Ubiquity Symposium

What is Computation?

Is the Symposium Question Harmful?

Consideration of the Question 'What is Computation?' Considered Harmful

by Peter A. Freeman
Georgia Tech

Editor's Introduction

In this fifteenth piece to the Ubiquity symposium discussing What is computation? Peter A. Freeman considers the theoretical and practical aspects of the question.

Peter J. Denning
Editor

Ubiquity Symposium

What is Computation?

Is the Symposium Question Harmful?

Consideration Of The Question 'What Is Computation?' Considered Harmful[1]

by Peter A. Freeman
Georgia Tech

Actually, such consideration can be useful—but only if done in a grounded context and in moderation. Let me explain.

Peter Denning opens this symposium with an excellent survey of many of the published attempts to define computation. He outlines the history of our attempts to do so and I largely agree with the definition that he puts forward—that computation is more about representations than it is about algorithms (or the mechanisms that execute them, usually called computers, but as we now understand there are other possible mechanisms)[2]. However, trying to define computation in a way that is valid for all time will likely result in frustration and/or a less-than-practical result.

To most of those whose daily work is ultimately associated with computers ('IT workers' in the sense of Freeman and Aspray[3]) the question is rarely asked. What they may ask, however, are questions such as: What practical computations can we perform on this particular device? If we change this representation, will the computation run faster, or take less working memory? Can I do this task on that computer system? What is computer science? What is computational science? What is e-science? What is information science? What is software engineering? And so on.

These questions ultimately depend on a coherent and accepted answer to the question "What is computation?" "How so?" you might ask. And "Why should I care?"

Computer scientists who devote their research to the theory of computation (broadly speaking) use the tools and methods of mathematics and logic to develop precise answers to questions such as "What is computation?" and many related questions. This forms the basis for designing algorithms (although most algorithms are developed without explicit knowledge of the theoreticians' results) and the intellectual basis for a good bit, but certainly not all, computer science and computing research. Some computer scientists who are not known as theoreticians (including myself) also opine on the answer to "What is

[1] I hope that a few readers under the age of 60 have read Dijkstra's famous paper on goto's with a similar title and appreciate my humor!
[2] The duality of representation (states) and processes (algorithms) argues that they should be interchangeable, even if we don't know what the process is.
[3] Freeman, P. and Aspray, W. *The Supply of Information Technology Workers in the United States*, Computer Research Association, Washington, D.C, 1999.

computation?" That, in turn, impacts what we do in our research, education, and other computing-related activities (including developing policies of various sorts).

So, my first—and perhaps most important—observation on the connection between what is inherently an abstract and non-practical question and the compelling questions you may face in your daily work in computing is that it forms a very important part of the intellectual basis for what we all do that in any way touches the very broad field of computing. Just as the abstract principles of democracy—consent of the governed, majority rule and minority rights, fairness in voting, and so on[4]—form the basis for much of what we do on a daily basis, we may not always be aware that what is meant by computation shapes and conditions much of what we do in the computing field. At the same time, what we do is shaping the meaning of "computation."

My second observation is more direct. For anyone engaged in computing education, it is essential to understand explicitly the inherent nature of what we are teaching. Further elaboration should be unnecessary.

The third observation is also direct, although perhaps of less urgent connection. Anyone involved in research and/or advanced development certainly needs to be aware of the intellectual basis of our field so that they can think broadly about their research questions. Sometimes this research involves explicit consideration of the theories of computation and their results.

I'm sure there are other reasons why consideration of the question "What is computation?" may *not* be harmful. Let me turn now to the other part of my admonition—*such consideration should be done in moderation.*

The most important reason to not spend too much time pondering the fundamental nature of what we do should be obvious. Unless you are a theoretical computer scientist or a professor developing new courses and curricula, such consideration will simply take away from the time you need to spend on your immediate tasks. I hasten to add, however, that everyone involved in computing should pause from time to time and consider the question.

That leads to a related observation. As we all know, our field is extremely dynamic. Even the fundamental, intellectual foundations are evolving—that is the focus of this symposium. When I implied above that it is important to have "a coherent and accepted answer to the question of 'What is computation?'" I chose my words carefully. I did not say that it had to be precise, rigid, or even "accepted" by everyone. It is important to have a common understanding of fundamentals in order to make progress in any field, but a rigid "standard" that is adopted too early is almost always an impediment to progress. Just think of where we would be today if computer science had remained merely a branch of mathematics or engineering or experiment-based science.

In closing, I want to offer a broader and still useful interpretation of the thinking of Professors Newell, Perlis, and Simon on the subject of "What is Computer Science?" In their famous Letter in *Science*[5] the definition they give ("computer science is the study of computers") should be read in the context of their discussion that asserts that "Phenomena breed sciences" and their reference to "the varied, rich, complex

[4] See, for example, http://www.america.gov/st/democracy-english/2008/May/20080609194207eaifas0.8688013.html

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

phenomena surrounding computers." As a graduate student at the time at CMU, working and interacting with all three men, I know that they certainly considered topics as diverse as theory of computation, compiler construction, artificial intelligence, and software engineering as part of computer science. The subsequent history of computer science and computing more generally at CMU to this day strongly supports the continued viability of this definition.

This also highlights the difference between asking, "What is computation?" and asking, "What is computer science?" In that sense, Peter Denning is correct in asking the broader question and illustrating it with examples from the life sciences and elsewhere. The practical question for those trying to advance computer science is whether or not the fundamentals of computer science suffice to answer the question "What is computation?" As this symposium of *Ubiquity* illustrates, it will largely be computer scientists and similar fellow travelers that consider and try to answer the question. Everyone else involved in computing should focus most of the time on what they are trying to do and not on what the philosophical basis for it may be!

About the Author

Peter A. Freeman (peter.freeman@mindspring.com) is emeritus Founding Dean and Professor at Georgia Tech and Former Assistant Director of NSF for CiSe.

