What is Computer Science? Constraints on Acceptable Answers*

Selmer Bringsjord
Department of Cognitive Science
Department of Computer Science
Rensselaer Polytechnic Institute (RPI)
Troy NY 12180 USA
selmer@rpi.edu

rough draft 8.11.06

*I'm greatly indebted to Bill Rapaport, for not only investigating PCS, but specifically for giving a provocative presentation at the 2006 North American Computing and Philosophy Conference on how he teaches PCS. (His course, continuously updated, is at http://www.cse.buffalo.edu/∼rapaport/philcs.html.) I wrote this first draft of this note while listening to this presentation.

Philosophy of Computer Science (PCS) is in its infancy. For this reason alone, the problems and questions that constitute PCS are far from determined. However, there should be no doubting that one of the driving questions in PCS is this one:

$\mathcal{Q}$ What is computer science?

This note presents and defends four constraints on acceptable answers to $\mathcal{Q}$.

The first constraint is that any answer to $\mathcal{Q}$ that includes some such notion as "Whatever computer scientists actually do." is unacceptable. This is easy to see: Suppose that tomorrow, due to some Earth-wide cognitive fluke (I leave the etiology to you: aliens do something to all the relevant brains, there is a massive coincidence — whatever), all computer scientists decide to restrict all their activity to abstract and physically realized finite state automata (FSAs), and eschew mention (let alone use) of information-processing models and artifacts able to process functions from $\mathbf{N}$ (natural numbers) to $\mathbf{N}$ beyond those that FSAs can handle. Would we now say that computer science is a field restricted to information processing that FSAs can muster, and nothing more? Clearly not. To bring the point home, suppose that a fluke like this strikes not computer scientists, but physicists. (Philosophy of physics is of course quite mature.) Specifically, imagine that physicists suddenly became constitutionally unable to consider modeling the physical world in ways outside of the Newtonian paradigm. Now suppose that, after this fluke occurs, and while it's still in force, a philosopher begins to systematically consider a thought experiment involving clock synchronization and simultaneity that involves trains arriving in stations and observers watching their watches.[1] We can leave the details of the (1905) thought experiment aside, since the important point (as you no doubt already know) is that the thought experiment cannot be modeled by Newtonian physics. Would we say that this philosopher isn't doing philosophy of physics, because physicists don't do anything beyond the Newtonian scheme? I should think not. In fact, we are likely to say that the philosopher here is a seminal philosopher of physics.

Now the second constraint. McCarthy has said that computer science is the science of "computational procedures." Shapiro has said that computer science is the "science of procedures." (Well, based on what I heard in Bill Rapaport's NA-CAP talk, these thinkers have said this — but I may have heard wrong.) No answer of this type,[2] can be palatable. The reason is simple: Computer science, as a matter of mathematical fact, can be taught exclusively from the standpoint of logic. Instead of procedures, we can talk only of first-order logic, and proofs and interpretations. We don't need to write procedural (= imperative) or functional computer programs, or formulate instructions for Turing machines, register machines, and so on. It's purely adventitious that our educational system cements, year in and year out, the idea that computation

[1] Einstein is in the Library of Living Philosophers, note.

[2] Unless it admits of recherché interpretations of 'procedure,' according to which the term doesn't mean anything like a step-by-step recipe or algorithm. But then why use the term in the first place?

is to be identified with step-by-step procedures or algorithms. (And make no mistake, the cementing has happened. Students sometimes ask me, paralyzed, "But how can you use your logicist approach to sort n numbers?") Things could have gone differently. The term 'procedure' (and all synonyms) could be absent from the curriculum, at all times. And yet students would still learn all the math, and all the programming, and so on.

The third constraint is that no such answer as that computer science is the study and application of computation by standard Turing machines (or their equivalents) is acceptable. The reason is clear: Theoretical computer scientists have long been investigating machines that can compute (or better: *hyper*compute) functions from $\mathbf{N}$ to $\mathbf{N}$ that aren't Turing-computable. After all, the Arithmetic Hierarchy is part and parcel of computer science.

The fourth and final constraint flows from the third, and is this: No acceptable answers to $\mathcal{Q}$ can entail that computer science is not, in significant part, a *formal* science. The formal sciences also include pure mathematics, formal logic, some technical parts of philosophy, mathematical psychology, and parts of economics. I understand that some hold that computer science is a field within engineering. (There is little question that *parts* of computer science fall within engineering. The claim here is that computer science falls *within* engineering.) This cannot be true. While engineering routinely partakes of mathematics, and while frontier-breaking engineering can be highly mathematical, engineering is clearly distinguished by the fact that it is inseparably linked to applications, to physical things in the physical world. Not so computer science. There, seminal problem solving can consist in making a formal advance only. Others may apply this advance, yes. But when the advance has intrinsically formal value, we are talking about a different animal. It may be a computer scientist who settles P=?NP, after perhaps a lifetime of formal investigation. Such a person will have made a contribution independent of any applications that ensue.