A Vision for the Theory of Computing: Connections and Ideas

Manuel Blum, Pittsburgh, PA, May 31, 2011

I. Which scientific ideas will make it big and which not? I am not particularly skilled at crystal ball gazing. What success I've had in producing good researchers comes mainly from recruiting bright students, giving them free reign, occasionally seeding ideas, but mainly encouraging them to explore connections, chart new directions, think out of the box, and push beyond incremental work. I have encouraged their ideas, whether those ideas bubble up out of their own heads, mine, other faculty, courses, books, whatever.

I'm an idea optimist. I believe that the best ideas eventually bubble to the top. The most useful thing we as mentors and directors can do, in my opinion, is to encourage their emergence. I'm somewhat of an idea elitist in that I think some ideas are much better than others, and that the best ideas can/should/will be pursued more deeply than anyone might have imagined.

To make the point, consider a couple ideas from physics: Galileo (1564-1642) pointed out that one who wakes up in the hold of a becalmed ship cannot tell whether that ship is docked or moving at constant velocity. This reasoning extends neatly to Einstein's description of a scientist in a rocket ship not being able to tell if the rocket is docked on earth or accelerating uniformly in space, which in turn leads to a precise prediction of the amount by which light is bowed by the sun. The point is that good ideas, like Galileo's, percolate through the ages, in this case from Galileo to Einstein.

For another example, around the same time, Fermat (1601-1665) explained the refraction of light when it enters water by arguing that (1) light travels slower in water than in air, and (2) when traveling from a point A in Air to a point W in Water, light takes whatever path takes least time. This is known as Fermat's principle. It's as if light took all possible paths and all except the fastest canceled out. This is a fundamental idea in Feynman's physics, one that bubbled up from Fermat to Feynman.

The point is that good ideas like Galileo's and Fermat's bubble up through the ages, gradually pushing incorrect ideas aside. (Descartes (1596-1650), for example, argued incorrectly that light like sound travels faster in water than in air.)

II. Computer science ideas in particular. A significant swath of modern (theoretical) computer science has to do with *formalizing the informal*. CS theorists in particular like to nail down definitions that are completely precise, rather in the tradition of the mathematical formalization of random variables. When it comes to formalizing informal notions, counting steps is one of the hardest things to get right, as it seems so very model dependent. RAMs, Turing Machines, what have you, all lead to slightly different counts of steps. But once the concept is nailed down in the rough yet precise way it has been nailed -- counting the number of steps taken by an algorithm precisely to within some multiplicative constant or (at worst) polynomial factor -- the definitions of theoretical computer science that build on it are completely precise.

What is the evidence that step-counting is hard to formalize? Gauss certainly counted steps, though he never published anything about it, most likely because he had not formalized the concept. In 1957, the introduction to Ernst Guillemin's textbook on linear circuit theory described Gaussian Elimination with a 4x4 example, and pointed out that Gauss's method works much better than the then-standard method based on exponential time computation of determinants. But Guillemin did not count steps. Instead, he merely said that while both methods (Gaussian Elimination and Determinants) appear to work equally well on the 4x4 example, Gaussian Elimination worked far better on large examples. He did not quantify; he had no mathematical way to express the improvement. The point is that neither Gauss nor anyone else well up into the 20th century had any idea how to formalize step-counting.

Once the concept of counting steps got formalized, most especially by taking advantage of the (enormously important) big O notation, a great many other concepts based on step-counting could be formalized. A tiny sample of such concepts includes: P vs RP vs NP vs PSPACE; pseudo-random functions/sequences; goodness of approximation algorithms; online algorithms and competitive ratio; PAC learning; probabilistic interactive zero knowledge proofs.

III. Extending computer science ideas in new directions. I am especially interested in connecting TCS with other sciences. Consider the special case of Cognitive Psychology. Here another really hard-to-formalize notion akin to step-counting, more like a notion of human-computation-steps, could be formalized. .

A fundamental problem for both the Science of Cognitive Psychology and Computer Science is to construct a model of an intelligent mechanism. For example, Polya's *How to Solve It* has many good suggestions, but no (even informal) model of the intelligent student to whom it is directed. The Dutch logician Hans Freudenthal suggested a language, LINCOS, for teaching an intelligent alien about our world, but no model for what such an intelligent alien might be. Formalizing intelligent mechanisms for these examples would enable to measure the effectiveness of Polya's and Freudenthal's books, and would be a great contribution for understanding intelligence.

The Cognitive Psychology approach seeks to create a model of the human brain. The CS (Science of the Artificial) approach looks for an artificial model of intelligent mechanism for solving these problems. The latter mechanism may or may not be brain-like (though in all likelihood it will be a computer model of some sort).

Is a computer model of an intelligent mechanism/brain reasonable? I'm surprised by how much more like an automaton I am than I had ever imagined. When I input a request to Google, I generally find that someone else has made the same request. If I have a problem with my iPhone, I generally find that someone else had the same problem, and described it in roughly the same way. Not only that, but someone else answered it, explaining how to fix the problem. While it will not be easy, I do think it possible to create a model for an intelligent mechanism that is capable of understanding Freudenthal's LINCOS and then understanding and making use of Polya's *How to Solve It.*

In 1969, Herb Simons described Computer Science as "The Science of the Artificial." And I thought: What kind of science is that? In what way is a (computer) science of the artificial (theoretical or not) of a stature and importance comparable to a science of the natural, like physics, biology, or chemistry? The real sciences are about the real world, not about artificial saccharin-like worlds. Of course, in the sciences, a great amount of work is done with imaginary worlds, if only because we don't know precisely how the real world works, and so our models, until we get them right, if ever, are imperfect. But at least the intent is to understand the real world. That said, an artificial world can be useful. It can be defined precisely, enabling us to say precise things, compute precise consequences, compare to reality, and so on. It can have complex behavior even in the case of (relatively) simple designs. In the case of artificial intelligence, it does not have to work the way the real world (natural intelligence) works. As a tool, it can work better.