Ubiquity Symposium

What is Computation?

The Enduring Legacy of the Turing Machine

by Lance Fortnow, Northwestern University

Editor's Introduction

In this eighth article in the ACM Ubiquity symposium discussing, What is Computation?, Lance Fortnow invokes the significance of the Turing machine in addressing the question.

Editor

Ubiquity Symposium

What is Computation?

The Enduring Legacy of the Turing Machine

by Lance Fortnow, Northwestern University

Supported in part by NSF grants CCF-0829754 and DMS-0652521

This is one of a series of Ubiquity articles addressing the question "What is Computation?" Alan Turing[1], in his seminal 1936 paper *On computable numbers, with an application to the Entscheidungsproblem* [4], directly answers this question by describing the now classic Turing machine model. The Church-Turing thesis is simply stated.

> Everything computable is computable by a Turing machine.

The Church-Turing thesis has stood the test of time, capturing computation models Turing could not have conceived of, including digital computation, probabilistic, parallel and quantum computers and the Internet. The thesis has become accepted doctrine in computer science and the ACM has named its highest honor after Turing. Many now view computation as a fundamental part of nature, like atoms or the integers.

So why are we having a series now asking a question that was settled in the 1930s?

A few computer scientists nevertheless try to argue that the thesis fails to capture some aspects of computation. Some of these have been published in prestigious venues such as Science [2], the Communications of the ACM [5] and now as a whole series of papers in ACM Ubiquity. Some people outside of computer science might think that there is a serious debate about the nature of computation. There isn't.

Peter Denning wrote the introductory article "What is Computation?" for the Ubiquity series [1]. He raised three questions about the applicability of the thesis.

1. Turing formulated his definition for the specific case of evaluating functions (computing numbers). Those computations terminate after a finite number of steps. Today we have

[1] In this article I will talk about Turing's computation model and the Church-Turing thesis. Who deserves credit for the computation model and the thesis remains an interesting discussion that is irrelevant for the points I make in this paper. The interested reader is referred to Robert Soare's careful treatment of the history [3].

many computations that persist indefinitely, such as operating systems and web services. Does Turing's definition cover this case?

2. Analog computation seems to be an historical loose end. The analog computer uses continuous signals to represent physical quantities. Does Turing's definition cover these computations?
3. Today we have scientists discussing natural information processes and possible natural computations. Turing's definition seems to apply to computations generated by machines. Does his definition cover computations from natural processes?

In none of these cases do we need to modify Turing's definition or even make new arguments.

Let's start with Denning's third question. Turing defined his machine not to model physical digital computers, which didn't exist in 1936, but to model the thinking process of a human. But even if Turing didn't think about natural computational processes, these processes are still nicely captured by his model. In fact the argument that there are natural computational processes rest on the observation that these processes act like Turing's model.

Denning's first question about finite versus infinite behavior stems from a common confusion between the notions of computation and computable numbers. As Turing states in his beautifully written justification of the Turing machine [4, Section 9]

> No attempt has yet been made to show that the "computable" numbers include all numbers which would naturally be regarded as computable. All arguments which can be given are bound to be, fundamentally, appeals to intuition, and for this reason rather unsatisfactory mathematically. The real question at issue is "What are the possible processes which can be carried out in computing a number?"

Computation is about process, about the transitions made from one state of the machine to another. Computation is not about the input and the output, point A and point B, but the journey. Turing uses the computable numbers as a way to analyze the power and limitations of computation but they do not reflect computation itself. You can feed a Turing machine an infinite digits of a real number (Siegelmann [2]), have computers interact with each other (Wegner-Goldin [5]), or have a computer that perform an infinite series of tasks (Denning [1]) but in all these cases the process remains the same, each step following Turing's model.

Denning's second question asks about analog computers that use continuous signals to represent a possibly infinite number of states and transitions. Turing anticipated that issue as well [4, Section 9].

> The behavior of the computer at any moment is determined by the symbols which he is observing and his "state of mind" at that moment. We may suppose that there is a bound B to the number of symbols or squares which the computer can observe at one moment. If he wishes to observe more, he must use successive observations. We will also suppose that the number of states of mind which need be taken into account is finite. The reasons for this are of the same character as those which restrict the number of symbols. If we admitted an infinity of states of mind, some of them will be "arbitrarily close" and will be confused. Again, the restriction is not one which seriously affects computation, since the use of more complicated states of mind can be avoided by writing more symbols on the tape.
>
> Let us imagine the operations performed by the computer to be split up into simple operations which are so elementary that it is not easy to imagine them further divided. Every such operation consists of some change of the physical system consisting of the computer and his tape. We know the state of the system if we know the sequence of symbols on the tape, which of these are observed by the computer (possibly with a special order), and the state of mind of the computer. We may suppose that in a simple operation not more than one symbol is altered. Any other changes can be set up into simple changes of this kind.

One cannot distinguish between two analog processes that end with results closer than the inherent limits of measurement. So one can take any analog process and simulate it by a discrete process that breaks it into a finite number of transitions approximated to an appropriate finite precision that will compute a result within that limit.

Peter Denning in his paper [1] offers the following "alternative" definition of computation.

> A representation is a pattern of symbols standing for something. A representation that stands for a method of evaluating a function is called an algorithm. A representation that stands for data is called a value. An information process is a sequence of representations. A computation is an information process in which the transitions from one element of the sequence to the next are controlled by a representation.

Denning proposed this model to "rethink computation to encompass not only the traditional definition, but also interactive, natural and continuous information processes" [1] but the alternative doesn't add anything new. Certainly the Turing machine fits this model Denning argues that representations can be infinite but again by Turing's argument we can assume that representations are finite. Although Denning doesn't define about what he means by

"controlled by" the only reasonable model we have for "controlled by" is the process defined by Turing. So Denning's new definition is just Turing machines reformulated.

So yes Virginia, the Earth is round, man has walked on the moon, Elvis is dead and everything computable is computable by a Turing machine.

About the Author

Lance Fortnow is a professor of Electrical Engineering and Computer Science at Northwestern University specializing in computational complexity and its applications to economic theory. He also hold a courtesy appointment at the Kellogg Graduate School of Management and an adjunct professorship at the Toyota Technological Institute–Chicago.

Acknowledgments

Thanks to Bryan Pardo and Bill Gasarch for useful comments, and Peter Denning for a spirited discussion on the nature of computation.

References

[1] P. Denning. What is computation? *Ubiquity*, November 2010. http://ubiquity.acm.org/article.cfm?id=1880067.

[2] H. Siegelmann. Computation beyond the Turing limit. *Science*, 268:545–548, April 1995.

[3] R. Soare. Turing oracle machines, online computing, and three displacements in computability theory. *Annals of Pure and Applied Logic*, 160(3):368 – 399, 2009. Computation and Logic in the Real World: CiE 2007.

[4] A. Turing. On computable numbers, with an application to the Etscheidungs problem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[5] P. Wegner and D. Goldin. Computation beyond Turing machines. *Communications of the ACM*, 46(4):100–102, 2003.

DOI: 10.1145/1895419.1921573

