A Ubiquity Symposium Opening Statement
FINAL v5– 8/26/10

What is Computation?

Peter J. Denning

Peter J. Denning (pjd@nps.edu) is Director of the Cebrowski Institute for innovation and information superiority at the Naval Postgraduate School in Monterey, California, and is a past president of ACM. He is currently the editor in chief of *Ubiquity*.

Abstract: Most people understand a computation as a process evoked when a computational model (or computational agent) acts on its inputs under the control of an algorithm to produce its results. The classical Turing machine model has long served as the fundamental reference model because an appropriate Turing machine can simulate every other computational model known. The Turing model is a good abstraction for most digital computers because the number of steps to execute a Turing machine algorithm is predictive of the running time of the computation on a digital computer. However, the Turing model is not as well matched for the natural, interactive, and continuous information processes frequently encountered today; other models more closely match the information processes involved and give better predictions of running time and space.

The question before us -- what is computation? -- is at least as old as computer science. It is one of those questions that will never be fully settled because new discoveries and maturing understandings constantly lead to new insights. It is like the fundamental questions in other fields -- for example, "what is life?" in biology and "what are the fundamental forces?" in physics -- that will never be fully resolved. Engaging with the question is more valuable than finding a definitive answer.

This symposium is an exploration of this question by many observers. To get the discussion going, I (as *Ubiquity* editor) have composed this opening

statement. I do not intend this as a final answer, but as a reflection to stimulate commentary and reactions. The commentators may not agree with everything in this opening statement or with what the other commentators have to say. Our hope is that readers will gain a greater appreciation of pervasiveness of computation and the value of the ongoing exploration of the nature of computation.

Why Now?

Why take up this question now? There are a variety of reasons.

It was selected as the most important question facing our field by over one hundred of the two hundred participants in the Rebooting Computing Summit of January 2009 (rebootingcomputing.org). They were trying to come to grips with the identity of the computing field as they worked to attract young people and collaborate with others in other fields.

It addresses the issue of our core identity, which has been under stress in the past decade because of job growth, outsourcing, expansion in the number of fields affected by computing, and internal tensions between various elements of the computing field.

Many of us desire to be accepted as peers the "table of science" and the "table of engineering". Our current answers to this question are apparently not sufficiently compelling for us to be accepted at those tables.

We need to answer friends and colleagues in other fields who claim they have discovered natural computational processes and look to us for insights in how they work, what algorithms might be behind them, and how we might collaborate to discover more.

The term "computational thinking" has recently become popular [win06], even though it has been used inside the field since the beginning. We are discovering that neither we in the field nor our friends outside agree on what this term means. Future education and research policies depend on the answer. We need a better answer.

History of the Term Computation

Our review of the history of computer science reveals an interesting progression of definitions for computer science [den08]:

Study of automatic computing (1940s)
Study of information processing (1950s)
Study of phenomena surrounding computers (1960s)
Study of what can be automated (1970s)
Study of computation (1980s)
Study of information processes, both natural and artificial (2000s)

Over time, the definition of computer science has been a moving target. These stages reflect increasingly sophisticated understandings of computation.

In the 1930s, Kurt Gödel [god34], Alonzo Church [chu36], Emil Post [pos36], and Alan Turing [tur37] independently gave us the first definitions of computation. Gödel defined it in terms of the evaluations of recursive functions. Church defined it in terms of the evaluations of "lambda expressions", a general notation for functions. Post defined it as series of strings successively rewritten according to a given rule set. Turing defined it the sequence of states of an abstract machine with a control unit and a tape (the Turing machine). Influenced by Gödel's incompleteness theorems, Church, Turing, and Post discovered functions that could not be evaluated by algorithms in their systems (undecidable problems). Church and Turing both speculated that any effective procedure could be represented within their systems (the Church-Turing thesis). These definitions underlay the information process notion of computing.

In the time that these men wrote, the terms "computation" and "computers" were already in common use, but with different connotations from today. Computation meant the mechanical steps followed to evaluate mathematical functions. Computers were people who did computations. In recognition of the social changes they were ushering in, the designers of the first digital computer projects all named their systems with acronyms ending in "-AC", meaning automatic computer -- resulting in names such as ENIAC, UNIVAC, and EDSAC.

The standard formal definition of computation, repeated in all the major textbooks, derives from these early ideas. Computation is defined as the execution sequences of halting Turing machines (or their equivalents). An execution sequence is the sequence of total configurations of the machine, including states of memory and control unit. The restriction to halting machine is there because algorithms were intended to implement functions: a nonterminating execution sequence would correspond to an algorithm trying to compute an undefined value.

The famous ACM Curriculum 1968 [acm68] was the first curriculum recommendation in computer science. It translated these formal definitions into practice by defining computation as information processes generated by algorithms. They said the field consists of three main parts: information structures and processes, information processing systems, and methodologies.

Around 1970, Edsger Dijkstra began to distinguish algorithms from computations. An algorithm, he said, is a static description, a computation the dynamic state sequence evoked from a machine by an algorithm. The computation is the actual work. He wanted to constrain the structure of algorithms so that the correctness of their computations could more easily be proved. With this he launched the structured programming movement.

The computer science formalists were not the only ones interested in a mathematical definition of computation. In the OS (operating systems) world, the process (short for computational process) became a central concern [dev66, dij68, cod73]. A process was intended as an entity containing the execution of an algorithm on a machine. Dennis and Van Horn defined a process as "locus of control through an instruction sequence" [dev66]. Coffman, Denning, and Organick defined process as the sequence of states of processor and memory for

a program in execution [cod73, org73]. The process abstraction enabled time-sharing: processes could be suspended from the processor and resumed later. The OS interpretation of process differed from the formal interpretation in one key point: An OS process could be intentionally nonterminating.

During the 1960s there was considerable debate on the definition of computer science. Many were concerned that formal definition of computation was too restrictive; for example, the new field of artificial intelligence was not obviously computational. Newell, Simon, and Perlis proposed to remedy this by broadening to include all "phenomena surrounding computers" [nps67, per62]. At the same time, many were also concerned about whether the word science in the title was deserved. Economics Nobel Laureate Herb Simon claimed that some fields of study, such as economics and computer science, were sciences even though they did not study natural processes [sim69]. Because they encompassed new phenomena that most people considered computational, these new definitions were widely accepted. This was a fundamental shift in the understanding of computation, which now became pegged to the activities in and around computers rather than to the presumed algorithmic nature of information processes.

A few years later, the COSERS project team, led by Bruce Arden, tied the definition of computation to a concern for automation [ard83]. This connected computation even more firmly to the actions of machines.

In the mid and late 1980s, the computational science movement, which was backed by scientists from many fields, claimed computation (and computational thinking) as a new way of doing science [def09, der09]. Supercomputers were their main tools. But now computation was more than the activity of machines; it was a practice of discovery and a way of thinking.

Finally, in the 1990s, scientists from natural science fields started to claim they had discovered information processes in their deep structures [den07]. David Baltimore argued this for biology [bal01] and those working on quantum computing argued it for physics [dav10]. Now computation is seen as a natural process as well as an artificial one. This is a serious challenge to the tradition of definitions tying computation to computers.

New Developments

Since the basic definition of computation was established, there have been three significant developments that call for rethinking the basic definition. All three indicate classes of processes that most people agree are computations, but which do not fit with the basic Turing definition.

Interactive computing. Many systems, such as operating systems, Web servers, and the Internet itself, are designed to run indefinitely and not halt. Halting is an abnormal event for these systems. The traditional definition of computation is tied to algorithms, which halt. Execution sequences of machines running indefinitely seem to violate the definition. Goldin, Smolka, and Wegner [gsw06] assembled a book of 18 contributions by authors who thought interactive computing to be computation, even though it didn't fit the standard formal

definition. The term "reactive system" is often used for a system that continues its operation indefinitely and responds to stimuli from the environment. A proposed solution to the definition problem is to expand the definition to include reactive systems as well as algorithmic computing machines. (I personally prefer the term "interactive system" to "reactive system" because interactive also allows the system to generate output signals and not just react to incoming signals.)

In discussing games, James Carse said: "A finite game is played for the purpose of winning, an infinite game for the purpose of continuing the play." [car86] When applied to our situation, his insight highlights the fundamental difference between computations from algorithms (they are finite games) and computations that continue forever (they are infinite games)

Natural information processes. Leading thinkers in various science fields have declared that they have discovered information processes in nature. The most conspicuous of these claims is in Biology, where DNA is seen as an encoded representation of a living organism and DNA translation is an information process that transforms the code into amino acids [bal01]. Similar claims are coming from physicists who see natural information processes behind quantum mechanics and other natural phenomena. Perhaps the most sweeping version of this claim is Wolfram's [wol02]; he believes that all of nature is an information process (best described by cellular automata) even though we do not know (or may never know) the algorithms that generate natural processes. Computer science has been challenged to redefine computation in a way that accommodates these discoveries.

Continuous information processes. Turing machines are discrete entities that work with finite strings of symbols from a finite alphabet. This definition excludes analog computing, which was very important in the 1920s and continues in some electrical engineering specialties today. In the 1920s, Vannevar Bush developed the differential analyzer, a machine of gears, levers, and rotating shafts that could solve differential equations. Electrical engineers developed their own versions of analog differential equation solvers; these technologies are still in use today. Why is solving a differential equation on a supercomputer a computation but solving the same equation with an electrical network is not?

Most of science and engineering deals with a continuous world. The mathematical models (such as partial differential equations) assume continuous functions on real numbers. Many optimal algorithms for making predictions with these models are formulated without reference to a Turing model and lead to more accurate predictions of computational work than can be obtained from a Turing model [trw80].

Claude Shannon (a student of Bush) said that continuous signals convey information and that his information theory applies to all such signals [shw49]; he focused on discrete (binary) signals because they were the coming wave in communication systems. Today we use his information theory for discrete computing but ignore it for continuous information signals. Software radios today use signal processing algorithms to sample and decode radio signals. Why is the action of a software radio a computation but not the action of an

Armstrong FM circuit? Paolo Rocchi (IBM, Rome Italy) has just completed a book showing that there is no clear boundary between analog and digital computation [roc10]. He challenges us to define computation in a way that encompasses both.

Rethinking the Definition

The definition of computation as an execution sequence of a computing machine under the direction of an algorithm that halts is clearly too limited. We need to rethink computation to encompass not only the traditional definition, but also interactive, natural, and continuous information processes.

The real difficulty is that we have linked the standard definition of computation to a single model, the Turing machine. The Turing model is an infinite tape inscribed with symbols that are changed by a finite state control as it moves up and down the tape. All the original computational models, such as recursive functions, lambda calculus, and rewriting rules were found to be equivalent, bolstering the Church-Turing thesis that any process anyone would be inclined to call a computation could be carried out by an appropriate Turing machine. The term "a computational model is Turing complete" means that the model can be simulated by a Turing machine. Later computational models such as Petri nets and neural networks were also found to be Turing complete, bolstering the credibility of the Church-Turing thesis.

Suppose that we dropped our insistence that the Turing model is the *unique* basis of computation? That would mean not only that we could use Turing-complete models that more clearly fit the domains we are studying, but also that we could get more accurate predictions of computational work than we could with Turing machines. Such a course of action would give us more options for designing optimal methods for solving problems.

An attractive way to preserve the obviousness of computation without tying ourselves to the Turing model is to look for a model based on information process. Perhaps Turing machines, recursive functions, lambda systems, Post systems, Petri nets, or neural networks are all manifestations of a more basic phenomenon, the information process.

Looking for a definition based on information process would be a return to our roots when the field got started in the 1940s. Unfortunately, this brings us to treacherous territory, since the term information is such an ill-defined and conflicted term. Most definitions of information involve an objective component (signs and the things represented by signs) and a subjective component (the meanings). Rocchi says that information consists of a sign (representation), a referent (the thing represented), and an observer [roc10]. The objective parts of information are in the signs and referents, while the subjective part is in the observer. How can we base a scientific field on something with such a strong subjective component?

Biologists have a similar problem with life. Life scientist Robert Hazen notes that biologists have no precise definition of life, but they do have a list of seven criteria for when an entity is living [haz07]. The observable affects of life, such as

chemistry, energy, and reproduction, are sufficient to ground the science of biology. In the same way, we can ground a science of information on the observable affects (signs and referents) without a precise definition of meaning.

A representation is a pattern of symbols that stands for something. The association between a representation and what it stands for can be recorded as a link in a table or database or as a memory in people's brains. There are two important aspects of representations: *syntax* and *stuff*. Syntax is the rules for constructing patterns; it allows us to distinguish patterns that stand for something from patterns that do not. Stuff is measurable physical states of the world that hold representations, usually in media or signals. Put these two together and we can build machines that can detect when a valid pattern is present.

A representation that stands for a method of evaluating a function is called an algorithm. A representation that stands for values is called data. When implemented by a machine, an algorithm controls the transformation of an input data representation to an output data representation. The distinction between the algorithm and the data representations is pretty weak; the executable code output by a compiler looks like data to the compiler and algorithm to the person running the code. (This is not a new observation; it was part of Church's lambda calculus [chu36].)

Even this simple notion of a representation has deep consequences. For example, there is no algorithm for finding the shortest possible representation of something [cha07].

Some scientists leave open the question of whether an observed information process is actually controlled by an algorithm. DNA translation can thus be called an information process; if someone discovers a controlling algorithm, it could be also called a computation.

Some mathematicians define computation separate from implementation. They do this by treating computations as logical orderings of strings in abstract languages, and are able to determine the logical limits of computation. However, to answer questions about running time of observable computations, they have to introduce costs representing the time or energy of storing, retrieving, or converting representations. Many real-world problems require exponential-time computations as a consequence of these implementable representations.

These notions of representation are sufficient to give us the definitions we need for computing. An information process is a sequence of representations. (In the physical world, it is a continuously evolving, changing representation.) A computation is an information process in which the transitions from one element of the sequence to the next are controlled by a representation. (In the continuous world, we would say that each infinitesimal time and space step is controlled by a representation.)

This way of discussing computation does not enlarge the class of functions that can be computed; representations and their representation-controlled transformations are Turing-complete. Its value is a model of computations that

naturally describes not only the traditional processes of computing machines, but also nonterminating processes, natural processes, and continuous processes.

Examples

Let's to a quick series of examples as sanity checks that this definition can work for the cases we described earlier.

The sequence of configurations of a Turing machine is obviously a computation in this interpretation.

The non-terminating, interactive processes of operating systems and the Web are also clearly computations in this interpretation.

DNA translation is a natural information process. The DNA is taken as "genetic code" and is composed of long strings made up of four types of base pairs. DNA translation is a rule-based process that "reads" the code and produces amino acids. Douglas Hofstadter was one of the first to notice that DNA transcription can be interpreted as Turing-machine like process [hof85].

Quantum computing represents information as "qubits" (superpositions of the "0" and "1" states of a bit) and quantum algorithms as methods of transforming qubits. In some cases, such as factoring with Schor's algorithm, a quantum computer can solve problems in polynomial time that would take exponential time on conventional computers.

Science problems represented as continuous mathematical models operating with real numbers are continuous information processes.

Conclusions

The definition proposed here refocuses from computers to information representations. It holds that representations are more fundamental than computers.

This is actually a fundamental shift. It relinquishes the early idea that "computer science is the study of phenomena surrounding computers" and returns to "computer science is the study of information processes". Computers are a means to implement some information processes. But not all information processes are implemented by computers -- e.g., DNA translation, quantum information, optimal methods for continuous systems. Getting computers out of the central focus may seem hard but is natural. Dijkstra once said: "Computer science is no more about computers than astronomy is about telescopes."

Are algorithms really the heart of computing? Or is the more fundamental and inclusive notion of representations the heart? Science is discovering information processes for which no algorithm is known; might some of those information processes have no algorithms at all?

This definition does not resolve the tortured question of "what is information?" It deals with the objective parts of information (representations and the mappings to their referents) and but does not depend on the subjective

aspect of information (the individual observer). We can proceed without solving the observer problem.

This definition of computation also supports a clear definition of computational thinking. Computational thinking is an approach to problem solving that represents the problem as an information process relative to a computational model (which may have to be invented or discovered) and seeks an algorithmic solution. The pioneers of our field used the term "algorithmic thinking" to describe how the thought processes of computer scientists differ from other sciences [per62]. In the 1980s, the term "computational thinking" was commonly used to describe the way that computational scientists approached problem solving, which they characterized as a new paradigm of science [den09]. I am concerned, however, that the conception of computational thinking as a method of problem solving may sound too close to Polya [pol56] and may not call sufficient attention the aspects that make computing unique in the world [ros06].

We have been too willing in our field to embrace major terms without clear definitions (for example, software engineering, structured programming, and cloud computing). When we do not explicitly declare that we are on a quest for a clearer definition, we allow others (and ourselves) to think we are satisfied with the imprecision. This is a paradox because we our field demands great precision -- the smallest error in a representation can invalidate all subsequent results. It surprises me that we don't collectively demand more precision in the words we use to describe who we are and what we do. I am hoping that this symposium will help us to accomplish that with one of our most fundamental notions.

Bibliography

[acm68] Atchison, William, Samuel Conte, John Hamblen, Thomas Hull, Thomas Keenan, William Kehl, Edward McCluskey, Silvio Navarro, Werner Rheinboldt, Earl Schweppe, William Viavant, and David Young. "Curriculum 68: Recommends for academic programs in computer science". *Communications of ACM 11*, 3 (March 1968), 151-197.

[ard83] Arden, B. W. *What Can Be Automated: Computer Science and Engineering Research Study (COSERS).* MIT Press (1983).

[bav10] Bacon, Dave, and Wim van Dam. "Recent progress in quantum algorithms". *Communications of ACM 53*, 2 (February 2010), 84-93.

[bal01] Baltimore, D. "How Biology Became an Information Science." In *The Invisible Future* (P. Denning, Ed.). McGraw-Hill (2001), 43-56.

[car86] Carse, James. *Finite and Infinite Games.* Random House (1986).

[cha07] Chaitin, G. *Meta Math!: The Quest for Omega.* Vintage (2006).

[chu36] Church, A. "A Note on the Entscheidungsproblem." *Am. J. of Mathematics 58* (1936), 345-363.

[cod73] Coffman, E. G., Jr., and Peter Denning. Operating Systems Theory. Prentice-Hall (1973).

[den07] Denning, P. Computing is a natural science. *Communications of the ACM 50*, 7 (July 2007), 15-18.

[den08] Denning, P. "Computing Field: Structure". In *Wiley Encyclopedia of Computer Science and Engineering* (B. Wah, Ed.). Wiley Interscience (2008).

[den09] Denning, P. "Beyond Computational Thinking". *Communications of the ACM 52*, 6 (June 2009), 28-30.

[def09] Denning, P., and P. Freeman. "Computing's Paradigm". *Communications of the ACM 52*, 12 (December 2009), 28-30.

[der09] Denning, P., and P. Rosenbloom. Computing: The fourth great domain of science. *Communications of the ACM 52*, 9 (September 2009).

[dev66] Dennis, Jack, and Earl Van Horn. "Programming Semantics for Multiprogrammed Computations." *Communications of the ACM 9*, 3 (March 1966), 143-155.

[dij68] Dijkstra, Edsger. "The Structure of THE Multiprogramming System." *Communications of the ACM 11*, 5 (May 1968), 341-346.

[god34] Gödel, Kurt. "On undecidable propositions of formal mathematics". Lectures at the Institute for Advanced Study, Princeton, NJ (1934). Documented in http://en.wikipedia.org/wiki/Kurt_Gödel

[gsw06] Goldin, D., S. Smolka, and P. Wegner (Eds.). *Interactive Computation: The New Paradigm.* Springer (2006).

[haz07] Hazen, R. *Genesis: The Scientific Quest for Life's Origins.* Joseph Henry Press (2007).

[hof85] Hofstadter, Douglas. *Metamagical Themas: Questing for the Essence of Mind and Pattern.* Basic Books (1985). See his essay on "The Genetic Code: Arbitrary?"

[nps67] Newell, A., A. J. Perlis, and Herbert A. Simon, "Computer Science," letter in *Science 157*(3795):1373–1374, September 1967.

[org73] Organick, Elliot I. *Computer Systems Organization: The B5700/B6700*. ACM Monograph Series, 1973. LCN: 72-88334.

[per62] Perlis, A. J. "The Computer in the University". In *Computers and the World of the Future* (M. Greenberger, ed.). MIT Press (1962), 180-219.

[pol56] Polya, G. *How To Solve It.* Princeton University Press (1957).

[pos36] Post, E. Finite Combinatory Processes - Formulation 1. *J. Symbolic Logic 1* (1936), 103-105.

[roc10] Rocchi, P. *Logic of Analog and Digital Machines.* Nova Publishers (2010).

[ros04] Rosenbloom, P. S. A new framework for computer science and engineering. IEEE *Computer* (November 2004), 31-36.

[shw49] Shannon, C., and W. Weaver. The Mathematical Theory of Communication. University of Illinois Press (1998). First edition 1949. Shannon's original paper is available on the Web: http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html

[sim69] Simon, H. *The Sciences of the Artificial*. MIT Press (1st ed. 1969, 3rd ed. 1996).

[tur37] Turing, A. "On Computable Numbers, With an Application to the Entscheidungsproblem." Proc. of the London Mathematical Society, series 2, 42 (1937), 230-265.

[trw80] Traub, J. F., and H. Wozniakowski. *A General Theory of Optimal Algorithms.* Academic Press (1980).

[win06] Wing, J. Computational thinking. *Communications of the ACM 49*, 3 (March 2006), 33-35.

[wof02] Wolfram, S. *A New Kind of Science.* Wolfram Media (2002).