
A BASIC SUMMARY TOWARDS COMPUTER SCIENCE: AN ATTEMPT TO GENERATE INFORMATION'S FOR THE PRACTICALITY TO READERS

Shah Md. Omer Farque Jubaer

Muhammd Nyeem Hassan

Syed Hasan Shahriar Rofi

Shuvrangshu Roy

Mohammad Kawsar Ahmed

Abstract

Computer science teaches students how to strengthen their computational and critical thinking skills as well as how to build rather than just use new technology. This foundational information is required for students to be prepared for the twenty-first century, regardless of their eventual field of study or career. In the same way that a physics course teaches essential principles about the laws of motion and energy, foundational computer science courses in K–12 teach the core concepts of computing. The new AP computer science course is being developed around seven broad ideas at the foundation of computer science: creativity, abstraction, data, algorithms, programming, the Internet, and effect. These ideas are important to computer science, but they can be applied to analysis in many disciplines. It is crucial to understand the fundamentals of computer science since it allows farmers all over the world to boost their food output. The food supply system is critical when you live on a world with roughly 8 billion people. Not to mention that analysts predict that by 2050, the world's population would have grown to 9.7 billion people. Computers assist farmers in a variety of ways. As a result, the primary goal of this research article is to identify and discuss the fundamentals of computer science.

Keywords: Computer Science, Automata Theory, Areas and extension of the computer science, Computation complexity theory.

Introduction

Problem solvers are computer scientists. When a computer scientist receives a challenge, they gather data and use programming languages and logic to connect with computers (e.g. scripting). They write a set of instructions for the computer to follow in order to solve the problem. Continue reading to learn more about computer science, its importance and benefits, professional tools and resources, and how to pursue a computer science career. Computer scientists work on issues including anticipating natural disasters, charting virus epidemic patterns, enhancing our health-care system,

and making education more accessible. People used to laugh at internet education, but nowadays, anyone can study practically anything online, from YouTube lessons to free learning sites like Khan Academy. Today, computer science continues to push boundaries. On a daily basis, wearable electronic devices, self-driving automobiles, and video communications affect our lives. The history of computer science is crucial for understanding today's advancements. We landed a human on the moon, connected the world with the internet, and put a portable computing device in the hands of six billion people thanks to computer science. George Forsythe coined the term "computer science" in 1961. Programming theory, data processing, numerical analysis, and computer system design are all terms used by Forsythe to describe the discipline. The first university computer science department was formed only a year later. Forsythe went on to build Stanford's computer science department (Computer science: An overview) .

The concept of computer Science: Computer science, abbreviated as CS or CompSci, is the study of computers, how they work, and the fundamentals of computer programming. A computer science degree holder has a thorough understanding of computer software and hardware, as well as proficiency in one or more programming languages. However, it is simple to state that computer science is the study of computers, including computational theory, hardware and software design, algorithms, and human-computer interaction. Information, protocols, and algorithms for idealized and real automata are studied in computer science (Computer science concept inventories: past and future)

1. Automaton: "self-moving" in our case, a self-determining or autonomous system with finite resources (time and space).
2. Information: knowledge represented in a way that can be transmitted, manipulated, and so on.
3. Protocol: a set of rules for efficiently sharing information.
4. Algorithm: a clear, finite explanation of a process in simple steps or activities.

The philosophy of computer science and extension: Computer science philosophy is concerned with the ontological and methodological concerns that arise both inside the academic discipline of computer science and in the practice of software development and its commercial and industrial deployment(Philosophy and computer science) . More specifically, computer science philosophy studies the ontology and epistemology of computational systems, with an emphasis on issues such as definition, programming, implementation, verification, and testing (Thinking machines and the philosophy of computer science: concepts and principles) . Because of the complexity of computer programs, many of the philosophical issues presented by computer science have parallels in philosophy of mathematics, philosophy of empirical sciences, and philosophy of technology. Computers and networks are becoming more prevalent in all parts of our daily lives, including business, manufacturing, education, and health care(Thinking machines and the philosophy of computer science: concepts and principles) . This program covers technological fundamentals, computer hardware,

computer software, and networking technology in a balanced way. The focus is on hardware and software operating principles, networking models, operating systems, and industry standards, as well as hands-on lab activities for building practical problem-solving abilities (The Blackwell guide to the philosophy of computing and information) . The capacity to configure and troubleshoot basic PCs, local area networks (LANs), and fundamental information technology is developed in students. Courses to prepare students for the CompTIA A+ and MCITP exams are included in the program (Undergraduate computer science education: a new curriculum philosophy & overview) . The study of computer science support to get few common grounds:

1. Problem-solving using algorithms.
2. Computing and analysis of data (managing, processing, visualizing and interpreting data).
3. Interaction between humans and computers.
4. Creating models and simulations of real-world issues.
5. Manipulation and creation of visuals
6. Computer programming (including game design).
7. Safety and security (including cryptography).
8. Developing a website (illustrating principles of programming, human-computer interaction and abstraction).
9. Robotics is number nine (designing and programming).
10. Computer ethics and social issues(Undergraduate computer science education: a new curriculum philosophy & overview)

In today's world, computational systems are everywhere(Optimization of linear control systems: analytical methods and computational algorithms) . The discipline of computer science is responsible for their design, development, and analysis. Instead, they are treated as theoretical objects in computer science philosophy. Its initial goal is to define such systems, i.e. to create a computational systems ontology. There are two primary approaches to the problem in the literature. The first interprets computational systems in terms of discrete ontologies for software and hardware, which are widely considered to be their fundamental components. A different perspective sees computational systems as containing several other elements around the software-hardware divide: in this view, computational systems are defined using a hierarchy of levels of abstraction, with hardware levels at the bottom and extending upwards to design elements and downwards to include the user. These two approaches are presented in the following sections (Computer science: The discipline) .

Learning factors in studying computer science: Computer science teaches students how to strengthen their computational and critical thinking skills as well as how to build rather than just use new technology. This foundational information is required for students to be prepared for the twenty-first century, regardless of their eventual field of study or career. Information technology is pervading many parts of daily life in the twenty-first century, with big data, software, and the Internet being integrated

into businesses and goods all over the world. Students who study computer science get information and abilities that can be used to a range of fields. Here are several examples:

1. Designing security software and hardware systems or building mobile communication devices, networks, and applications in the field of information technology.
2. In manufacturing, product design and simulations are used to improve products. What is computer science and what do people do with it once they've mastered it?
3. In healthcare, analyzing the massive amounts of data generated by new DNA sequencing techniques, inventing new patient remote monitoring systems, and designing security and privacy for medical records are all possibilities.
4. Using data to predict trends and enhance inventory management in retail.
5. in hurricane forecasting, building and analyzing models that predict hurricane activity.
6. In the arts, creating new special effects for movies or digital music composition.
7. Designing and managing automated trading systems in the financial services industry.
8. Learning computer science: Perceptions, actions and roles .

Computation theory and extensive construction: The transfer and transformation of data that occurs during the transition of data or the processing of data based on a set of processes is referred to as computation. The fundamental mathematical features of computer hardware, software, and applications are included in computation theory. It's a branch of computer science that deals with how to solve a problem quickly by applying an algorithm to a computation model. The theory of computation field is divided into three concepts, which are as follows –

1. Automated theory and language.
2. Computability theory.
3. Complexity theory (Handbook of computational group theory) .

Automata theory and Language: Automata theory's modern uses go well beyond compiler algorithms and hardware verification. Automata are frequently used in software, distributed systems, real-time systems, and structured data modeling and verification. They also contain qualities that allow them to model time and probabilities. Automata theory is the study of abstract machines and automata, as well as the computational problems that can be solved with them. It is a computer science theoretical theory. The word "automata" comes from the Greek word "o," which implies "self-acting, self-willed, and self-moving." An automaton (plural: automata) is a self-propelled computing device that follows a predetermined set of instructions. An automaton with a finite number of states is known as a Finite Automaton (FA) or Finite-State Machine (FSM). A finite-state machine, as seen in the figure to the right, is a well-known type of automata. This automaton is made up of states (shown by circles in the diagram) and transitions (represented by arrows) (represented by arrows)(Introduction to automata theory, languages, and computation) .

It is rarely seen in the software engineering industry, at least not in the way I see it on a regular basis. However, automata could represent finite state machines. They're useful for programming lifts, turnstiles, washers, and robot movements, among other things. While formula complexity is important from time to time, I don't believe it is necessary when it comes to complexity class. Usually, only $O(n)$ and $O(n^2)$ are used (An introduction to automata theory. Blackwell Scientific Publications).

When the automaton detects an input symbol, it uses its transition function, which takes the previous state and the current input symbol as parameters, to transition (or jump) to a different state. Automata theory and formal language theory are closely related. In this context, automata are used as finite representations of formal languages that may be endless. The Chomsky hierarchy, which explains a nested relationship between different types of automata, divides them into classes based on the formal languages they can understand. Computer theory, compiler design, artificial intelligence, parsing, and formal verification all use automata. So it is easy to say that the word automaton, which is closely connected to the word "automation," refers to automated processes that carry out specified operations. Simply expressed, automata theory is concerned with the logic of computation in the context of basic machines known as automata.

Related construction of theories: In automata theory, each model performs a key function in a variety of applications. Finite automata are used in text processing, compilers, and hardware design. Context-free grammars (CFGs) are used in computer languages and artificial intelligence. CFGs were first used in the research of human languages. Cellular automata are utilized in the field of artificial life, with John Conway's Game of Life being the most well-known example. The growth and color patterns of mollusks and pine cones are examples of biological phenomena that could be explained using automata theory. Some scientists even advocate for a theory in which the entire universe is computed by a discrete automaton. Additionally, Automata simulators are teaching, learning, and research tools for automata theory. An automata simulator accepts an automaton's description as input and simulates its operation for any input string. The automaton's description can be entered in a variety of ways. A symbolic language can be used to define automata, or its specification can be entered in a pre-designed form, or its transition diagram can be produced by dragging the mouse. Turing's World, JFLAP, VAS, TAGS, and SimStudio are all well-known automata simulators. Following the automata classification into different types given in the previous section, several separate categories of automata can be defined. The Cartesian closed category of deterministic automata, sequential machines or sequential automata, and Turing machines with automata homomorphisms defining the arrows between automata includes both categorical limits and colimits. A quintuple of an automaton A_i is mapped onto the quintuple of another automaton A_j via an automata homomorphism. When the automaton's state space, S , is defined as a semigroup S_g , automata homomorphisms are also known as automata

transformations or semi-group homomorphisms. In monoidal categories, monoids are also seen to provide a good setting for automata.

Computational Complexity theory: One of the fundamental goals of computational complexity theory, which is an area of theoretical computer science, is to classify and compare the practical difficulty of solving issues involving finite combinatorial objects for example, are two natural numbers n and m relatively prime? Is there a satisfying assignment for a propositional formula ϕ Does white have a winning strategy from a given starting position if we play chess on a board of size $n \times n$? In the sense that they are all effectively decidable, these issues are all equally tough from the standpoint of classical computability theory. Despite this, they appear to have a considerable difference in terms of practical difficulty. After being given a pair of values $m > n > 0$, a method (Euclid's algorithm) that requires a number of steps proportional to \log can be used to establish their relative primacy (n). All known approaches for addressing the latter two issues, on the other hand, involve a 'brute force' search across a large class of examples that grows at least exponentially in size. Complexity theory tries to clarify these disparities by proposing a formal condition for what it means for a mathematical problem to be feasibly decidable that is, that it can be solved by a traditional Turing machine in a number of steps proportionate to a polynomial function of its input size. The first of the three issues presented above belongs to the P or polynomial time class of problems with this feature. P can be officially distinguished from other classes such as EXP or exponential time which incorporates the third problem from the previous section. The second problem from the previous section belongs to the NP or non-deterministic polynomial time complexity class, which includes problems that can be correctly solved by a non-deterministic Turing computer in a number of steps that is a polynomial function of the amount of its input. P is also correctly contained in NP, i.e. $P \subseteq NP$, according to a well-known conjecture, which is often regarded as the most fundamental in all of theoretical computer science. The non-coincidence of these and other complexity classes are still an outstanding subject in complexity theory. Even at its current state of development, however, this subject connects numerous problems in logic, mathematics, and related sciences in a way that has implications for the nature and breadth of our knowledge in these fields. Reflection on the foundations of complexity theory could thus be useful not only for computer science philosophy, but also for philosophy of mathematics and epistemology.

In a nutshell, complexity theory categorizes computable issues according to their difficulty. As an example,

- Any problem can be solved quickly if it is approached correctly. Sorting by sequence, for example, or looking up a name.
- If a problem cannot be solved efficiently, it is difficult to solve. Factoring a 500-digit integer into its prime factor, for example.
- The basic goal of computation theory is to create a formal mathematical model of computation that is representative of real-world computers.

Conclusion

It's difficult to properly comprehend the impact of computer science advancement. People all across the world can communicate immediately, enjoy longer lives, and share their voices because to computer science. Tech experts in a variety of computer science vocations contribute to society in a variety of ways. This section examines how computer science's t has affected our present and future. Computer science makes an impact in everything from combating climate change to predicting natural disasters. Professionals in the field of computer science have a long history of improving the world. Students thinking about pursuing a computer science degree should be aware of the history of the field, as well as the potential harm that technology can inflict. Tech professionals can comprehend the responsibility that their field entails by educating themselves with computer science materials. By practicing computer science ethically; experts may ensure that the development of technology helps society while simultaneously safeguarding individual security, privacy, and equality.

References

1. Brookshear, J. G. (1991). Computer science: An overview. Benjamin-Cummings Publishing Co., Inc.
2. Taylor, C., Zingaro, D., Porter, L., Webb, K. C., Lee, C. B., & Clancy, M. (2014). Computer science concept inventories: past and future. *Computer Science Education*, 24(4), 253-276.
3. Colburn, T. (2015). *Philosophy and computer science*. Routledge.
4. Vallverdú, J. (Ed.). (2010). *Thinking machines and the philosophy of computer science: concepts and principles*. IGI Global.
5. Vallverdú, J. (Ed.). (2010). *Thinking machines and the philosophy of computer science: concepts and principles*. IGI Global.
6. Floridi, L. (Ed.). (2008). *The Blackwell guide to the philosophy of computing and information*. John Wiley & Sons.
7. Knight, J. C., Prey, J. C., & Wulf, W. A. (1994, March). Undergraduate computer science education: a new curriculum philosophy & overview. In *Proceedings of the twenty-fifth SIGCSE symposium on Computer science education* (pp. 155-159).
8. Knight, J. C., Prey, J. C., & Wulf, W. A. (1994, March). Undergraduate computer science education: a new curriculum philosophy & overview. In *Proceedings of the twenty-fifth SIGCSE symposium on Computer science education* (pp. 155-159).
9. Aliev, F. A., & Larin, V. B. (Eds.). (1998). *Optimization of linear control systems: analytical methods and computational algorithms*. CRC Press.
10. Denning, P. J. (2000). Computer science: The discipline. *Encyclopedia of computer science*, 32(1), 9-23.
11. Hassan, Muhammed & Jubaer, Shah. (2021). An introduction to Computer Science and its History. 10.13140/RG.2.2.12948.83849.

12. Berglund, A., Eckerdal, A., Pears, A., East, P., Kinnunen, P., Malmi, L., ... & Thomas, L. (2009). Learning computer science: Perceptions, actions and roles. *European Journal of Engineering Education*, 34(4), 327-338.
13. Holt, D. F., Eick, B., & O'Brien, E. A. (2005). *Handbook of computational group theory*. Chapman and Hall/CRC.
14. Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2001). *Introduction to automata theory, languages, and computation*. *Acm Sigact News*, 32(1), 60-65.
15. Shields, M. W. (1987). *An introduction to automata theory*. Blackwell Scientific Publications, Ltd.
16. Jubaer, S., & Hoque, L. (2021). The Concept of Education: A Western Rationalist Approach. *International Journal of Educational Advancement*, 4, 138-150.
17. Hulden, M., 2009, April. Foma: a finite-state compiler and library. In *Proceedings of the Demonstrations Session at EACL 2009* (pp. 29-32).
18. Jubaer, S. M. O. F., & Hassan, M. N. (2021). THE ARTIFICIAL INTELLIGENCE FOR THE COMMON LEARNERS: A COMPARATIVE LEARNING APPROACH. *Web of Scientist: International Scientific Research Journal*, 2(05), 333-3525.
19. Yusuf, S.A., 2013. *Design and Implementation of Multimedia Content on Smartphone for Learning Automata Theory* (Doctoral dissertation, AUST).
20. Hedetniemi, S.T., 1966. *HOMOMORPHISMS OF GRAPHS AND AUTOMATA*. MICHIGAN UNIV ANN ARBOR COMMUNICATION SCIENCES PROGRAM.
21. Grellet, F., & Francoise, G. (1981). *Developing reading skills: A practical guide to reading comprehension exercises*. Cambridge university press.
22. Backhouse, R. and Ferreira, J.F., 2011. On Euclid's algorithm and elementary number theory. *Science of Computer Programming*, 76(3), pp.160-180.
23. Ellis, T. J., & Levy, Y. (2009). *Towards a Guide for Novice Researchers on Research Methodology: Review and Proposed Methods*. *Issues in Informing Science & Information Technology*, 6.
24. Bansal, N., Chalermsook, P., Laekhanukit, B., Nanongkai, D. and Nederlof, J., 2019. New tools and connections for exponential-time approximation. *Algorithmica*, 81(10), pp.3993-4009.
25. Myers, M. D., & Avison, D. (Eds.). (2002). *Qualitative research in information systems: a reader*. Sage.
26. Romiszowski, A. J. (2016). *Designing instructional systems: Decision making in course planning and curriculum design*. Routledge.
27. Bødker, Susanne. *Through the interface: A human activity approach to user interface design*. CRC Press, 2021.
28. Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003, May). New directions on agile methods: a comparative analysis. In *25th International Conference on Software Engineering, 2003. Proceedings.* (pp. 244-254). Ieee.
29. Rubin, K. S. (2012). *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley.