Hindawi
Human Behavior and Emerging Technologies
Volume 2024, Article ID 5044787, 6 pages
https://doi.org/10.1155/2024/5044787

Review Article

Computer, Computer Science, and Computational Thinking: Relationship between the Three Concepts

Pinaki Chakraborty

Department of Computer Science and Engineering, Netaji Subhas University of Technology, New Delhi 110078, India

Correspondence should be addressed to Pinaki Chakraborty; pinaki_chakraborty_163@yahoo.com

Received 28 November 2023; Revised 27 February 2024; Accepted 9 March 2024; Published 28 March 2024

Academic Editor: Mirko Duradoni

Digital computers were invented in the 1940s. They are sophisticated and versatile machines whose functioning is grounded in elaborate theory. Advances in theory and the availability of computers helped computer science to develop as an academic discipline, and university departments for the same started coming up in the 1960s. Computer science covers all phenomenon related to computers and consists primarily of man-made laws governing building, programming, and using computers. Computational thinking is a way of thinking influenced by computers and computer science. There are two schools of thought on computational thinking. The first school sees computational thinking as the use of computers to explore the world, while the other sees computational thinking as the application of concepts from computer science to solve real-world problems. Scholars typically agree that computational thinking has four essential components, *viz*., abstraction, decomposition, algorithm design, and generalization. Computational thinking is often feted by computer scientists as a useful skill that can be used by anybody anywhere. However, it is necessary to find out ways for successfully using computational thinking in domains other than computer science before it can be declared a universal skill.

1. Introduction

Computers distinguish themselves from other man-made devices by the level of sophistication in their design, their wide range of uses, and the theoretical foundation of their working. Computers have been evolving steadily for the last eight decades. Advances in computers have been complemented by developments in computer science which is perhaps the only academic discipline focusing on a single man-made artefact. Computers [1] and computer science [2] can influence how people associated with them think and work. However, not many scholars have commented upon the interplay of computers, computer science, and computational thinking. This paper analyzes the relationship between computers and computer science and how they together influence the human thought process today. The paper is a narrative review on the influence of computers and computer science on human thought processes written with an emphasis on the most important developments in the field in the last eight decades. Care has been taken to showcase the rich history of the topic in this paper.

2. Digital Computer

A digital computer is a machine that can execute programs. A program is a sequence of instructions presented in a predetermined binary format. A program can be used to find the solution to a problem which is difficult to solve manually. Such programs can be written to solve problems in different fields of human endeavor. These make computers powerful and perhaps the most versatile machines in the world. (The term "computer" is now typically used to denote a digital computer. However, the term was used in the past for a person with expertise in performing mathematical calculations, i.e., a "human computer." A digital computer is also different from an "analog computer" which is a device

performing specific types of calculations using analog signals representing various physical quantities.) The first computers were built in the 1940s, e.g., Z3 and Z4 computers in Germany and ENIAC computer in the USA.

The computer is the Proteus of machines. (Seymour A. Papert [1] *(p. viii))*

(In Greek mythology, Proteus is the god of the sea and characterizes its constantly changing nature.)

Since the late 1940s, almost all computers follow a model popularized by von Neumann [3]. According to this model, known as the von Neumann architecture, a computer has a memory unit which stores programs and data. The processing unit of the computer fetches one instruction from the memory, decodes it, and executes it. Executing an instruction may require reading some data from the memory and/or writing some data in the memory. After executing an instruction, the processing unit repeats these steps with the next instruction in the sequence or an instruction elsewhere in the program as dictated by the logic of the program. Since the programs are stored within the computer, such a computer is known as a stored-program computer. This technique of executing programs in which instructions are fetched, decoded, and executed one by one is known as the instruction cycle. Instruction cycle is a characteristic feature of computers. Hence, any machine capable of implementing instruction cycle is a computer. Commonly used computers include desktop computers, laptops, tablets, smartphones, and smartwatches. Dijkstra remarked that computers are "usually electronic" machines which can store large volumes of data and process them at a high speed with a high reliability [4]. (Although most computers built so far use electronic components, it is possible to build computers without any electronic part.)

On the theory front, Turing proposed a theoretical model of computation which could be used to solve a large class of computational problems [5]. The model later became known as Turing machine. (Turing [5] used the terms "automatic machine" and "universal computing machine" to denote the Turing machine and universal Turing machine, respectively. The term "Turing machine" was first used in a review of Turing's work by his doctoral advisor Alonzo Church [6].) Turing also defined a universal Turing machine which is a theoretical model of computation that can emulate an arbitrary Turing machine [5]. In other words, a Turing machine can solve a particular computational problem while a universal Turing machine can solve all computational problems solvable by the Turing machines. A device is called Turing complete if it can emulate all Turing machines; *i.e.*, it is as powerful as a universal Turing machine. Computers are Turing complete. Further, no theoretical model or physical device is known to be able to solve any computational problem that cannot be solved by a Turing machine. Therefore, computers can solve all computational problems which can be solved mechanically.

3. Computer Science

Scholars have provided several interesting definitions of computer science. However, the definition by Newell et al. [7] quoted below is certainly the most succinct one. Newell et al. [7] viewed computer science as the study of phenomena surrounding computers. Following the same line of thought, Denning defined computer science as the science of information processes and their interaction with the world [8]. There is an emphasis on the role played by computation in the real world in the definitions of Newell et al. [7] and Denning [8]. Alternatively, definitions by Knuth [9] and Wing [2] are more procedure oriented. Knuth defined computer science as the study of algorithms [9], while Wing saw computer science as the study of computations which primarily dealt with determining what can be computed and how they can be computed [2].

...computer science is the study of computers. (*Newell et al.* [7])

Among all phenomena related to computers, algorithms are perhaps the most important [9]. An algorithm is a finite sequence of steps that can be followed to transform the given input information to the desired output information. A programmer designs algorithms intended for mechanical execution [4] and writes a program according to the algorithm in a programming language. A program is actually a particular way of representing an algorithm [9]. Theoretically, an algorithm for a function f is equivalent to a Turing machine that computes f [10]. Programming languages are models of computation equivalent to a universal Turing machine and hence can be used to implement various algorithms. Programmers however prefer writing algorithms intuitively rather than as Turing machine functions. Dijkstra felt that programming is a type of mathematical activity and such activities are characterized by high precision, wide generalizability, and high confidence level [4]. Mathematics and computer science have man-made laws which can be proved, instead of natural laws which have to be discovered but could never be known with certainty [9]. Computer science involves systematic study of algorithmic processes that describe and transform information [11]. (Denning et al. [11] used the term "computing." Knuth [9] discussed the terms used to denote computer science in several (Western) languages and countries and tacitly accepted their equivalence.) Appropriate algorithms are necessary for mechanical realization of information processes, and according to Denning et al. [11], the fundamental question underlying all computational tasks is "what can be (effectively) automated?"

Computer science was born as an academic discipline with the joining together of mathematical logic and algorithmic theory and the invention of stored-program computers in the mid-1940s [11]. By the early 1960s, there was a sufficient body of knowledge to justify starting academic programs and even establishing university departments for computer science [12]. (The Diploma in Numerical Analysis and Automatic Computing of the University of Cambridge started in 1953 was perhaps the first academic program in computer science, while Purdue University established a computer science department in 1962 and was perhaps the first university to do so.) Once computer science was established as an academic discipline, universities played a crucial role in expanding its body of knowledge and disseminating the same among people. Computers are novel and complex equipment [7] which can act as universal computing

machines [12]. In other words, computers are tools to implement, study, and predict information processes [8]. As a result, computers lie at the center of computer science, and much of the body of knowledge in the discipline is concerned about computers and phenomena surrounding them [12]. Programmers design algorithms and write programs for existing and conceivable computers [4]. (Lovelace [13] and Shor [14] designed algorithms for a mechanical computer and a quantum computer, respectively, which were conceivable but did not exist at their respective time.) Computer science is however more than just programming [2]. Computer science covers the underlying principles of computation, development of working computing systems with hardware and software components, and how people interact with them.

4. Computational Thinking

Papert is believed to have coined the term "computational thinking," and he used it in his book [1]. (Papert [1] used the word "computational" as an adjective to qualify things which were related to computers in some way. The word was used 51 times in the book including the index. The term "computational thinking" appeared only once in the book (p. 182) and was not listed in the index. Nevertheless, concepts pertaining to computational thinking were discussed at various places in the book.) Papert recognized the potential of computers in the field of education and, in the 1960s, started developing computer-based tools and techniques for school-aged children. He shared his experience on this work in his book. Papert [1] appreciated computers for their utility and versatility. He predicted that the world will be "computer-rich" in the future (p. 3). He believed that computers are powerful tools which can enable children to explore a wide range of topics. His team developed a device which they named "turtle." (Papert [1] went on to call the turtle a "computer-controlled cybernetic animal" (p. 11).) The turtle was connected to a computer and could plot lines and curves on a flat surface. The turtle's movement and plotting activities were controlled by programs written in a purpose-designed programming language called LOGO. LOGO was a simple language, and children could write programs in it intuitively. Papert described how children explored concepts of geometry and physics by writing LOGO programs [1]. Papert felt writing programs helped children in two ways; viz., they attained mastery over computers, and they could connect with various concepts of science, mathematics, and arts [1]. (Babbage [15] predicted that computers will influence the course of science in the future (p. 137).) Papert [1] believed that using "computer as pencil" will help children explore and learn new concepts (p. 210). Years later, Papert reiterated that digital technology is a liberator and can help in implementing new concepts in education [16]. He felt that computers can help children to think of new ideas and even realize some of them [17]. Papert's philosophy was that children can learn computer programming and learning to program influences how they learn other things [17]. It can be inferred that for Papert, computational thinking was thinking influenced by computers, thinking in terms of computers, and thinking to use computers as tools in various activities.

Papert's work influenced scholars from many disciplines for years, but computational thinking did not receive extraordinary attention. Things changed suddenly when Wing wrote about her views on computational thinking and the perceived benefits of the same in an essay [2]. (It is difficult to say if Wing was influenced by Papert's view on computational thinking. Wing [2] did not cite any literature because the format of the article did not allow her to do so. Wing [17] cited some papers but none by Papert.) Wing described computational thinking as a universally applicable attitude and skill set that builds upon the concepts of computer science [2]. She later elaborated her point by stating that computational thinking is taking an approach for solving problems, designing systems, and understanding human behavior using concepts that are fundamental to computer science [18]. According to Wing [2], computational thinking allows finding approximate solutions problems, reducing a problem into another, and solving problems using recursion and parallel processing. Computational thinking is a way of thinking for human beings and not computers and can be used by everyone everywhere. Computational thinking is about conceptualization and not programming, and it is an idea and not an artefact. Wing claimed that computational thinking is a fundamental skill for everyone and not just for computer scientists [2]. She went on to say that computational thinking should be taught to every child along with reading, writing, and arithmetic. Further, Wing remarked that computational thinking will influence everyone in every field of endeavor [18]. Wing's work brought computational thinking in the spotlight, and researchers from multiple disciplines started investigating its utility. Computational thinking lessons were introduced in some schools as well.

Papert mentioned computational thinking in context of an alternative and computer-assisted approach for teaching mathematics and other disciplines to children [19]. Papert recommended the use of computer as a tool for learners to explore various topics. For example, writing simple programs can be an effective way to understand Euclidean geometry. Papert's version of computational thinking is concerned about forging ideas and implementing and explaining them. Alternatively, Wing saw computational thinking as an effective way of solving problems in the daily life. For example, one can use the concept of topological sorting to schedule the activities for a busy day. Wing's version of computational thinking focuses on abstraction, analysis, and automation. What Papert and Wing had in common was their emphasis on reasoning, and both of them indicated that computational thinking is more than just using computers to solve particular problems [20].

Computational thinking is a way of thinking [20] and acting [21] in which problems are formulated in such a way that their solutions can be represented as a series of computational steps [10]. The solutions may be implemented with or without the assistance of computers and should be reusable in different contexts [21]. Alan J. Perlis used to call the quantitative analysis of the way one does things "algorithmizing" and recommended that computers

be treated as general tools to facilitate reasoning rather than devices to solve specific problems [22]. Computational thinking is a type of analytical thinking and has similarities with mathematical, scientific, and engineering thinking [18]. Analytical thinking is not unique to any discipline [23], and computational thinking cannot lay claim on very broad ways of thinking [24]. Computational thinking is an important element of computer science [23]. However, it does not represent the entire discipline of computer science [23] and is certainly not an independent discipline [25].

Since Wing [2] published her essay, researchers started investigating what all consist computational thinking and several scholars like Selby and Wollard [26], Shute et al. [21], Lodi and Martini [25], and Yadav and Chakraborty [27] came up with their lists of components that comprise computational thinking. There is consensus among scholars about four components which are considered to be fundamental to computational thinking as listed below.

(i) *Abstraction*. A problem should be modeled in a way that highlights the computational aspects and obfuscates the rudimentary aspects of the problem space. Such an abstraction is a way of understanding the problem [28] and is not bound by algebraic properties or dimensions of the physical world [18]

(ii) *Decomposition*. A complex problem is divided into simpler subproblems. The subproblems are solved separately, and their solutions are combined to obtain the solution of the initial problem. Decomposition is often realized using divide-and-conquer, *i.e.*, a problem is divided into subproblems that are similar in nature, or recursion, *i.e.*, a problem is reformulated in terms of smaller instances of the same problem

(iii) *Algorithm Design*. A basic skill in computer science is to formulate a sequence of computational steps which if followed generates a solution to a given problem [12]. The sequence of steps thus formulated applies mathematical logic to obtain efficient, fair, and secure solutions [28]

(iv) *Generalization*. Computational thinking is concerned about the application of concepts drawn from computer science to solve various real-world problems. Computational thinking will be more effective if the procedure followed to solve a problem can be reused to solve another problem in some other domain [21]

Computer programming is a part of the standard practices in computer science [11], and writing computer programs influences the way one thinks. (Computer programming began in earnest when six young female mathematics majors, viz., Betty Jennings, Betty Snyder, Fran Bilas, Kay McNulty, Marlyn Meltzer, and Ruth Lichterman, were hired to operate ENIAC. Their work, although remained unnoticed for years, proved that programming a computer is not same as using an arbitrary apparatus but needs training, practice, and thinking.) Computer programming helps people to acquire skills necessary for thinking rigorously and effectively expressing the thought process [29]. Lewis [30] argued that it is reasonable to assume that engaging in intellectually demanding tasks facilitates cognitive development in people, and if this argument is correct, then computer programming is an activity which can certainly help people to enhance their ability to think and reason. However, the computer science community, including accomplished computer scientists like Seymour A. Papert and Mitchel Resnick, is understandably biased when it comes to reflect on the utility of computer programming and computational thinking in the real world. (It may be noted that Papert and Resnick developed influential programming languages. Papert and his team developed LOGO as a programming language to facilitate children to explore mathematics and science. Alternatively, Resnick and his team developed Scratch as a language to introduce computer programming to learners. A comparison of these two types of programming languages has been provided by Papert [1] using the examples of LOGO and BASIC programming languages (p. 29).) For example, Papert [1] tried to show how engaging in computer programming can help children to think computationally and explore the world (pp. 28, 98, 105, 114, 154, 176). Similarly, Resnick et al. [31] claimed that computer programming is necessary to design, create, and invent using digital technologies, and Brennan and Resnick [32] claimed that indulging in computer programming improves computational thinking skills. Unfortunately, there is little evidence to prove that proficiency in computer programming can help a person in nonprogramming endeavors [33] and the claim that computational thinking can easily be transferred to other disciplines is largely unsubstantiated [25]. Every academic discipline has its own set of concepts and techniques. Nevertheless, there are some skills, such as fundamental mathematics and basic language skills, which find application across academic disciplines and professions. There is a dearth of evidence to show that computational thinking can be considered in that category. Consequently, discussions on computational thinking should be based on the principles of computer science rather than skills of computer programming [34]. One of the key aspects of computational thinking is explainability, and it can be helpful in upholding ethical considerations in many domains. Nevertheless, algorithm design bias needs to be taken care of while researching computational thinking. In computational thinking, one is expected to be able to unambiguously explain the way a problem is analyzed and a solution is designed. This facilitates communication and collaboration.

5. Conclusion

Computers have now become omnipresent. A vast majority of the world population today uses computers of some type for performing some of their daily life tasks. Scientists, engineers, and professionals in other fields use computers to perform highly specialized work. Computers also provide employment to many people in developed as well as developing countries. In the last eighty years, computer science

has evolved a lot, perhaps more than any other academic discipline. Computer science has helped in developing computers with increasing sophistication and extending their use for the benefit of a growing number of people.

Every academic discipline deals with a large body of knowledge which is continuously refined by scholars. Scholars from all disciplines engage in critical thinking, and thinking cannot be considered a monopoly of any particular discipline. Hemmendinger [24] felt that computer scientists often present computational thinking in terms which can be interpreted as arrogant or overstatement by others. Computational thinking can be considered a universally valuable skill only if people other than computer scientists use it and find it beneficial. This paper only discusses how thinking can be influenced by computers and computer science. The influence of other academic disciplines like mathematics and philosophy on thinking is also important and needs to be researched.

As of now, there are three important reasons to teach computational thinking to learners of various ages. First, the nature of science and mathematics is changing rapidly [35], and they are becoming increasingly computational as Babbage and Papert envisioned long ago. Possessing computational thinking skills may help people in understanding science and mathematics better. Second, concrete thinking is present in five-year-old children, while children develop formal thinking skills by twelve years of age. Using computers and engaging in computational thinking can help children to develop formal thinking skills at a younger age [25]. Third, children who receive lessons in computational thinking may choose to study computer science when they grow up not only for career opportunities but also for its intellectual content [25]. Yadav and Chakraborty [27] hypothesized that incorporating computational thinking lessons at the K-12 level can encourage children to enroll in engineering programs in the future. Although Wing's essay [2] and later research on the topic led to the introduction of lessons in computational thinking in several schools around the world, there are many challenges in teaching computational thinking such as lack of suitable textbooks, tools and techniques, assessment methods, and trained teachers.

Data Availability

No primary data was used in this study.

Conflicts of Interest

The author declares that he has no conflicts of interest.

References

[1] S. Papert, *Children, Computers and Powerful Ideas*, Basic Books, 1980.

[2] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006.

[3] J. von Neumann, *First Draft of a Report on the EDVAC*, University of Pennsylvania, 1945.

[4] E. W. Dijkstra, "Programming as a discipline of mathematical nature," *The American Mathematical Monthly*, vol. 81, no. 6, pp. 608–612, 1974.

[5] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society*, vol. s2-42, pp. 230–265, 1937.

[6] A. Church, "Reviews: A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, 2 s. vol. 42 (1936-7), pp. 230–265," *The Journal of Symbolic Logic*, vol. 2, no. 1, pp. 42-43, 1937.

[7] A. Newell, A. J. Perlis, and H. A. Simon, "Computer science," *Science*, vol. 157, no. 3795, pp. 1373-1374, 1967.

[8] P. J. Denning, "Is computer science science?," *Communications of the ACM*, vol. 48, no. 4, pp. 27–31, 2005.

[9] D. E. Knuth, "Computer science and its relation to mathematics," *The American Mathematical Monthly*, vol. 81, no. 4, pp. 323–343, 1974.

[10] A. V. Aho, "Computation and computational thinking," *The Computer Journal*, vol. 55, no. 7, pp. 832–835, 2012.

[11] P. J. Denning, D. E. Comer, D. Gries et al., "Computing as a discipline," *Computer*, vol. 22, no. 2, pp. 63–70, 1989.

[12] P. J. Denning, "Computer science," in *Encyclopedia of Computer Science*, A. Ralston, E. D. Reilly, and D. Hemmendinger, Eds., pp. 405–419, Nature Publishing Group, 2000.

[13] A. A. Lovelace, "Sketch of the analytical engine invented by Charles Babbage, Esq. by L. F. Menabrea, of Turin, Officer of the Military Engineers," in *Scientific Memoirs Selected from the Transactions of Foreign Academies of Science and Learned Societies and from Foreign Journals, Vol. III*, R. Taylor, Ed., pp. 666–731, Richard and John E. Taylor, 1843.

[14] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, Santa Fe, NM, USA, 1994.

[15] C. Babbage, *Passages from the Life of a Philosopher*, Longman, Green, Longman, Roberts and Green, 1864.

[16] S. Papert, "Keynote lecture," in *ICMI Study 17 Conference*, Hanoi, Vietnam, 2006.

[17] S. Papert, "What's the big idea? Toward a pedagogy of idea power," *IBM Systems Journal*, vol. 39, no. 3.4, pp. 720–729, 2000.

[18] J. M. Wing, "Computational thinking and thinking about computing," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 366, no. 1881, pp. 3717–3725, 2008.

[19] A. Juškevičienė and V. Dagienė, "Computational thinking relationship with digital competence," *Informatics in Education*, vol. 17, no. 2, pp. 265–284, 2018.

[20] L. Mannila, V. Dagiene, B. Demo et al., "Computational thinking in K-9 education," in *Proceedings of the Working Group Reports of the Innovation & Technology in Computer Science Education Conference*, pp. 1–29, Uppsala, Sweden, 2014.

[21] V. J. Shute, C. Sun, and J. Asbell-Clarke, "Demystifying computational thinking," *Educational Research Review*, vol. 22, pp. 142–158, 2017.

[22] D. L. Katz, Ed., "Conference report on the use of computers in engineering classroom instruction," *Communications of the ACM*, vol. 3, no. 10, pp. 522–527, 1960.

[23] P. J. Denning, "The profession of IT–beyond computational thinking," *Communications of the ACM*, vol. 52, no. 6, pp. 28–30, 2009.

[24] D. Hemmendinger, "A plea for modesty," *ACM Inroads*, vol. 1, no. 2, pp. 4–7, 2010.

[25] M. Lodi and S. Martini, "Computational thinking, between Papert and Wing," *Science & Education*, vol. 30, no. 4, pp. 883–908, 2021.

[26] C. C. Selby and J. Wollard, "Computational thinking: the developing definition," 2010, https://eprints.soton.ac.uk/356481/.

[27] S. Yadav and P. Chakraborty, "Introducing schoolchildren to computational thinking using smartphone apps: a way to encourage enrollment in engineering education," *Computer Applications in Engineering Education*, vol. 31, no. 4, pp. 831–849, 2023.

[28] J. J. Lu and G. H. Fletcher, "Thinking about computational thinking," in *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, pp. 260–264, Chattanooga, TN, USA, 2009.

[29] W. Feurzeig, S. Papert, M. Bloom, R. Grant, and C. Solomon, "Programming-languages as a conceptual framework for teaching mathematics," *ACM SIGCUE Outlook*, vol. 4, no. 2, pp. 13–17, 1970.

[30] C. M. Lewis, "Good (and bad) reasons to teach all students computer science," in *New Directions for Computing Education: Embedding Computing across Disciplines*, S. Fee, A. Holland-Minkley, and T. Lombardi, Eds., pp. 15–34, Springer, 2017.

[31] M. Resnick, J. Maloney, A. Monroy-Hernández et al., "Scratch," *Communications of the ACM*, vol. 52, no. 11, pp. 60–67, 2009.

[32] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," in *Proceedings of the 2012 annual meeting of the American educational research association*, Vancouver, Canada, 2012.

[33] R. Scherer, "Learning from the Past-The need for empirical evidence on the transfer effects of computer programming skills," *Frontiers in Psychology*, vol. 7, p. 1390, 2016.

[34] M. M. Sysło and A. B. Kwiatkowska, "Informatics for All High School Students," in *Informatics in Schools. Sustainable Informatics Education for Pupils of all Ages. ISSEP 2013*, I. Diethelm and R. T. Mittermeir, Eds., vol. 7780 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2013.

[35] D. Weintrop, E. Beheshti, M. Horn et al., "Defining computational thinking for mathematics and science classrooms," *Journal of Science Education and Technology*, vol. 25, no. 1, pp. 127–147, 2016.