared as early as the nineteenth century in reference to the use of quantitative analysis in science, and appeared later regarding the emphasis on reasoning in teaching arithmetic (Childs, 2015). The modern association of the term with computers and education is due to Papert.

Since emerging from the laboratory in the 1950s, computers have become ever-present in modern life. With that growth has come increasing attention from the educational establishment, minuscule at first but building to a groundswell over the past decade. At the tertiary level, the first computer science degree programme began at the University of Cambridge in 1953, while the first programme in the US opened at Purdue University in 1962. There were 89 computer science Bachelor's degrees awarded in the US in 1966, compared to 60,000 such degrees in 2015 (National Academies of Sciences, Engineering, and Medicine, 2018).

Introducing K-12 students to CT has been slower and more sporadic. One seminal line of work stems from the invention of the BASIC programming language by John Kemeny and Thomas Kurtz at Dartmouth College in 1964. Their vision for BASIC (Beginner's All-purpose Symbolic Instruction Code) was that everyone, or at least every Dartmouth undergraduate, would learn to code. BASIC ran on terminals connected to the Dartmouth Time-Sharing system DTSS, which the university made available to Dartmouth undergraduates and to several universities and high schools. BASIC grew explosively with the introduction of the Apple II and other home computers in 1977, and it remains extremely popular with hobbyists and school computer clubs with a focus on learning to program.

A second stream in K-12 CT—and the direct precursor of present-day CTE—originated from the work of Seymour Papert and the 1967 creation of the Logo computing language by Papert, Cynthia Solomon and Wallace Feurzeig (see Solomon, 1986, for a comparative overview of BASIC, Logo and other early research in computing for K-12 learners).

Logo was created at the Cambridge research firm Bolt Beranek and Newman and then moved to the MIT Artificial Intelligence Laboratory with the start of the MIT Logo project in 1969. Papert had worked at the University of Geneva with the renowned Swiss psychologist Jean Piaget, and he brought to Logo Piaget's constructivist theory of learning, which emphasizes that children construct meaning through the interaction between experience and ideas. Papert's extension of constructivism (and wordplay on the term), which he called constructionism, holds that learning happens 'especially felicitously in a context where the learner is engaged in constructing a public entity' (Papert, 1991).

For Papert, constructing entities with the Logo computer language could provide such a felicitous context. The perspective arose that computing could be a powerful intellectual tool for all children, and that technology, as Papert wrote, could become

> …something children themselves will learn to manipulate, to extend, to apply to projects, thereby gaining a greater and more articulate mastery of the world, a sense of the power of applied knowledge and a self-confidently realistic image of themselves as intellectual agents. (Papert, 1971)

The full expression of this idea and its characterisation as 'computational thinking' first appeared in Papert's book Mindstorms (Papert, 1980), although Papert also referred to the same idea as 'procedural thinking'. He wrote:

> In this book I have clearly been arguing that procedural thinking is a powerful intellectual tool and even suggested analogizing oneself to a computer as a strategy or doing it. … The cultural assimilation of the computer presence will give rise to computer literacy. This phrase is often taken as meaning knowing how to program, or knowing about the varied uses made of computer. But true computer literacy is not just knowing how to make use of computers and computational ideas. It is knowing when it is appropriate to do so. (Papert, 1980, p. 155)

Even here, in the first articulation of the idea, there was a concern to clarify the distinction between computational (or procedural) thinking and the knowledge of how to program or to use computational tools. This concern for clarification has persisted through the growth of the CT movement and is present in several papers in the present volume.

The appearance of the personal computer in the late 1970s produced an outburst of optimism about computing's potential to play a major role in K-12 education. Apple II BASIC appeared in 1978 and Apple Pascal in 1979. MIT worked with Texas Instruments to create a Logo implementation for the TI 99/4 home computer, piloting it in a 450-student Dallas elementary school in 1980 and later in several New York City public schools. Versions of Logo for the Apple II appeared in 1982 (Abelson, 1982a, b).

While the impact of these implementations on the emerging home hobbyist computer community was significant, there was little take-up in K-12 education. School adoption was meagre, with little adherence to the vision that computation could be a powerful learning framework for everyone, not only for students working towards careers involving computer programming. As several leaders of the school computing movement observed in 2003: 'while the literature points to the potential for

impact, the reality is sobering: to a first order approximation, the impact of computing technology over the past 25 years on primary and secondary education has been essentially zero' (Norris, Sullivan, Poirot, & Soloway, 2003).

However, the adoption of computers in K-12 began to increase with the tremendous increase in the impact of information technology in society and with the emergence of computing as a continuous presence in daily life. An important catalyst for change was Jeanette Wing's (2006) seminal essay 'Computational Thinking' (Wing, 2006). Wing reintroduced the term 'computational thinking' together with the notion in Papert's tradition that CT was not just programming and that it should be a fundamental skill for everyone. The appearance of Wing's essay, contemporaneous with the start of an enormous upwelling of the computing industry, led to a surge of interest in CT and computing in K-12 education that surprised many long-time observers of computing education. Even Wing herself observed:

> "Not in my lifetime." That's what I said when I was asked whether we would ever see computer science taught in K-12. It was 2009, and I was addressing a gathering of attendees to a workshop on computational thinking convened by the National Academies. I'm happy to say that I was wrong. (Wing, 2016)

Yet even with this burgeoning interest, there remains a widespread lack of clarity about what exactly CT is, and the struggle continues to articulate its fundamentals. The report of the 2009 National Academies workshop that Wing mentions above expressed the motivation behind it:

> Various efforts have been made to introduce K-12 students to the most basic and essential computational concepts, and college curricula have tried to provide students a basis for lifelong learning of increasingly new and advanced computational concepts and technologies. At both ends of this spectrum, however, most efforts have not focused on fundamental concepts.
>
> One common approach to incorporating computation into the K-12 curriculum is to emphasize computer literacy, which generally involves using tools to create newsletters, documents, Web pages, multimedia presentations, or budgets. A second common approach is to emphasize computer programming by teaching students to program in particular programming languages such as Java or C++. A third common approach focuses on programming applications such as games, robots, and simulations.
>
> But in the view of many computer scientists, these three major approaches—although useful and arguably important—should not be confused with learning to think computationally. (National Research Council, 2010)

It is sobering that the concern to distinguish CT from programming and from the use of computer tools is the same as that expressed by Papert in Mindstorms at the genesis of the CT movement 30 years previous.

Many of the papers in this volume grapple with this same concern, and readers will find several discussions of what 'computation thinking' means in the papers that follow. Much of the 2009 NRC workshop was devoted to a discussion of this same definitional question. The workshop participants, all experts in the field, did not come to any clear agreement, nor do the authors in this volume. Yet as in the

NRC report, they agree that the cluster of ideas around CT is important in a world being increasingly transformed by information technology.

A second theme in the papers in this volume is the need to confront issues of educational computing at scale. One result of the increasing attention to CT is that jurisdictions are beginning to mandate computing education in K-12. Estonia, Australia, New Zealand, Taiwan, the United Kingdom and the US states of Virginia, Arkansas and Indiana have already taken this step, and other nations are formulating strategies to do so. This has led to serious issues of equipment availability and teacher education; several of the papers below present overviews of plans enacted or in progress. Key among the issues here is assessment, as the increasing mandates for computer learning are requiring increasing accountability from citizens and policy makers.

1.2 Conceptual Framework and Chapters in This Book

The chapters of the book were selected based on our conceptual framework of computational thinking education with six sub-themes, as illustrated in Fig. 1.1. At the top of Fig. 1.1 is 'Computational Thinking and Tool Development', the basic building block of CTE, which involves issues of the definition of CT and the design of the programming environment for facilitating CT. Students' CT development can occur in K-12 and can be combined with STEM education and non-formal learning, as captured by the sub-themes of 'Computational Thinking and Programming Education in K-12' and 'Computational Thinking in K-12 STEM Education and Non-formal Learning', respectively. To evaluate the effectiveness of students' CT development, we need to consider assessment issues, which include the articulation of the competencies involved, and the latest methods of assessing CT, as reflected in the sub-theme of 'Student Competency and Assessment'. Teacher and mentor development is a key factor to support the implementation of CTE, as captured by the sub-theme of 'Teacher and Mentor Development in K-12 Education'. From a broader perspective, policy matters can also play a supportive role in CTE, as illustrated in the sub-theme of 'Computational Thinking in Educational Policy and Implementation'. The chapters in this book were chosen according to these six sub-themes.

1.2.1 Sub-theme 1: Computational Thinking and Tool Development

The sub-theme of 'Computational Thinking and Tool Development' includes two chapters. Hoppe and Werneburg consider the abstraction involved in CT and in scientific inquiry, arguing that the former has more 'representational flexibility', and that therefore an important goal in CT education is to identify the specificity of CT

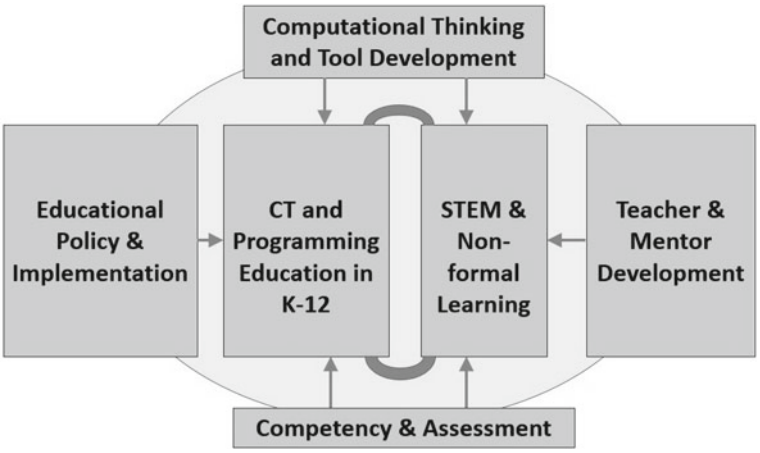

Fig. 1.1 Conceptual framework of computational thinking education

arising from its abstraction. The available abstractions, from either data representation or processing, form a repertoire of possible choices to generate computational artefacts. The programming environment supports different models of computation for users to choose from (e.g. visual programming interface), thus enabling programming flexibility. Furthermore, in terms of abstract operational mechanisms and data structure, computational media in inquiry learning contexts are of finite representational flexibility. In the other chapter, Patton, Tissenbaum and Harunani document the development of an online platform for facilitating CT, the App Inventor, at the Massachusetts Institute of Technology (MIT). They identify the logic and goals underlying the design of the platform and document how it can be used for educational purposes, focusing on empowerment through programming. These two chapters indicate the importance of the use of abstraction and a well-designed programming environment to facilitate students in learning to think computationally.

1.2.2 Sub-theme 2: Student Competency and Assessment

Among the key issues that require further exploration is how to measure students' CT ability and examine the effects of the teaching of CT, especially in terms of major assessment approaches and research methodologies. The sub-theme of 'Student Competency and Assessment' includes five chapters on related issues. Eickelmann presents a large-scale international comparative study on CT, with problem conceptualisation and solution operationalisation as the two main strands of constructs of students' achievements in CT to be assessed. The identification of these constructs echoes Papert's idea that to think computationally involves 'not just knowing how to make use of computers and computational ideas… [but] knowing when

it is appropriate to do so' (Papert, 1980, p. 155). Labusch, Eickelmann and Vennemann align CT with problem solving and information processing, reporting on the design of a cross-national study of students' processes of CT with a focus on a cognitive approach. Roman-Gonzalez, Moreno-Leon and Robles review and classify tools for assessing CT, with categories of CT diagnostic, CT summative, CT formative-iterative, CT data-mining, CT skill-transfer, CT perceptions-attitude and CT vocabulary assessment. They present the findings of two convergent validity studies conducted using a variety of tools, and they put forward a comprehensive framework that involves the chronological uses of various tools to evaluate CT. Swanson, Anton, Bain, Horn and Wilensky evaluate the effectiveness of a computational biology curriculum among ninth-grade students and find that their modelling and simulation practices, which are an important strand of CT practices, improved significantly. Kong conducts a comprehensive review of the literature and identifies the key components and methods for evaluating students' CT development based on the differentiation of CT concepts, practices and perspectives proposed by Brennan and Resnick (2012). The consideration of computational identity and programming empowerment as important components of CT perspectives particularly merits research attention.

1.2.3 Sub-theme 3: Computational Thinking and Programming Education in K-12

There are three chapters under the sub-theme of 'Computational Thinking and Programming Education in K-12'. Kong, using an example of learning prime and composite numbers through the development of an app, illustrates how learners' CT can be developed in their learning of primary mathematics, highlighting the pedagogies that can be used. Tan, Yu and Lin, who regard CT as the cultivation of logical thinking and problem-solving skills, present a study on how CT can be taught using mathematical gamification. They develop mobile games to help students develop problem-solving skills and gain mathematical insights by solving linear equations. The difficulty at each level is calibrated based on the users' performance to ensure a reasonable growing curve and prevent users from becoming frustrated at early levels. They argue that gamification can be considered an effective educational approach to gaining arithmetic proficiency and computational skills. Lee and Chan document the design of an educational website guided by a fun-based CT framework integrated with a knowledge management approach, and they discuss how the different components in this website can facilitate students' mathematics learning. These chapters illustrate how CT and programming education can be implemented in K-12 classrooms.

1.2.4 Sub-theme 4: Computational Thinking in K-12 STEM Education and Non-formal Learning

The sub-theme of 'Computational Thinking in K-12 STEM Education and Non-formal Learning' contains four chapters. Zhang and Biswas extend a framework to evaluate students' synergistic learning of CT skills and science content. With the use of a computer-based learning environment, the authors illustrate students' learning gains in CT and science concepts, and they indicate that by focusing on the synergies between STEM and CT, the difficulties that students might encounter in their simulation tasks in the learning of science content could be overcome. Keith, Sullivan and Pham use a microgenetic case study approach to investigate the roles played by two groups of girls in educational robotics activities, finding that the emergence of distinct roles is related to the time of collaboration and the time of solo work, and that it thus affects students' engagement in CT. The authors also find that in the more collaborative group, students shared their major roles, while in the less collaborative group, the roles were adopted much earlier and adhered to thereafter, leaving some group members with no chance to engage in algorithmic thinking. Ch'ng, Low, Lee, Chia and Yeong examine the correlation between video gaming experience and individual CT skills, finding a significant correlation between the former and a specific category of CT skills, abstraction and problem decomposition. This finding can help address the concern that computer games might negatively influence students' introductory programming course performances. Non-formal learning happens outside of the formal education system, and training in a company can be seen as a perfect setting for equipping targeted people with specific knowledge, such as CT. Chong and Wong, who believe that employees in the Industry 4.0 era need to be equipped with computational and logical thinking, document the experience of a textile and apparel company in which employees were empowered to solve problems and use creative ideas to improve their daily work by developing mobile apps. These works are dedicated to providing more teaching and learning practices on CT and stimulate further questions regarding how to evaluate and validate non-formal learning outcomes.

1.2.5 Sub-theme 5: Teacher and Mentor Development in K-12 Education

Three chapters are included in the sub-theme of 'Teacher and Mentor Development in K-12 Education'. Sanford and Naidu (2016) argue that 'computational thinking does not come naturally and requires training and guidance' and thus that qualified teachers for future CT education are urgently needed. Fields, Lui and Kafai identify the teaching practices that can support students' iterative design in CT, including teachers' modelling of their own CT processes and mistakes and of students' mistakes in front of the whole class. They highlight an iterative design process as a crucial

aspect of CT and the importance of revision and working through mistakes. Hsu considers the readiness of CT education from the perspective of school leaders, rating computer hardware readiness and leadership support readiness most favourably and instructor readiness and instructional resources readiness lower. Wong, Kwok, Cheung, Li and Lee discuss the supportive roles played by teaching assistants in CT classes and analyse their self-development through the process of service-oriented stress-adaption-growth.

1.2.6 Sub-theme 6: Computational Thinking in Educational Policy and Implementation

The sub-theme of 'Computational Thinking in Educational Policy and Implementation' contains two chapters. Seow, Looi, Wadhwa and Wu review the educational policy of CT in Singapore, which uses a pragmatic approach that depends on an eco-system with a focus on cultivating students' interests and allowing schools to opt in rather than making CT education compulsory. Singapore's policy offers good opportunities to educational stakeholders with interests in coding. Learning activities correspond to age and cognitive discrepancies, aiming for learning through playing in pre-school, interest cultivation in primary school and computing application in secondary school. In the final chapter, Sridhar describes a 10-year project called Computer Masti in India that embeds thinking skills such as CT into computer studies. The project involves the development of a curriculum and textbooks and the large-scale training of teachers to implement the curriculum. These chapters indicate the importance of good policies and good planning in facilitating everyone in learning to think computationally.

References

Abelson, H. (1982a). *Logo for the Apple II*. Byte Books.

Abelson, H. (1982b). *Apple Logo*. Byte Books.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *2012 Annual Meeting of the American Educational Research Association (AERA'12)*, Canada.

Childs, K. (2015). Computational thinking—The origins (part 1). *Computing & Digital Making*. Blog post, https://primaryicttech.wordpress.com/2015/12/29/the-origins-of-computational-thinking-part-1/.

National Academies of Sciences, Engineering, and Medicine (2018). *Assessing and responding to the growth of computer science undergraduate enrollments*. Washington, DC: The National Academies Press. https://doi.org/10.17226/24926.

National Research Council. (2010). *Report of a workshop on the scope and nature of computational thinking*. National Academies Press.

Norris, C., Sullivan, T., Poirot, J., & Soloway, E. (2003). No access, no use, no impact: Snapshot surveys of educational technology in K-12. *Journal of Research on Technology in Education, 36*(1), 15–27.

Papert, S. (1971). *Teaching children thinking*. MIT Artificial Intelligence Laboratory Memo no. 2247, Logo Memo no. 2. http://hdl.handle.net/1721.1/5835.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

Papert, S. (1991). Situating constructionism. In S. Papert & I. Harel (Eds.), *Constructionism* (pp. 1–11). Norwood, NJ: Ablex.

Sanford, J. F., & Naidu, J. T. (2016). Computational thinking concepts for grade school. *Contemporary Issues in Education Research, 9*(1), 23–32.

Solomon, C. (1986). *Computer environments for children: A reflection on theories of learning and education*. Cambridge, MA: MIT press.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Wing, J. M. (2016). Computational thinking, 10 years later. *Microsoft Research Blog*. https://www.microsoft.com/en-us/research/blog/computational-thinking-10-years-later/.