Level 2

Computation, Philosophical Issues about

Matthias Scheutz, University of Notre Dame, Indiana, USA

CONTENTS

'Computation' is a cluster concept and has been characterized in many different ways (e.g. 'the execution of algorithms'). It underwrites philosophical analyses of what can be done in principle by a mechanism, and is intrinsically connected to the idea of manipulating symbols or representations by formal rules.

INTRODUCTION

0209.001 The notion of computation is undoubtedly one of the very central, increasingly influential notions of our time. It has captured the attention of researchers from many disciplines for different reasons. In cognitive science it was the capacity of computers to process information that inspired cognitive psychologists to think of cognitive functions in terms of programs and of the brain as a computer running these programs. To be able to appreciate this view of cognition and the central role of computation within it, one needs a clear understanding of what 'computation' means and what computations are.

WHAT IS COMPUTATION?

An Intuitive Perspective

0209.002 Like many widely used notions 'computation' does not have a single, clear-cut meaning, but rather, *qua* cluster concept, takes on different meanings depending on the context in which it is used. A glance in Webster's dictionary reveals the ordinary language conception of 'to compute': derived from the Latin 'com + putare' – to consider, it means something like 'to determine or to calculate especially by mathematical means'. However, this definition is rather vague and furthermore too restrictive to do justice to the variety of uses to which the notion of computation is put in computer science alone.

More to the point is defining computation as 'the execution of algorithms', which, in turn, puts the burden on the notion of *algorithm* and what 'executing an algorithm' means. Roughly speaking, an algorithm consists of a finite set of instructions, which operate on certain entities (symbols, representations of numbers, etc.) and can be *implemented* in some mechanism. To execute an algorithm then intuitively means to have the mechanism carry out the instructions for any given input in a deterministic, discrete, stepwise fashion (without resorting to random or analogue methods and devices). The mechanism goes through a sequence of atomic steps in such a way that (one or more of) these steps correspond to some instruction, for all the instructions specified by the algorithm. Note that nothing is said about the nature of the mechanism yet: it could be concrete or abstract, natural or artificial. Depending on the kind of mechanism, the algorithmic specification will take different forms: in the case of computers, it is expressed in a *programming language*; in the case of humans, instructions may be given in ordinary language (as long as the individual steps are clearly distinguishable and described at a sufficient level of precision) – just think of cooking recipes or the instructions on public phones for making phone calls. 0209.003

Computation defined as the execution of algorithms does not commit one as to what the computation is about or what it is supposed to achieve. Rather, it ties algorithmic descriptions to *mechanically realizable processes*. This leaves two issues to be addressed: first, it needs to be made clear what a mechanism is, and secondly a precise specification of the notion of algorithm is required. The following brief historical overview reveals the origin of the idea of mechanism as well as that of using representations for calculations. 0209.004

A Historical Perspective

0209.005 The history of computation traces back to Leibniz and before, when daring philosophers pondered mechanical systems that could aid humans in performing calculations, and possibly even calculate by themselves without any assistance. The first functioning mechanical calculators were built in the seventeenth century and were composed of various mechanical parts (such as gears, cogs, etc.). Leibniz, having constructed calculators himself, was one of the first to envision an application quite different from their typical commercial and military use, that of 'mechanical reasoners'. His view that calculations, in particular, and logical reasoning (i.e. thinking), in general, could be *mechanized* lies at the heart of the notion of computation as used in cognitive science today.

0209.006 Another crucial contribution to the modern notion of computation is also a product of that time (due to Descartes, Hobbes, Locke, and others), namely the idea that reasoning or, more generally, thinking involves *representations*. The mathematical practice of using marks and signs as representations in calculations became a paradigm for thought itself, as expressed by Hobbes' famous dictum that everything done by our mind is a computation (Pratt, 1987).

0209.007 Computation was, therefore, already very much tied to the idea of mechanically manipulating representations, and prototypical manipulators were found in the mechanical calculators of those days. While many attempts were made at building mechanical calculators up to the end of the nineteenth century (with varying success: e.g. see Williams, 1997), the computing capabilities of these machines remained very modest. It was only the twentieth century that witnessed major progress in the construction of computers and the conception of computing. This was largely due to two quite independent developments: (1) the thorough logical analysis of the notions 'formal system' and 'formal proof' (leading to further studies of notions such as 'effectively computable function' and 'algorithm'), and (2) the rapid progression in the engineering of electronic components (from vacuum tubes, to transistors, to integrated circuits, and beyond).

A Logico-philosophical Perspective

0209.008 In the 1930s, logicians laid the main philosophical groundwork for a well-defined formal notion of computation in their attempt to make the intuitive notion of computation, then called 'effective calculability', formally precise. Being logicians, they were solely concerned with the class of functions (over the positive integers) that can be effectively calculated *in principle* – besides, digital computers did not exist yet. Church (1936) was the first to give this class of effective calculable functions a formal characterization through a definition postulate, which later came to be known as 'Church's Thesis' (CT): 'We now define the notion ... of an effectively calculable function of positive integers by identifying it with the notion of a recursive function on positive integers (or of a λ-definable function of positive integers)' (Church, 1936, p. 356).

0209.009 While this was a first step to capture the meaning of 'computable', it was not quite satisfactory, as CT is silent about what 'effectiveness' of a calculation means. As it stands, the notion of 'effectively calculable function' implies that two ingredients are needed to understand computation: a notion of 'effective procedure or algorithm' and a notion of 'function computed by an algorithm'. The latter can be straightforwardly explicated: it is the mapping obtained by pairing all possible inputs with the corresponding outputs resulting from applying the algorithm to them. The former, however, received a satisfactory account only after Turing (1936) had introduced his machine model of a 'computer', which resulted from his analysis of the possible processes a human – what he then called 'the computer' – can go through while performing a calculation using paper and pencil applying rules from a given finite set. It was crucial to Turing's conception of computation that the human computer follow the rules 'blindly', that is, without using insight or ingenuity. In his analysis of the limitations of the human sensory and mental apparatus five major constraints for doing 'automatic computations' crystallize: (1) only a finite number of symbols can be written down and used in any computation; (2) there is a fixed bound on the amount of scratch paper (and the symbols on it) that a human can 'take in' at a time in order to decide what to do next; (3) at any time a symbol can be written down or erased (in a certain area on the scratch paper called 'cell'); (4) there is an upper limit to the distance between cells that can be considered in two consecutive computational steps; (5) there is an upper bound to the number of 'states of mind' a human can be in, and the current state of mind together with the last symbol written or erased determine what to do next.

0209.010 Turing then defined a mathematical model of an 'imagined mechanical device' that satisfies all of the above, later referred to as a 'Turing machine'

(TM) by Church. A TM consists of an unbounded tape divided into squares, each of which can hold exactly one symbol, a tape head for reading and writing symbols from a given alphabet on the squares, and a controller, which is in exactly one of finitely many states at any given time. Each computational step of the machine first involves reading the symbol under the tape head and then, depending on the current state of the controller, writing a new symbol on the square, possibly switching to another state and possibly moving the tape head one square to the left or to the right. 'The computation proceeds by discrete steps and produces a record consisting of a finite (but unbounded) number of cells, each of which is blank or contains a symbol from a finite alphabet. At each step the action is local and is locally determined, according to a finite table of instructions.' (Gandy, 1988, p. 81). This way the TM became a model of human computing, an *idealized* model to be precise, since it could process and store *arbitrarily long, finite sequences of symbols*. The TM model is also a very abstract model, for it only captures high-level processes that take place in humans when they compute (as opposed to low-level neuronal processes, for example).

0209.011 Turing intended his analysis to show that *any function computable by a human being following fixed rules can be computed by a TM*. And, furthermore, he also believed the converse, that every function computed by a Turing machine could (in principle) be computed by a human computer. Note that this equivalence *per se* does not preclude humans from being able to find answers to problems (expressed in terms of functions) which no TM can compute (e.g. using intuition).

PHILOSOPHICAL VIEWS OF COMPUTATION

Turing Computability and Beyond

0209.012 The logico-philosophical analyses of the intuitive notion of computation led to the crucial insight that different attempts to characterize it can all be proven extensionally equivalent: recursive functions, λ-definable functions, and TM-computable functions all define the same class of functions. These equivalence results are possible, because what 'computing' means with respect to any of the suggested formalisms is expressed in terms of functions from inputs to outputs, which are used as mediators in the comparison of the various classes of functions defined by the different formalisms. Later, other formalisms such as Markov algorithms, Post systems, universal grammars, PASCAL programs, as well as various kinds of automata, were also shown to give rise to the same class of functions. Hence, by CT, any of the above mentioned formalisms captures our intuitive notion of computation, that is, *what it means to compute*. (Some disagree with this conclusion, arguing that the equivalence results capture only a restricted notion of computation as shared by certain philosophers of mathematics and logicians, e.g. Sloman (1996).)

0209.013 Common to all the above computational formalisms (besides their attempts to specify formally the intuitive notion of 'computation') is their property of being independent from the physical: computations in any of these formalisms are defined *without* recourse to the nature of physical systems that (potentially) realize them. Even the TM model, which is often considered the prototype of a 'mechanical device', does not incorporate physical descriptions of its inner workings, but abstracts from the mechanical details of a *physical realization*. The first to incorporate physically motivated mathematical constraints into a formal model of computation was Gandy (1980) in his attempt to define a notion of computation for any discrete, deterministic, physical machine. He formulated five conditions to determine whether any system qualifies as a 'mechanical machine' and proved that any function computable by a discrete deterministic device (in his sense) is effectively computable and vice versa. Hence, TM-computability (i.e. effective computability) and computability by mechanical devices are equivalent notions. Some even extend the claim by suggesting that the behaviour of any finitely realizable physical system can be 'computed' (in the sense of 'perfectly simulated') by a TM (e.g. see Deutsch, 1985).

0209.014 It is not clear, however, whether computation should be equated with 'effective computability', since there are, at least in principle, imaginable computing devices that give rise to 'Super Turing computability' (i.e. compute functions that no TM can compute). An example of such a device is Turing's 'oracle machine' (O-TM), which is a TM with additional atomic operations to query an 'oracle'. The oracle itself is a device that somehow produces values of a particular (possibly TM-uncomputable) function – how the results are obtained is left unspecified. It is easy to see that any O-TM with an oracle for any uncomputable function can compute more functions than any TM. Whether such a machine could be physically realized is an open question (maybe there are physical quantities that happen to encode some TM-uncomputable function). The interesting point is simply that an O-TM would be perfectly mechanistic in the classical sense without being effective as it uses some noneffective device, namely the oracle. O-TMs, hence, drive a wedge between the notions of 'effectiveness' and 'mechanism' (e.g. see Copeland, 2000). A similar point can be made with respect to the notions of 'effectiveness' and 'algorithm'.

0209.015 There are other suggestions along the same lines coming from neural network research: it can be shown, for example, that certain neural networks (consisting of about 1000 neurones) with rational-valued connection weights between neurones can compute any TM-computable function (Siegelmann and Sontag, 1995). And if real-valued weights are allowed, they can compute any function whatsoever.

Other Construals of Computation

0209.016 Although TMs have become the canonical models of computation and permeate various academic disciplines in that role, there are other construals targeted more towards possible philosophical merits and potential practical applications of computation. Following Smith (forthcoming), for example, the following views should all be distinguished as they emphasize and capture different aspects of computation:

1. *formal symbol manipulation*: the manipulation of symbols by virtue of their formal properties (without regard to possible interpretations or semantic content);
2. *effective computability*: what can be done effectively by a mechanism;
3. *rule-following* or *execution of an algorithm*: what is involved in following rules or instructions;
4. *finite (digital) state machines*: automata with a finite set of internal states;
5. *information processing*: what is involved in storing, manipulating, and displaying information;
6. *interactive systems*: computation as interaction and communication embedded in an environment;
7. *dynamical systems*: computation expressed in the language of dynamic systems (using concepts like state space, trajectory, attractors, etc.).

0209.017 To some extent all of the above notions play a role in various disciplines (especially in computer science), but some of them are more dominant in specific intellectual areas: (1) figures mainly in philosophical debates and meta-mathematics, where (2) and (3) are tied to logical investigations; (4) is largely an engineering concept, while (5)–(7) have become increasingly important in cognitive science, the theory of complex systems, and, of course, computer science.

While the above list is far from exhaustive, it is intended to give a flavour of the wealth of different aspects the notion of computation has acquired, especially in the course of the last century. For that very reason, it is argued, we are still lacking the 'grand unified theory' (similar to physics) that can accommodate all these multiple facets, if such a theory is possible in the first place. 0209.018

Real-life Computation

Despite the theoretical success of TM-computability, computer science *qua* practice is concerned not so much with the limits of what can be computed in theory, but rather with the more modest, mundane question of what can be computed within reasonable limits (using given resources). A whole new discipline within computer science called 'complexity theory' – an offspring of the classical investigations of effective computability – is dedicated to the study of what is computationally feasible. Still other issues arise from computational practice with which the TM model, for example, can hardly cope, in particular, the need for computational systems (embedded in various kinds of devices) to continually interact with their environments: what function does an operating system compute (or the world wide web, for that matter)? According to the classical view, such questions cannot be answered easily as the underlying functions are simply not defined for inputs on which computational processes run forever. Yet, there is a strong intuition that computational processes as they occur in operating systems or web browsers do have a purpose, can accomplish certain tasks or fail at achieving them. As a consequence the notion of 'computation of a function', and with it the classical notion of algorithm, had to make room for the notion of interaction: 0209.019

> Interaction is shown to be more powerful than rule-based algorithms for computer problem-solving, overturning the prevalent view that all computing is expressible as algorithms. The radical notion that interactive systems are more powerful problem-solving engines than algorithms is the basis for a new paradigm for computing technology built around the unifying concept of interaction. (Wegner, 1997)

This paradigm shift from programs to processes renders many of the old reservations to the notion of computation obsolete, which were a consequence of taking computation to be defined solely in abstract syntactic terms thereby abstracting over 0209.020

physical realization, real-world interaction, and semantics. The new approach reveals computation, contrary to standard orthodoxy, as interactive and embodied, hence very much concerned with the constraints imposed on computational processes by the real world.

ROLE OF COMPUTATION IN COGNITIVE SCIENCE

The Midwife: Computation and the Birth of Cognitive Science

0209.021 The independence of computations (in the sense of TM- computations) from their physical realizers was one major source of attraction for cognitive psychologists in the late 1950s. The information-processing capabilities of computers, an ability thought to underlie human cognition, and the potential of computer programs to specify exactly *how* information is processed was another. Together they led to the thought that cognition, viewed as 'the processing of information', could be completely understood and explained in terms of computations: if cognitive functions *are* computations, then explanations of mental processes in terms of programs are scientifically justifiable without having to take the 'implementing' neurological mechanisms into account, similar to computers, where it is the programs implemented on the computer hardware, not the hardware itself, that explain (if not entirely, then at least for the most part) what the computer does. The *computer metaphor* implicit in this view has been summarized as the claim that 'the mind is to the brain as the program is to the hardware' (Johnson-Laird, 1988) (note that this should really read 'the mind is to the brain as *computational processes* are to the hardware' to avoid conflating the program–process distinction). Its guiding ideas eventually became so prominent (originally in psychology, later in artificial intelligence) as to assist in the birth of cognitive science and establish *the computational claim about mind*, also called *computationalism*, as a genuine research paradigm.

The Paradigm: Computation and the Computational Claim about Mind

0209.022 As with the notion of computation, computationalism is not a unified view, but construed differently by philosophers, psychologists, or neuroscientists. Various condensed slogan-like phrases such as 'the brain is a computer', 'the mind is the program of the brain', or 'cognition is computation' can be found in the literature, to name just a few. Yet, they cannot be taken at face value, for if they were read together, they would equivocate essentially distinct notions (such as program and process, mind and cognition, etc.). Furthermore, depending on their subdiscipline within cognitive science, researchers stress different aspects of computations: their information-processing capabilities, their formal nature, their control functions, their potential to have semantics, and so on.

Common to different views of computationalism are the assumptions that (1) mental processes are computational processes and (2) the same kind of relation that obtains between programs and computer hardware (i.e. the *implementation* relation) obtains between mental descriptions and brains too. It follows that cognitive functions can be described by and explained in terms of programs, and that the right level of abstraction at which to understand cognition is the computational level and not the level of the implementing mechanism (i.e. the brain), even though it might be helpful to know the functional organization and role of certain brain areas in determining what they implement. 0209.023

Computationalism has many appealing facets, especially when it comes to high-level cognition: many features related to logic and language (such as systematicity, productivity, compositionality, and interpretability of syntax or the compositionality of meaning, e.g. see Fodor and Pylyshyn, 1988) are supported by computations 'almost for free', and many mental operations on various kinds of representations such as rotating three-dimensional images, searching for a good move in a chess game, reasoning about other people's behaviour, planning a route through a city avoiding construction sites, etc. can be described computationally and implemented on computers. After all, this is what computers do: they manipulate symbol tokens (e.g. strings of bits), some of which are representations of the subject matter the computation is about. These representations, in turn, have both formal and semantic properties, of which the former are causally efficacious. Computational processes then manipulate symbols by virtue of their formal and not their semantic properties (e.g. Fodor, 1981). 0209.024

While computationalists take this to be a virtue of their approach, it is a major shortcoming for others and various arguments have been advanced to establish that formal symbol manipulation is not sufficient for human intentionality and semantics (e.g. Searle's Chinese Room, 1980) or that minds are not TM-computable (e.g. Lucas's Gödelian argument, 1961). More recently, connectionists and 0209.025

dynamicists have tried to replace the notion of computation with alternatives, arguing that the representational level of description of a cognitive system so crucial to computationalism cannot be taken for granted. In fact, most dynamicists find the symbolic/representational level of description superfluous altogether and argue instead for an explanation of cognition in terms of *dynamic systems* (e.g. Port and van Gelder, 1995).

The Method: Computation and the Simulation of Cognition

0209.026 While there are undoubtedly tendencies in cognitive science to replace the classical notions of computation, either by dynamic systems or by more adequate notions of computation (e.g. notions based on interaction, real-time constraints, etc.), even those opposed to computationalism agree at least that computation is still a valuable tool in the study of cognition (regardless of its explanatory success). In particular, computer simulations and computational models (of aspects) of cognition have become increasingly important in cognitive science. While computational models, at least to some extent, presuppose that whatever is modelled is computational, simulation models do not have to make such an assumption. Rather, they implement a computational approximation of the mathematical description of the phenomenon under scrutiny, and as long as any resultant error is within predetermined bounds the simulations are considered 'models'. In particular, they might elucidate complex dynamical relations between various parts of the simulated model, which are difficult to see (and often not 'visible' at all) from the formal, mathematical description. From trajectories through complex state spaces of dynamic systems to evolutionary processes in artificial environments, computer simulations provide a testbed for cognitive scientists to evaluate their hypotheses without always having to study them in 'real systems'. Furthermore, simulations can focus on different aspects of cognitive systems at different levels of description, they can be reproduced, slowed down, sped up, and modified in various other ways (e.g. simulating damage, disease, and various other disorders), in which no real cognitive system could be manipulated while preserving its normal functionality (obviously, ethical considerations would enter the picture here as well).

0209.027 A crucial difference between simulation and computational models is, however, that the former usually does not share all the relevant causal properties with the modelled system, whereas the latter, being a computational model of computational processes, can in principle have the right causal structure (depending on various constraints on inputs, outputs, real-time, etc.).

SUMMARY

'Computation' is a multifarious notion, which defies a single, simple characterization. Yet, it is often explicated as 'executing an algorithm', presupposing some sort of mechanism able to 'execute' instructions as specified by the algorithm. For many logicians and philosophers it was the notion of Turing machine computability that for the first time gave precise meaning to the intuitive notion of computation understood as 'blindly following rules or instructions', thereby answering the question of what 'effective calculability' is supposed to mean. The connection of 'effectiveness' and computation goes back at least to the seventeenth century, when 'calculation' was very much tied to mechanical devices. Only in the twentieth century did effectiveness, mechanism, and computation become separated, when alternative models of computations such as interactive systems were considered. Various construals of the notion of computation (such as 'formal symbol manipulation' or 'information processing') emphasize different aspects of computation, although none of them seems to capture what computation may signify in its entirety. In cognitive science, computation played a crucial role right from the start. It figured prominently in the emergence of the discipline and became the basis of computationalism, the paradigmatic view that mental processes are computational, leading to the development of computational models of cognitive functions. Even for researchers objecting to computationalism, computations can be of great utility when used to simulate cognitive processes. 0209.028

References

Church A (1936) An unsolvable problem of elementary number theory. *American Journal of Mathematics* **58**: 345–363.

Copeland BJ (2000) Wide vs. narrow mechanism. *Journal of Philosophy* **97**: 5–32.

Deutsch D (1985) Quantum theory, the Church–Turing principle and the universal quantum computer. *Proceedings of the Royal Society*, Series A, **400**: 97–117.

Fodor JA (1981) *Representations*. Cambridge, MA: MIT Press.

Fodor JA and Pylyshyn ZW (1988) Connectionism and cognitive architecture: a critical analysis. *Cognition* **28**: 3–71.
Gandy R (1980) Church's thesis and principles for mechanism. In: Barwise J, Keisler HJ and Kunen K (eds) *Proceedings of the Kleene Symposium*. New York: North-Holland Publishing Company.
Gandy R (1988) The confluence of ideas in 1936. In: Herken R (ed.) *The Universal Turing Machine: A Half-Century Survey*. Berlin: Kammerer & Unverzagt.
Johnson-Laird PN (1988) *The Computer and the Mind*. Cambridge, MA: Harvard University Press.
Lucas JR (1961) Minds, machines, and Gödel. *Philosophy* **36**: 122–127.
Port R and Van Gelder T (1995) *Mind as Motion: Explorations in the Dynamics of Cognition*. Cambridge, MA: MIT Press.
Pratt V (1987) *Thinking Machines – The Evolution of Artificial Intelligence*. Oxford, UK: Basil Blackwell.
Searle J (1980) Minds, brains and programs. *The Behavioral and Brain Sciences* **3**: 417–424.
Sloman A (1996) Beyond Turing equivalence. In: Millican PJR and Clark A (eds) *Machines and Thought: The Legacy of Alan Turing*, **vol. I**, pp. 179–219. Oxford, UK: Clarendon Press.
Siegelmann HT and Sontag ED (1995) On the computational powers of neural nets. *Journal of Computer System Sciences* **50**: 132–150.
Smith BC (forthcoming) *The Age of Significance. An Essay on the Foundations of Computation and Intentionality*, **vols I–VII**. Cambridge, MA: MIT Press.
Turing AM (1936) On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, Series 2, **42**: 230–265.
Wegner P (1997) The paradigm shift from algorithms to interaction. *Communications of the ACM* 1997.
Williams MR (1997) *A History of Computing Technology*, 2nd edn. Los Alamitos: IEEE Computer Society Press.

Further Reading

Cleland CE (1993) Is the Church–Turing thesis true? *Minds and Machines* **3**: 283–312.
Copeland BJ (1996) What is computation? *Synthese* **8** (3): 335–359.
Davis M (1958) *Computability and Unsolvability*. New York: McGraw-Hill Book Company.
Dietrich E (1990) Computationalism. *Social Epistemology* **4** (2): 135–154.
Gardner H (1985) *The Mind's New Science: A History of the Cognitive Revolution*. New York: Basic Books.
Haugeland J (1985) *Mind Design I*. Cambridge, MA: MIT Press.
Haugeland J (1996) *Mind Design II*. Cambridge, MA: MIT Press.
Herken R (ed.) (1988) *The Universal Turing Machine: A Half-Century Survey*. Berlin: Kammerer & Unverzagt.
Hopcroft JE and Ullman JD (1979) *Introduction to Automata Theory, Languages, and Computation*. Massachusetts: Addison-Wesley Publishing Company.
Searle J (1992) *The Rediscovery of Mind*. Cambridge, MA: MIT Press.
Smith BC (1996) *The Origin of Objects*. Cambridge, MA: MIT Press.
Sterelny K (1990) *The Representational Theory of Mind*. Oxford, UK: Blackwell.
Van Gelder TJ (1998) The dynamical hypothesis in cognitive science. *The Behavioral and Brain Sciences* **21**: 615–665.
Webb J (1980) *Mechanism, Mentalism, and Mathematics: An Essay on Finitism*. Boston, MA: Reidel.

Glossary

Algorithm A sequence of instructions that can be implemented and executed by some computing device.
Attractor A region in state space towards which all trajectories within a certain range lead (i.e. no trajectory within a certain range of the region will ever leave the region).
Computationalism The foundational view in cognitive science that mental processes are computational processes.
Dynamic System A mathematical system of differential or difference equations in finitely many variables, where time is the independent variable.
Effective calculability The formal transformation of strings of symbols (possibly representing numbers) by applying finitely many transformation rules.
Implementation The relation between a program description of a computation and the physical system 'executing' the computation.
Lambda-definable function A function definable in Church's -calculus.
Mechanism A physical system composed of various interacting parts, whose behaviour can be described in the physical language of mechanics.
Recursive function A function in the class of functions, which contains *three initial functions* (the 'zero function' $z(x) = 0$, the 'successor function' $s(x) = x+1$, and the 'identity function' $id(x) = x$) and is closed under *three operations on functions(composition, primitive recursion*, and *minimization*).
State space The space of all possible states (i.e. possible values of all variables) of a dynamical system.
Trajectory A sequence of states in state space.
Turing machine An abstract computing device consisting of a finite, but unbounded, tape divided into squares, and a finite controller with a read/write tape head that can read symbols from and write symbols on the tape.

Keywords: (Check)

Computation; computationalism; Turing machine; algorithm; representation