Computation

From Wikipedia, the free encyclopedia

A **computation** is any type of arithmetic or non-arithmetic calculation that is well-defined.[1][2] Common examples of computation are mathematical equation solving and the execution of computer algorithms.

Mechanical or electronic devices (or, historically, people) that perform computations are known as *computers*. Computer science is an academic field that involves the study of computation.

Introduction [edit]

The notion that mathematical statements should be 'well-defined' had been argued by mathematicians since at least the 1600s,[3] but agreement on a suitable definition proved elusive.[4] A candidate definition was proposed independently by several mathematicians in the 1930s.[5] The best-known variant was formalised by the mathematician Alan Turing, who defined a well-defined statement or calculation as any statement that could be expressed in terms of the initialisation parameters of a Turing machine.[6] Other (mathematically equivalent) definitions include Alonzo Church's *lambda-definability*, Herbrand-Gödel-Kleene's *general recursiveness* and Emil Post's *1-definability*.[5]

Today, any formal statement or calculation that exhibits this quality of well-definedness is termed **computable**, while the statement or calculation itself is referred to as a **computation**.

Turing's definition apportioned "well-definedness" to a very large class of mathematical statements, including all well-formed algebraic statements, and all statements written in modern computer programming languages.[7]

Despite the widespread uptake of this definition, there are some mathematical concepts that have no well-defined characterisation under this definition. This includes the halting problem and the busy beaver game. It remains an open question as to whether there exists a more powerful definition of 'well-defined' that is able to capture both computable and 'non-computable' statements.[note 1][8]

Some examples of mathematical statements that are computable include:

- All statements characterised in modern programming languages, including C++, Python, and Java.[7]
- All calculations carried by an electronic computer, calculator or abacus.
- All calculations carried out on an analytical engine.
- All calculations carried out on a Turing Machine.
- The majority of mathematical statements and calculations given in maths textbooks.

Some examples of mathematical statements that are *not* computable include:

- Calculations or statements which are ill-defined, such that they cannot be unambiguously encoded into a Turing machine: ("Paul loves me twice as much as Joe").
- Problem statements which do appear to be well-defined, but for which it can be proved that no Turing machine exists to solve them (such as the halting problem).

Computation can be seen as a purely physical process occurring inside a closed physical system called a computer. Turing's 1937 proof, *On Computable Numbers, with an Application to the Entscheidungsproblem*, demonstrated that there is a formal equivalence between computable statements and particular physical systems, commonly called computers. Examples of such physical systems are: Turing machines, human mathematicians following strict rules, digital computers, mechanical computers, analog computers and others.

Alternative accounts of computation [edit]

The mapping account [edit]

An alternative account of computation is found throughout the works of Hilary Putnam and others. Peter Godfrey-Smith has dubbed this the "simple mapping account."[9] Gualtiero Piccinini's summary of this account states that a physical system can be said to perform a specific computation when there is a mapping between the state of that system and the computation such that the "microphysical states [of the system] mirror the state transitions between the computational states."[10]

The semantic account [edit]

Philosophers such as Jerry Fodor[11] have suggested various accounts of computation with the restriction that semantic content be a necessary condition for computation (that is, what differentiates an arbitrary physical system from a computing system is that the operands of the computation represent something). This notion attempts to prevent the logical abstraction of the mapping account of pancomputationalism, the idea that everything can be said to be computing everything.

The mechanistic account [edit]

Gualtiero Piccinini proposes an account of computation based on mechanical philosophy. It states that physical computing systems are types of mechanisms that, by design, perform physical computation, or the manipulation (by a functional mechanism) of a "medium-independent" vehicle according to a rule. "Medium-independence" requires that the property can be instantiated[clarification needed] by multiple realizers[clarification needed] and multiple mechanisms, and that the inputs and outputs of the mechanism also be multiply realizable. In short, medium-independence allows for the use of physical variables with properties other than voltage (as in typical digital computers); this is imperative in considering other types of computation, such as that which occurs in the brain or in a quantum computer. A rule, in this sense, provides a mapping among inputs, outputs, and internal states of the physical computing system.[12]

Mathematical models [edit]

Main article: Model of computation

In the theory of computation, a diversity of mathematical models of computation has been developed. Typical mathematical models of computers are the following:

- State models including Turing machine, pushdown automaton, finite-state automaton, and PRAM
- Functional models including lambda calculus
- Logical models including logic programming
- Concurrent models including actor model and process calculi

Giunti calls the models studied by computation theory *computational systems,* and he argues that all of them are mathematical dynamical systems with discrete time and discrete state space.[13]:ch.1 He maintains that a computational system is a complex object which consists of three parts. First, a mathematical dynamical system DS with discrete time and discrete state space; second, a computational setup $H=(F,B_F)$, which is made up of a theoretical part F, and a real part B_F; third, an interpretation $I_{DS,H}$, which links the dynamical system DS with the setup H.[14]:pp.179–80

See also [edit]

- Computationalism
- Computational problem
- Computability theory
- Hypercomputation
- Limits of computation
- Numerical computation

Notes [edit]

1. ^ The study of non-computable statements is the field of hypercomputation.

References [edit]

1. ^ "Definition of COMPUTATION". *www.merriam-webster.com*. 2024-10-11. Retrieved 2024-10-12.
2. ^ "Computation: Definition and Synonyms from Answers.com". *Answers.com*. Archived from the original on 22 February 2009. Retrieved 26 April 2017.
3. ^ Couturat, Louis (1901). *la Logique de Leibniz a'Après des Documents Inédits*. Paris. ISBN 978-0343895099.
4. ^ Davis, Martin; Davis, Martin D. (2000). *The Universal Computer*. W. W. Norton & Company. ISBN 978-0-393-04785-1.
5. ^ *a b* Davis, Martin (1982-01-01). *Computability & Unsolvability*. Courier Corporation. ISBN 978-0-486-61471-7.
6. ^ Turing, A.M. (1937) [Delivered to the Society November 1936]. "On Computable Numbers, with an Application to the Entscheidungsproblem" (PDF). *Proceedings of the London Mathematical Society*. 2. Vol. 42. pp. 230–65. doi:10.1112/plms/s2-42.1.230.
7. ^ *a b* Davis, Martin; Davis, Martin D. (2000). *The Universal Computer*. W. W. Norton & Company. ISBN 978-0-393-04785-1.
8. ^ Davis, Martin (2006). "Why there is no such discipline as hypercomputation". *Applied Mathematics and Computation*. **178** (1): 4–7. doi:10.1016/j.amc.2005.09.066.
9. ^ Godfrey-Smith, P. (2009), "Triviality Arguments against Functionalism", *Philosophical Studies*, **145** (2): 273–95, doi:10.1007/s11098-008-9231-3, S2CID 73619367
10. ^ Piccinini, Gualtiero (2015). *Physical Computation: A Mechanistic Account*. Oxford: Oxford University Press. p. 18. ISBN 9780199658855.
11. ^ Fodor, J. A. (1986), "The Mind-Body Problem", *Scientific American*, **244** (January 1986)
12. ^ Piccinini, Gualtiero (2015). *Physical Computation: A Mechanistic Account*. Oxford: Oxford University Press. p. 10. ISBN 9780199658855.
13. ^ Giunti, Marco (1997). *Computation, Dynamics, and Cognition*. New York: Oxford University Press. ISBN 978-0-19-509009-3.
14. ^ Giunti, Marco (2017), "What is a Physical Realization of a Computational System?", *Isonomia -- Epistemologica*, **9**: 177–92, ISSN 2037-4348

Categories: Theoretical computer science | Computability theory