IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 9, Issue 11, November 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.542

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542|| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 11, November 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0911049 |

Role of Mathematics in Computer Science

Dr. Pawan Chanchal

Assistant Professor, Department of Mathematics, Government Girls College, Ajmer, Rajasthan, India

ABSTRACT: Computational mathematics is an area of mathematics devoted to the interaction between mathematics and computer computation.[1]

A large part of computational mathematics consists roughly of using mathematics for allowing and improving computer computation in areas of science and engineering where mathematics are useful. This involves in particular algorithm design, computational complexity, numerical methods and computer algebra.

Computational mathematics refers also to the use of computers for mathematics itself. This includes mathematical experimentation for establishing conjectures (particularly in number theory), the use of computers for proving theorems (for example the four color theorem), and the design and use of proof assistants. Computer-Based Math is an educational project started by Conrad Wolfram in 2010[1][2][3][4] to promote the idea that routine mathematical calculations should be done with a computer.

Conrad Wolfram believes that mathematics education should make the greatest possible use of computers for performing computation leaving students to concentrate on the application and interpretation of mathematical techniques.[5] Wolfram also argues that computers are the basis of doing math in the real world and that education should reflect that [6] and that programming should be taught as part of math education.[7]

Wolfram contends that this approach is fundamentally different from most of the use of Computers in the classroom (or Computer-based mathematics education),[8] whose role is to help to teach students to perform hand calculations, rather than to perform those computations and is also distinct from delivery tools such as E-learning systems.

In 2010 the website www.computerbasedmath.org was set up to start developing a new curriculum and interactive digital learning materials to support it. It holds an annual conference.

In February 2013, Estonia announced that it would be piloting a Computer-Based Math developed statistics course[9][10][11] in cooperation with the University of Tartu.[12] The African Leadership University plans to use materials developed by ComputerBasedMath.org in its Data and Decisions curriculum.[13]

UNICEF supported the third Computer-Based Math Education Summit in New York, in 2013.[14]

Examples of calculations that should be done with a computer include arithmetical operations such as long division or integration techniques such as trigonometric substitution.

In 2020 Wolfram published a book "The Math(s) Fix" detailing the problems and his proposed solution.[15]

KEYWORDS: mathematics, computers, role, computational, e-learning, engineering, algorithm, algebra, education,numerical

I. INTRODUCTION

Computer science is the study of computation, information, and automation.[1][2][3] Computer science spans theoretical disciplines (such as algorithms, theory of computation, and information theory) to applied disciplines (including the design and implementation of hardware and software).[4][5][6] Though more often considered an academic discipline, computer science is closely related to computer programming.[7]

Algorithms and data structures are central to computer science.[8] The theory of computation concerns abstract models of computation and general classes of problems that can be solved using them. The fields of cryptography and computer security involve studying the means for secure communication and for preventing security vulnerabilities. Computer graphics and computational geometry address the generation of images. Programming language theory considers different ways to describe computational processes, and database theory concerns the management of repositories of data. Human–computer interaction investigates the interfaces through which humans and computers interact, and software engineering focuses on the design and principles behind developing software. Areas such as operating systems, networks and embedded systems investigate the principles and design behind complex systems. Computer architecture describes the construction of computer components and computer-

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542|| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 11, November 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0911049 |

operated equipment. Artificial intelligence and machine learning aim to synthesize goal-orientated processes such as problem-solving, decision-making, environmental adaptation, planning and learning found in humans and animals. Within artificial intelligence, computer vision aims to understand and process image and video data, while natural language processing aims to understand and process textual and linguistic data.

The fundamental concern of computer science is determining what can and cannot be automated.[2][9][3][10][11] The Turing Award is generally recognized as the highest distinction in computer science.[12][13]

The earliest foundations of what would become computer science predate the invention of the modern digital computer. Machines for calculating fixed numerical tasks such as the abacus have existed since antiquity, aiding in computations such as multiplication and division. Algorithms for performing computations have existed since antiquity, even before the development of sophisticated computing equipment.[17]

Wilhelm Schickard designed and constructed the first working mechanical calculator in 1623.[18] In 1673, Gottfried Leibniz demonstrated a digital mechanical calculator, called the Stepped Reckoner.[19] Leibniz may be considered the first computer scientist and information theorist, because of various reasons, including the fact that he documented the binary number system. In 1820, Thomas de Colmar launched the mechanical calculator industry[note 1] when he invented his simplified arithmometer, the first calculating machine strong enough and reliable enough to be used daily in an office environment. Charles Babbage started the design of the first automatic mechanical calculator, his Difference Engine, in 1822, which eventually gave him the idea of the first programmable mechanical calculator, his Analytical Engine.[20] He started developing this machine in 1834, and "in less than two years, he had sketched out many of the salient features of the modern computer".[21] "A crucial step was the adoption of a punched card system derived from the Jacquard loom"[21] making it infinitely programmable.[note 2] In 1843, during the translation of a French article on the Analytical Engine, Ada Lovelace wrote, in one of the many notes she included, an algorithm to compute the Bernoulli numbers, which is considered to be the first published algorithm ever specifically tailored for implementation on a computer.[22] Around 1885, Herman Hollerith invented the tabulator, which used punched cards to process statistical information; eventually his company became part of IBM. Following Babbage, although unaware of his earlier work, Percy Ludgate in 1909 published[23] the 2nd of the only two designs for mechanical analytical engines in history. In 1914, the Spanish engineer Leonardo Torres Quevedo published his Essays on Automatics,[24] and designed, inspired by Babbage, a theoretical electromechanical calculating machine which was to be controlled by a read-only program. The paper also introduced the idea of floating-point arithmetic. In 1920, to celebrate the 100th anniversary of the invention of the arithmometer, Torres presented in Paris the Electromechanical Arithmometer, a prototype that used relays to implement the functions of an arithmetic unit, on which commands could be typed and the results printed automatically.[25] In 1937, one hundred years after Babbage's impossible dream, Howard Aiken convinced IBM, which was making all kinds of punched card equipment and was also in the calculator business[26] to develop his giant programmable calculator, the ASCC/Harvard Mark I, based on Babbage's Analytical Engine, which itself used cards and a central computing unit. When the machine was finished, some hailed it as "Babbage's dream come true".[27]

During the 1940s, with the development of new and more powerful computing machines such as the Atanasoff–Berry computer and ENIAC, the term computer came to refer to the machines rather than their human predecessors.[28] As it became clear that computers could be used for more than just mathematical calculations, the field of computer science broadened to study computation in general. In 1945, IBM founded the Watson Scientific Computing Laboratory at Columbia University in New York City. The renovated fraternity house on Manhattan's West Side was IBM's first laboratory devoted to pure science. The lab is the forerunner of IBM's Research Division, which today operates research facilities around the world.[29] Ultimately, the close relationship between IBM and Columbia University was instrumental in the emergence of a new scientific discipline, with Columbia offering one of the first academic-credit courses in computer science in 1946.[30] Computer science began to be established as a distinct academic discipline in the 1950s and early 1960s.[7][31] The world's first computer science degree program, the Cambridge Diploma in Computer Science, began at the University of Cambridge Computer Laboratory in 1953. The first computer science department in the United States was formed at Purdue University in 1962.[32] Since practical computers became available, many applications of computing have become distinct areas of study in their own rights.

Although first proposed in 1956,[33] the term "computer science" appears in a 1959 article in Communications of the ACM,[34] in which Louis Fein argues for the creation of a Graduate School in Computer Sciences analogous to the creation of Harvard Business School in 1921.[35] Louis justifies the name by arguing that, like management science, the subject is applied and interdisciplinary in nature, while having the characteristics typical of an academic discipline.[34] His efforts, and those of others such as numerical analyst George Forsythe, were rewarded: universities went on to create such departments, starting with Purdue in 1962.[36] Despite its name, a significant amount of computer science does not involve the study of computers themselves. Because of this, several alternative names have been proposed.[37] Certain departments of major universities prefer the term computing science, to emphasize precisely that dif-

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542|| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 11, November 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0911049 |

ference. Danish scientist Peter Naur suggested the term datalogy,[38] to reflect the fact that the scientific discipline revolves around data and data treatment, while not necessarily involving computers. The first scientific institution to use the term was the Department of Datalogy at the University of Copenhagen, founded in 1969, with Peter Naur being the first professor in datalogy. The term is used mainly in the Scandinavian countries. An alternative term, also proposed by Naur, is data science; this is now used for a multi-disciplinary field of data analysis, including statistics and databases.

In the early days of computing, a number of terms for the practitioners of the field of computing were suggested in the Communications of the ACM—turingineer, turologist, flow-charts-man, applied meta-mathematician, and applied epistemologist.[39] Three months later in the same journal, comptologist was suggested, followed next year by hypologist.[40] The term computics has also been suggested.[41] In Europe, terms derived from contracted translations of the expression "automatic information" (e.g. "informazione automatica" in Italian) or "information and mathematics" are often used, e.g. informatique (French), Informatik (German), informatica (Italian, Dutch), informática (Spanish, Portuguese), informatika (Slavic languages and Hungarian) or pliroforiki (πληροφορική, which means informatics) in Greek. Similar words have also been adopted in the UK (as in the School of Informatics, University of Edinburgh).[42] "In the U.S., however, informatics is linked with applied computing, or computing in the context of another domain."[43]

A folkloric quotation, often attributed to—but almost certainly not first formulated by—Edsger Dijkstra, states that "computer science is no more about computers than astronomy is about telescopes." The design and deployment of computers and computer systems is generally considered the province of disciplines other than computer science. For example, the study of computer hardware is usually considered part of computer engineering, while the study of commercial computer systems and their deployment is often called information technology or information systems. However, there has been exchange of ideas between the various computer-related disciplines. Computer science research also often intersects other disciplines, such as cognitive science, linguistics, mathematics, physics, biology, Earth science, statistics, philosophy, and logic.

Computer science is considered by some to have a much closer relationship with mathematics than many scientific disciplines, with some observers saying that computing is a mathematical science.[7] Early computer science was strongly influenced by the work of mathematicians such as Kurt Gödel, Alan Turing, John von Neumann, Rózsa Péter and Alonzo Church and there continues to be a useful interchange of ideas between the two fields in areas such as mathematical logic, category theory, domain theory, and algebra.[33]

The relationship between computer science and software engineering is a contentious issue, which is further muddied by disputes over what the term "software engineering" means, and how computer science is defined.[44] David Parnas, taking a cue from the relationship between other engineering and science disciplines, has claimed that the principal focus of computer science is studying the properties of computation in general, while the principal focus of software engineering is the design of specific computations to achieve practical goals, making the two separate but complementary disciplines.[45]

The academic, political, and funding aspects of computer science tend to depend on whether a department is formed with a mathematical emphasis or with an engineering emphasis. Computer science departments with a mathematics emphasis and with a numerical orientation consider alignment with computational science. Both types of departments tend to make efforts to bridge the field educationally if not across all research.

Despite the word "science" in its name, there is debate over whether or not computer science is a discipline of science,[46] mathematics,[47] or engineering.[48] Allen Newell and Herbert A. Simon argued in 1975,

Computer science is an empirical discipline. We would have called it an experimental science, but like astronomy, economics, and geology, some of its unique forms of observation and experience do not fit a narrow stereotype of the experimental method. Nonetheless, they are experiments. Each new machine that is built is an experiment. Actually constructing the machine poses a question to nature; and we listen for the answer by observing the machine in operation and analyzing it by all analytical and measurement means available.[48]

It has since been argued that computer science can be classified as an empirical science since it makes use of empirical testing to evaluate the correctness of programs, but a problem remains in defining the laws and theorems of computer science (if any exist) and defining the nature of experiments in computer science.[48] Proponents of classifying computer science as an engineering discipline argue that the reliability of computational systems is investigated in the same way as bridges in civil engineering and airplanes in aerospace engineering.[48] They also argue that while empirical sciences observe what presently exists, computer science observes what is possible to exist and while scientists discover

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542|| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 11, November 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0911049 |

laws from observation, no proper laws have been found in computer science and it is instead concerned with creating phenomena.[48]

Proponents of classifying computer science as a mathematical discipline argue that computer programs are physical realizations of mathematical entities and programs can be deductively reasoned through mathematical formal methods.[48] Computer scientists Edsger W. Dijkstra and Tony Hoare regard instructions for computer programs as mathematical sentences and interpret formal semantics for programming languages as mathematical axiomatic systems.[48]

A number of computer scientists have argued for the distinction of three separate paradigms in computer science. Peter Wegner argued that those paradigms are science, technology, and mathematics.[49] Peter Denning's working group argued that they are theory, abstraction (modeling), and design.[7] Amnon H. Eden described them as the "rationalist paradigm" (which treats computer science as a branch of mathematics, which is prevalent in theoretical computer science, and mainly employs deductive reasoning), the "technocratic paradigm" (which might be found in engineering approaches, most prominently in software engineering), and the "scientific paradigm" (which approaches computer-related artifacts from the empirical perspective of natural sciences,[50] identifiable in some branches of artificial intelligence).[51] Computer science focuses on methods involved in design, specification, programming, verification, implementation and testing of human-made computing systems.[52]

As a discipline, computer science spans a range of topics from theoretical studies of algorithms and the limits of computation to the practical issues of implementing computing systems in hardware and software.[53][54] CSAB, formerly called Computing Sciences Accreditation Board—which is made up of representatives of the Association for Computing Machinery (ACM), and the IEEE Computer Society (IEEE CS)[55]—identifies four areas that it considers crucial to the discipline of computer science: theory of computation, algorithms and data structures, programming methodology and languages, and computer elements and architecture. In addition to these four areas, CSAB also identifies fields such as software engineering, artificial intelligence, computer networking and communication, database systems, parallel computation, distributed computation, human–computer interaction, computer graphics, operating systems, and numerical and symbolic computation as being important areas of computer science.[53]

Computer science is no more about computers than astronomy is about telescopes.

Theoretical Computer Science is mathematical and abstract in spirit, but it derives its motivation from the practical and everyday computation. Its aim is to understand the nature of computation and, as a consequence of this understanding, provide more efficient methodologies.

According to Peter Denning, the fundamental question underlying computer science is, "What can be automated?"[3] Theory of computation is focused on answering fundamental questions about what can be computed and what amount of resources are required to perform those computations. In an effort to answer the first question, computability theory examines which computational problems are solvable on various theoretical models of computation. The second question is addressed by computational complexity theory, which studies the time and space costs associated with different approaches to solving a multitude of computational problems.

II. DISCUSSION

Programming language theory is a branch of computer science that deals with the design, implementation, analysis, characterization, and classification of programming languages and their individual features. It falls within the discipline of computer science, both depending on and affecting mathematics, software engineering, and linguistics. It is an active research area, with numerous dedicated academic journals.

Formal methods are a particular kind of mathematically based technique for the specification, development and verification of software and hardware systems.[59] The use of formal methods for software and hardware design is motivated by the expectation that, as in other engineering disciplines, performing appropriate mathematical analysis can contribute to the reliability and robustness of a design. They form an important theoretical underpinning for software engineering, especially where safety or security is involved. Formal methods are a useful adjunct to software testing since they help avoid errors and can also give a framework for testing. For industrial use, tool support is required. However, the high cost of using formal methods means that they are usually only used in the development of high-integrity and life-critical systems, where safety or security is of utmost importance. Formal methods are best described as the application of a fairly broad variety of theoretical computer science fundamentals, in particular logic calculi, formal languages, automata theory, and program semantics, but also type systems and algebraic data types to problems in software and hardware specification and verification.

Scientific computing (or computational science) is the field of study concerned with constructing mathematical models and quantitative analysis techniques and using computers to analyze and solve scientific problems. A major usage of

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542|| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 11, November 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0911049 |

scientific computing is simulation of various processes, including computational fluid dynamics, physical, electrical, and electronic systems and circuits, as well as societies and social situations (notably war games) along with their habitats, among many others. Modern computers enable optimization of such designs as complete aircraft. Notable in electrical and electronic circuit design are SPICE,[60] as well as software for physical realization of new (or modified) designs. The latter includes essential design software for integrated circuits.[61] Artificial intelligence (AI) aims to or is required to synthesize goal-orientated processes such as problem-solving, decision-making, environmental adaptation, learning, and communication found in humans and animals. From its origins in cybernetics and in the Dartmouth Conference (1956), artificial intelligence research has been necessarily cross-disciplinary, drawing on areas of expertise such as applied mathematics, symbolic logic, semiotics, electrical engineering, philosophy of mind, neurophysiology, and social intelligence. AI is associated in the popular mind with robotic development, but the main field of practical application has been as an embedded component in areas of software development, which require computational understanding. The starting point in the late 1940s was Alan Turing's question "Can computers think?", and the question remains effectively unanswered, although the Turing test is still used to assess computer output on the scale of human intelligence. But the automation of evaluative and predictive tasks has been increasingly successful as a substitute for human monitoring and intervention in domains of computer application involving complex real-world data.[35]

III. RESULTS

In mathematics and computer science,[1] computer algebra, also called symbolic computation or algebraic computation, is a scientific area that refers to the study and development of algorithms and software for manipulating mathematical expressions and other mathematical objects. Although computer algebra could be considered a subfield of scientific computing, they are generally considered as distinct fields because scientific computing is usually based on numerical computation with approximate floating point numbers, while symbolic computation emphasizes exact computation with expressions containing variables that have no given value and are manipulated as symbols.

Software applications that perform symbolic calculations are called computer algebra systems, with the term system alluding to the complexity of the main applications that include, at least, a method to represent mathematical data in a computer, a user programming language (usually different from the language used for the implementation), a dedicated memory manager, a user interface for the input/output of mathematical expressions, a large set of routines to perform usual operations, like simplification of expressions, differentiation using chain rule, polynomial factorization, indefinite integration, etc.[37]

Computer algebra is widely used to experiment in mathematics and to design the formulas that are used in numerical programs. It is also used for complete scientific computations, when purely numerical methods fail, as in public key cryptography, or for some non-linear problems.

Some authors distinguish computer algebra from symbolic computation using the latter name to refer to kinds of symbolic computation other than the computation with mathematical formulas. Some authors use symbolic computation for the computer science aspect of the subject and "computer algebra" for the mathematical aspect.[2] In some languages the name of the field is not a direct translation of its English name. Typically, it is called calcul formel in French, which means "formal computation". This name reflects the ties this field has with formal methods.

Symbolic computation has also been referred to, in the past, as symbolic manipulation, algebraic manipulation, symbolic processing, symbolic mathematics, or symbolic algebra, but these terms, which also refer to non-computational manipulation, are no longer used in reference to computer algebra.

Therefore, the basic numbers used in computer algebra are the integers of the mathematicians, commonly represented by an unbounded signed sequence of digits in some base of numeration, usually the largest base allowed by the machine word. These integers allow to define the rational numbers, which are irreducible fractions of two integers.[38]

Programming an efficient implementation of the arithmetic operations is a hard task. Therefore, most free computer algebra systems and some commercial ones such as Mathematica and Maple (software), use the GMP library, which is thus a de facto standard.

Except for numbers and variables, every mathematical expression may be viewed as the symbol of an operator followed by a sequence of operands. In computer algebra software, the expressions are usually represented in this way. This representation is very flexible, and many things that seem not to be mathematical expressions at first glance, may be represented and manipulated as such. For example, an equation is an expression with "=" as an operator, a matrix may be represented as an expression with "matrix" as an operator and its rows as operands.[39]

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542|| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 11, November 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0911049 |

Even programs may be considered and represented as expressions with operator "procedure" and, at least, two operands, the list of parameters and the body, which is itself an expression with "body" as an operator and a sequence of instructions as operands. Conversely, any mathematical expression may be viewed as a program. For example, the expression a + b may be viewed as a program for the addition, with a and b as parameters. Executing this program consists in evaluating the expression for given values of a and b; if they are not given any values, the result of the evaluation is simply its input.

This process of delayed evaluation is fundamental in computer algebra. For example, the operator "=" of the equations is also, in most computer algebra systems, the name of the program of the equality test: normally, the evaluation of an equation results in an equation, but, when an equality test is needed, either explicitly asked by the user through an "evaluation to a Boolean" command, or automatically started by the system in the case of a test inside a program, then the evaluation to a boolean result is executed.

As the size of the operands of an expression is unpredictable and may change during a working session, the sequence of the operands is usually represented as a sequence of either pointers (like in Macsyma) or entries in a hash table (like in Maple).

At the beginning of computer algebra, circa 1970, when the long-known algorithms were first put on computers, they turned out to be highly inefficient.[10] Therefore, a large part of the work of the researchers in the field consisted in revisiting classical algebra in order to make it effective and to discover efficient algorithms to implement this effectiveness. A typical example of this kind of work is the computation of polynomial greatest common divisors, which is required to simplify fractions. Surprisingly, the classical Euclid's algorithm turned out to be inefficient for polynomials over infinite fields, and thus new algorithms needed to be developed. The same was also true for the classical algorithms from linear algebra.[50]

IV. CONCLUSIONS

Theoretical computer science (TCS) is a subset of general computer science and mathematics that focuses on mathematical aspects of computer science such as the theory of computation, lambda calculus, and type theory.

It is difficult to circumscribe the theoretical areas precisely. The ACM's Special Interest Group on Algorithms and Computation Theory (SIGACT) provides the following description:[1]

TCS covers a wide variety of topics including algorithms, data structures, computational complexity, parallel and distributed computation, probabilistic computation, quantum computation, automata theory, information theory, cryptography, program semantics and verification, algorithmic game theory, machine learning, computational biology, computational economics, computational geometry, and computational number theory and algebra. Work in this field is often distinguished by its emphasis on mathematical technique and rigor.

While logical inference and mathematical proof had existed previously, in 1931 Kurt Gödel proved with his incompleteness theorem that there are fundamental limitations on what statements could be proved or disproved.

Information theory was added to the field with a 1948 mathematical theory of communication by Claude Shannon. In the same decade, Donald Hebb introduced a mathematical model of learning in the brain. With mounting biological data supporting this hypothesis with some modification, the fields of neural networks and parallel distributed processing were established. In 1971, Stephen Cook and, working independently, Leonid Levin, proved that there exist practically relevant problems that are NP-complete – a landmark result in computational complexity theory[51]

With the development of quantum mechanics in the beginning of the 20th century came the concept that mathematical operations could be performed on an entire particle wavefunction. In other words, one could compute functions on multiple states simultaneously. This led to the concept of a quantum computer in the latter half of the 20th century that took off in the 1990s when Peter Shor showed that such methods could be used to factor large numbers in polynomial time, which, if implemented, would render some modern public key cryptography algorithms like RSA insecure. Computational geometry is a branch of computer science devoted to the study of algorithms that can be stated in terms of geometry. Some purely geometrical problems arise out of the study of computational geometric algorithms, and such problems are also considered to be part of computational geometry.[52]

The main impetus for the development of computational geometry as a discipline was progress in computer graphics and computer-aided design and manufacturing (CAD/CAM), but many problems in computational geometry are classical in nature, and may come from mathematical visualization.

Other important applications of computational geometry include robotics (motion planning and visibility problems), geographic information systems (GIS) (geometrical location and search, route planning), integrated cir-

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542|| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 11, November 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0911049 |

cuit design (IC geometry design and verification), computer-aided engineering (CAE) (mesh generation), computer vision (3D reconstruction).[53]

REFERENCES

1. "SIGACT". Retrieved 2017-01-19.
2. ^ "Any classical mathematical algorithm, for example, can be described in a finite number of English words". Rogers, Hartley Jr. (1967). Theory of Recursive Functions and Effective Computability. McGraw-Hill. Page 2.
3. ^ Well defined with respect to the agent that executes the algorithm: "There is a computing agent, usually human, which can react to the instructions and carry out the computations" (Rogers 1967, p. 2).
4. ^ "an algorithm is a procedure for computing a function (with respect to some chosen notation for integers) ... this limitation (to numerical functions) results in no loss of generality", (Rogers 1967, p. 1).
5. ^ "An algorithm has zero or more inputs, i.e., quantities which are given to it initially before the algorithm begins" (Knuth 1973:5).
6. ^ "A procedure which has all the characteristics of an algorithm except that it possibly lacks finiteness may be called a 'computational method'" (Knuth 1973:5).
7. ^ "An algorithm has one or more outputs, i.e. quantities which have a specified relation to the inputs" (Knuth 1973:5).
8. ^ Whether or not a process with random interior processes (not including the input) is an algorithm is debatable. Rogers opines that: "a computation is carried out in a discrete stepwise fashion, without the use of continuous methods or analog devices . . . carried forward deterministically, without resort to random methods or devices, e.g., dice" (Rogers 1967, p. 2).
9. ^ "NIH working definition of bioinformatics and computational biology" (PDF). Biomedical Information Science and Technology Initiative. 17 July 2000. Archived from the original (PDF) on 5 September 2012. Retrieved 18 August 2012.
10. ^ "About the CCMB". Center for Computational Molecular Biology. Retrieved 18 August 2012.
11. ^ Rivest, Ronald L. (1990). "Cryptology". In J. Van Leeuwen (ed.). Handbook of Theoretical Computer Science. Vol. 1. Elsevier.
12. ^ Bellare, Mihir; Rogaway, Phillip (21 September 2005). "Introduction". Introduction to Modern Cryptography. p. 10.
13. ^ Menezes, A. J.; van Oorschot, P. C.; Vanstone, S. A. (1997). Handbook of Applied Cryptography. ISBN 978-0-8493-8523-0.
14. ^ Paul E. Black (ed.), entry for data structure in Dictionary of Algorithms and Data Structures. U.S. National Institute of Standards and Technology. 15 December 2004. Online version Accessed May 21, 2009.
15. ^ Entry data structure in the Encyclopædia Britannica (2009) Online entry accessed on May 21, 2009.
16. ^ Coulouris, George; Jean Dollimore; Tim Kindberg; Gordon Blair (2011). Distributed Systems: Concepts and Design (5th ed.). Boston: Addison-Wesley. ISBN 978-0-132-14301-1.
17. ^ Andrews (2000). Dolev (2000). Ghosh (2007), p. 10.
18. ^ R. W. Butler (2001-08-06). "What is Formal Methods?". Retrieved 2006-11-16.
19. ^ C. Michael Holloway. "Why Engineers Should Consider Formal Methods" (PDF). 16th Digital Avionics Systems Conference (27–30 October 1997). Archived from the original (PDF) on 16 November 2006. Retrieved 2006-11-16.
20. ^ Monin, pp.3–4
21. ^ F. Rieke; D. Warland; R Ruyter van Steveninck; W Bialek (1997). Spikes: Exploring the Neural Code. The MIT press. ISBN 978-0262681087.
22. ^ Huelsenbeck, J. P.; Ronquist, F.; Nielsen, R.; Bollback, J. P. (2001-12-14). "Bayesian Inference of Phylogeny and Its Impact on Evolutionary Biology". Science. American Association for the Advancement of Science (AAAS). 294 (5550): 2310–2314. Bibcode:2001Sci...294.2310H. doi:10.1126/science.1065889. ISSN 0036-8075. PMID 11743192. S2CID 2138288.
23. ^ Rando Allikmets, Wyeth W. Wasserman, Amy Hutchinson, Philip Smallwood, Jeremy Nathans, Peter K. Rogan, Thomas D. Schneider, Michael Dean (1998) Organization of the ABCR gene: analysis of promoter and splice junction sequences, Gene 215:1, 111–122
24. ^ Burnham, K. P. and Anderson D. R. (2002) Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach, Second Edition (Springer Science, New York) ISBN 978-0-387-95364-9.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542|| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 11, November 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0911049 |

25. ^ Jaynes, E. T. (1957-05-15). "Information Theory and Statistical Mechanics". Physical Review. American Physical Society (APS). 106 (4): 620–630. Bibcode:1957PhRv..106..620J. doi:10.1103/physrev.106.620. ISSN 0031-899X.
26. ^ Charles H. Bennett, Ming Li, and Bin Ma (2003) Chain Letters and Evolutionary Histories, Scientific American 288:6, 76–81
27. ^ David R. Anderson (November 1, 2003). "Some background on why people in the empirical sciences may want to better understand the information-theoretic methods" (PDF). Archived from the original (PDF) on July 23, 2011. Retrieved 2010-06-23.
28. ^ Ron Kovahi; Foster Provost (1998). "Glossary of terms". Machine Learning. 30: 271–274. doi:10.1023/A:1007411609915.
29. ^ C. M. Bishop (2006). Pattern Recognition and Machine Learning. Springer. ISBN 978-0-387-31073-2.
30. ^ Wernick, Yang, Brankov, Yourganov and Strother, Machine Learning in Medical Imaging, IEEE Signal Processing Magazine, vol. 27, no. 4, July 2010, pp. 25–38
31. ^ Mannila, Heikki (1996). Data mining: machine learning, statistics, and databases. Int'l Conf. Scientific and Statistical Database Management. IEEE Computer Society.
32. ^ Friedman, Jerome H. (1998). "Data Mining and Statistics: What's the connection?". Computing Science and Statistics. 29 (1): 3–9.
33. ^ Gottlieb, Allan; Almasi, George S. (1989). Highly parallel computing. Redwood City, Calif.: Benjamin/Cummings. ISBN 978-0-8053-0177-9.
34. ^ S.V. Adve et al. (November 2008). "Parallel Computing Research at Illinois: The UPCRC Agenda" Archived 2008-12-09 at the Wayback Machine (PDF). Parallel@Illinois, University of Illinois at Urbana-Champaign. "The main techniques for these performance benefits – increased clock frequency and smarter but increasingly complex architectures – are now hitting the so-called power wall. The computer industry has accepted that future performance increases must largely come from increasing the number of processors (or cores) on a die, rather than making a single core go faster."
35. ^ Asanovic et al. Old [conventional wisdom]: Power is free, but transistors are expensive. New [conventional wisdom] is [that] power is expensive, but transistors are "free".
36. ^ Asanovic, Krste et al. (December 18, 2006). "The Landscape of Parallel Computing Research: A View from Berkeley" (PDF). University of California, Berkeley. Technical Report No. UCB/EECS-2006-183. "Old [conventional wisdom]: Increasing clock frequency is the primary method of improving processor performance. New [conventional wisdom]: Increasing parallelism is the primary method of improving processor performance ... Even representatives from Intel, a company generally associated with the 'higher clock-speed is better' position, warned that traditional approaches to maximizing performance through maximizing clock speed have been pushed to their limit."
37. ^ Hennessy, John L.; Patterson, David A.; Larus, James R. (1999). Computer organization and design : the hardware/software interface (2. ed., 3rd print. ed.). San Francisco: Kaufmann. ISBN 978-1-55860-428-5.
38. ^ "Quantum Computing with Molecules" article in Scientific American by Neil Gershenfeld and Isaac L. Chuang
39. ^ Manin, Yu. I. (1980). Vychislimoe i nevychislimoe [Computable and Noncomputable] (in Russian). Sov.Radio. pp. 13–15. Archived from the original on 10 May 2013. Retrieved 4 March 2013.
40. ^ Feynman, R. P. (1982). "Simulating physics with computers". International Journal of Theoretical Physics. 21 (6): 467–488. Bibcode:1982IJTP...21..467F. CiteSeerX 10.1.1.45.9310. doi:10.1007/BF02650179. S2CID 124545445.
41. ^ Deutsch, David (1992-01-06). "Quantum computation". Physics World. 5 (6): 57–61. doi:10.1088/2058-7058/5/6/38.
42. ^ Finkelstein, David (1968). "Space-Time Structure in High Energy Interactions". In Gudehus, T.; Kaiser, G. (eds.). Fundamental Interactions at High Energy. New York: Gordon & Breach.
43. ^ "New qubit control bodes well for future of quantum computing". Retrieved 26 October 2014.
44. ^ Quantum Information Science and Technology Roadmap for a sense of where the research is heading.
45. ^ The 2007 Australian Ranking of ICT Conferences Archived 2009-10-02 at the Wayback Machine: tier A+.
46. ^ MFCS 2017
47. ^ CSR 2018
48. ^ The 2007 Australian Ranking of ICT Conferences Archived 2009-10-02 at the Wayback Machine: tier A.
49. ^ FCT 2011 (retrieved 2013-06-03)

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.542|| A Monthly Peer Reviewed & Referred Journal |

|| Volume 9, Issue 11, November 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0911049 |

50. Nelson, Richard. "Hewlett-Packard Calculator Firsts". Hewlett-Packard. Archived from the original on 2010-07-03.
51. ^ "REDUCE Computer Algebra System at SourceForge". reduce-algebra.sourceforge.net. Retrieved 2015-09-28.
52. ^ Interview with Gaston Gonnet, co-creator of Maple Archived 2007-12-29 at the Wayback Machine, SIAM History of Numerical Analysis and Computing, March 16, 2005.
53. ^ Bhattacharya, Jyotirmoy (2020-05-12). "Wolfram|Alpha: a free online computer algebra system". The Hindu. ISSN 0971-751X. Retrieved 2020-04-26.

INNO SPACE
SJIF Scientific Journal Impact Factor
Impact Factor: 7.542

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

Scan to save the contact details