doi:10.1093/comjnl/bxs064

Reflections on a Symposium on Computation

PETER J. DENNING[1,2,*]

[1]*Naval Postgraduate School, Monterey, California*
[2]*Editor in Chief, ACM Ubiquity*
[*]*Corresponding author: pjd@nps.edu*

ACM Ubiquity **hosted a symposium in 2010–2011 on Turing's question, 'What is computation?' The editor reflects on how the symposium was organized and what conclusions it reached. The authors showed strong consensus around the propositions that computation is a process, computational model matters, many computations are natural, many important computations are continuous, many important computations are nonterminating and computational thinking has emerged as a core practice of computing. They left open the questions of whether the Turing model is the best reference model, is computational necessarily a physical process, what is information and what is an algorithm.**

Keywords: algorithm; computation; computational model; computational process; computational thinking; information; information process; interactive computation; natural computation; nonterminating computation; physical computation; reactive computation; turing; turing computability; turing machines

1. TURING'S QUESTION

1.1. What numbers can be calculated by a machine?

That was Alan Turing's question when he wrote his famous 1937 paper about computable numbers and the Entscheidungsroblem [1]. Written at the dawn of the electronic digital computing age, his paper defined the field and dominated much of its thinking ever since.

Turing wrote with the cloud of WWII hanging over Europe. At the time, machines that would automatically calculate complicated equations were a military priority: artillery officers could not properly aim their sophisticated guns without ballistic tables. His question was timely. In Britain, an even higher priority was breaking the German Enigma code; he helped them build a machine to do that.

After the war, the business world wanted computers not only to process numbers, but to tabulate lists of names and other texts. Turing's question was transformed: What can be calculated by a machine?

Behind this question was an implicit assumption that machine processes for computation are different from human processes. Machines are mechanical; they can do some things very fast, and they operate by rigid rules. Turing's question was also trying to draw a line between machines and humans. Initially, it seemed obvious that machines were less powerful. Over time, however, we learned how to automate more human processes; and we became less sure.

Thus Turing's question was transformed to: What is computation?

The field of computer science began in the 1940s to study this question. As our understanding of computation grew and matured, we progressed to ever more sophisticated definitions of computer science:

> Study of automatic calculation (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, natural and artificial (2000s)

Even today, 75 years later, Turing's question has not been settled. Each generation tried to understand what can and cannot be computed and thought it had an answer, only to be challenged by new thinking and new developments in the next generation. The initial notion that computation is the action of a digital computer was too narrow to deal with new fields like artificial intelligence and software engineering, which examined interactions between computer programs and humans. That led to the expanded idea

that computation is the phenomena surrounding computers. Eventually that transformed into a generic question—what can be automated?—that encompassed the workings of computers as well as the human processes that used computers. In recent times, new questions about whether DNA translation is a natural computation have undermined the idea that computation is done solely by machines. Where is the machine in DNA translation?

Given that the question is never fully settled, the evolution of answers to the question at different times also gives an excellent overview of the evolution of computing as a field.

2. AN ANOMALY AND A SUMMIT

Computer science boomed for many years as the use of computers spread into every aspect of society and work. Starting suddenly in 2000, computer science enrollments began to drop, declining 50% over the next 6 years. All the while, the number of jobs for computing professionals kept growing. Many people became concerned about the widening gap between graduates and industry needs. Surveys among young people showed that computer science was seen as a low-level coding profession. How could a field with such an impact have such a poor reputation?

In 2008, with the help of an NSF grant, I organized a summit meeting called 'Rebooting Computing: The Magic and Beauty of Computer Science'. My steering committee believed that we needed to take a fresh look at what the computing field is about: we had a crisis of identity rather than a cyclical decline of interest. We wanted to regain the sense of magic and beauty that had enticed us to the field, and learn how to transmit that sense to the next generation.

The summit, held in January 2009, brought together 220 people from all sectors of the computing field, including educators and students from all grade levels. They included 30 international attendees from a dozen countries and three Turing Award winners.

There was a clear consensus in the group that we as a field lacked a clear conception of what computing is about. Without self-understanding, we cannot expect others to understand who we are. Over half of the attendees allied themselves with one of the 10 action groups concerned with the question, What is computing?

That led to a new wave of interest in the fundamental principles of computing. The words 'magic', 'joy' and 'beauty' made it into the vocabulary of professionals describing the principles of their field and into the titles of new (and very popular) courses at some universities.

3. THE UBIQUITY SYMPOSIUM

Inspired by the summit, Peter Wegner asked me, in my role as editor of *ACM Ubiquity*, whether Ubiquity could host a symposium on the summit question, what is computing? We created a plan to gather essays from a dozen leading educators, scientists and engineers to re-examine Turing's question, What is Computation?

We told our authors that a thoughtful, contemporary response to the question would help others understand why computing is so important to their lives and work. It could help rekindle the lost sense of magic and beauty of computing. We also asked them to consider how three new developments might have affected the traditional answers to the question. The three developments are:

Interactive computation. Turing's definition of computation was based on computing functions. The input was presented before the computation started, and the output was available after the finite computation finished. In contrast, operating systems and networks are based on computations that do not terminate and regularly interact with their environments.

Natural computation. Since Turing's time, computation was seen as the action of computing machines. In recent years, other fields have claimed to study information processes found in nature, and they entered into collaboration with computer scientists. Computer science is no longer a 'science of the artificial'.

Continuous computation. The new field of computational science relies on continuous models of the physical world, which are then discretized for computation. Analysts make better performance prediction when starting with their continuous models than when applying traditional discrete complexity models.

We released the symposium papers over a 14-week period beginning October 2010 and ending February 2011 [2].

4. CONCLUSIONS OF THE SYMPOSIUM

The symposium participants agreed on these points:

Computation is a process. Computation is an information process evolving from the actions of a host 'machine' under the control of an 'algorithm'. By allowing flexibility in the definitions of machine and algorithm, we can include DNA translation, and other natural information processes, in this definition.

The computational model matters. Designers seek algorithms that get the most efficient computations within the host environment. Performance predictions depend on the computational model. Some models are more suitable than others for particular problems. The Turing model, even though universal, is often not the most convenient for many domains. Different computation models lead to different approaches to design and performance prediction.

Many computations are natural. Many authors acknowledged that some natural processes behave like

computations, including DNA translation, neuronal brain function and social interactions in networks. This is a huge shift from the past, where, as recently as two decades ago, almost everyone insisted that computations were the state sequences of computing machines.

Many important computations are nonterminating. Most computations in operating systems and networks are designed to never halt. Their high-availability services interact continually with their environments. There is an emerging consensus that interactive models are fundamentally different from Turing machines. The combination of humans and machines working together can be far more powerful than either one working alone.

Many important computations are continuous. Many authors acknowledged that computations to solve continuous models of physical processes are better analyzed in terms of the continuous model rather than a discrete model. Error management resulting from discretizing the model is an important part of design.

Computational thinking had emerged as a central practice of computing. The term 'computational thinking' was first used during the 1980s computational science movement. This mode of problem solving had previously been called 'algorithmic thinking' by computer scientists. In the past decade, it has become popular in many fields as a way to describe the problem-solving processes people regularly use.

The symposium authors did not show agreements on three more questions:

Is the Turing machine the only acceptable reference model? Other, non-Turing, models of computation include recursive functions, rewriting rules, parallel-processor systems, cooperating sequential processes, dataflow graphs, neural networks, Petri nets, finite-state machines, pushdown machines and more. They are all 'Turing equivalent', meaning that the same set of functions is computable in each model. At the same time, the Turing model does not fit with many problem domains. Why are we wedded to the notion that the Turing machine is the best reference model?

Is information observable? Now that we agree computing is concerned with information processes, we must confront the fact that we do not have a consistent definition of information. What is information? How do we observe it? How can we measure it?

Is all computation physical? Many wonder if computation, like a mathematical abstraction, exists independently of the physical world. All the known examples of computation depend on some underlying physical medium that represents bits and patterns of bits. Computational state changes correspond with physical state changes. Does the power of computing machines come from the physical connection? How do the physical aspects constrain computation?

5. TWO QUESTIONS SINCE THE SYMPOSIUM

Two questions that were brought into sharper focus during the symposium remain open and are the subjects of continuing inquiries. What is information? What is an algorithm?

With respect to the first question, if computing is fundamentally the study of information processes, what is information? We have ignored this problem by concentrating on computers, but we cannot ignore it any more. Is DNA a code? What does it encode? Who is it a message from? Who is it a message to? When a computation gives a surprising result, has it generated new information or simply revealed information hidden in a sea of bits and signals?

Two recent books make some progress toward answering this question. James Gleick aims to make information real and tangible [3]. He begins with a history of humans transmitting signals with codes that include enough redundancy that the receiver can recover the original information. He dwells on how Claude Shannon in 1948 pulled all this practical wisdom together into a mathematical 'information theory'. A key tenant of the theory is that codes, senders and receivers do not need to understand the meaning of the signals and bits they are transmitting. Meaning is irrelevant to the workings of communication systems. This gives a nice, clean division of labor: the communication system gets signals to people, and the people figure out what the signals mean to them.

Gleick then follows the consequences of a theory that separates meaning from transmission in a new world of digital devices. Moore's Law shows that the relentless trend in computer and communication chips has been to increase speed and capacity by 100-fold per decade. The resulting information flood has brought on a crisis of meaning. We struggle with the frustration of too much information and search for tools to help us make sense of it.

Although Gleick was unable to offer a precise definition of information, he clearly does not agree with Shannon that meaning is irrelevant. Trying to find the meaning of information and our lives in a computational world is a very important question.

The other notable book on information is by Paolo Rocchi [4]. He inventoried and interpreted a huge variety of analog and digital devices, all designed to process information. He wanted to see what consensus they might have on what is information. He eventually found it impossible to draw a sharp dividing line between analog and digital computation. It is all processing and transmission, he concluded, regardless of whether it is continuous signals or digital signals.

But he did find a useful common factor about information. Despite many differences, everyone agreed that information includes 'signs' and 'referents'. A sign is a physical inscription or object that stands for something. A referent is an object or

concept that gives the meaning to a sign. Computational systems do not need to know the referents in order to process signs. Humans judge computations useful if the referents of the results make sense to them and guide them toward useful actions.

These two authors show us that the traditional answers are incomplete and there is still much to learn. Can we find a model for information that unifies sign and referent? How do the common functions of sensing and recording fit into a definition of information? What is the role of human or machine interpretation of some else's sign?

The other big open question is 'What is an algorithm?" Moshe Vardi recently called attention to new thinking on this question [5]. He challenged the folklore that Turing offered the Turing machine as a model of algorithms. Turing was interested in computability, not algorithms. He showed what it meant for a machine to compute a function, and he argued that Turing machines could calculate any function that a human computer could.

Vardi cites recent works of Yannis Moskovakis and Yuri Gurevich for new thinking. Moskovakis argues that an algorithm is defined in terms of a recursor, which is a recursive description built on a set of primitive operations. This definition would appeal to those who design algorithms as methods to calculate functions. Gurevich argues for a more machine-oriented approach. An algorithm is a description of an abstract state machine, where states can be any data structure, and each operation can cause only a bounded change of state. I find this definition particularly intriguing because it pulls the principle of locality into a fundamental definition of computation. Locality has long been accepted as a fundamental principle of memory behavior, but it has never been so clearly linked to effective computation.

These three authors show us that traditional answers are incomplete and there is still much to do to get us to a much more precise definition of algorithm than we now have.

6. CONCLUSIONS

Not everyone will have the time to look into these fundamental questions. In the symposium, Peter Freeman noted an analogy: not everyone has the time to consider the question of 'What is democracy' but it is comforting to know that some are looking at this and occasionally sharing their findings with the rest of us. That we in computing keep making new discoveries by re-examining our fundamental questions is healthy and a sign of depth. We may struggle with these questions; but the struggle is perhaps more important than finding final answers. Through the struggle, we learn more not only about Turing's question, but also about ourselves.

ACKNOWLEDGEMENTS

We are grateful to ACM, and to the contributing authors, for their agreements to include the papers from the Ubiquity symposium as an appendix to this reflection. The appendix is copyright by ACM, Inc., 2010–2011, and is included here by permission of ACM.

REFERENCES

[1] Turing, A.M. (1937) On computable numbers, with an application to the Entscheidungsproblem. *Proc. Lond. Math. Soc.* Ser. 2, **42**, 230–265.

[2] Association for Computing Machinery. Ubiquity, the online peer-reviewed magazine about the future of computing. http://ubiquity.acm.org/symposia.cfm

[3] Gleick, J. *The Information: A History, A Theory, A Flood*. Vintage (2012).

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

[5] Vardi, M. (2012) What is an algorithm? ACM *Commun.* **55**, 5.