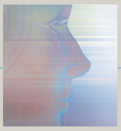


BY JEANNETTE M. WING

FIVE DEEP QUESTIONS IN COMPUTING

Even if they seem unanswerable, just trying to answer them will advance the field's scientific foundations and help engineer the systems we can only imagine.

The field of computing is driven by boundless technological innovation and societal expectations. The field runs at such a maddening pace that we barely have time to pause and enjoy the ride. The time between an ingenious idea springing from a research laboratory or coffeehouse conversation and its emergence as a product or service is so short and the frequency of the commercialization cycle of ideas so great that we rarely stop to savor even our own successes.

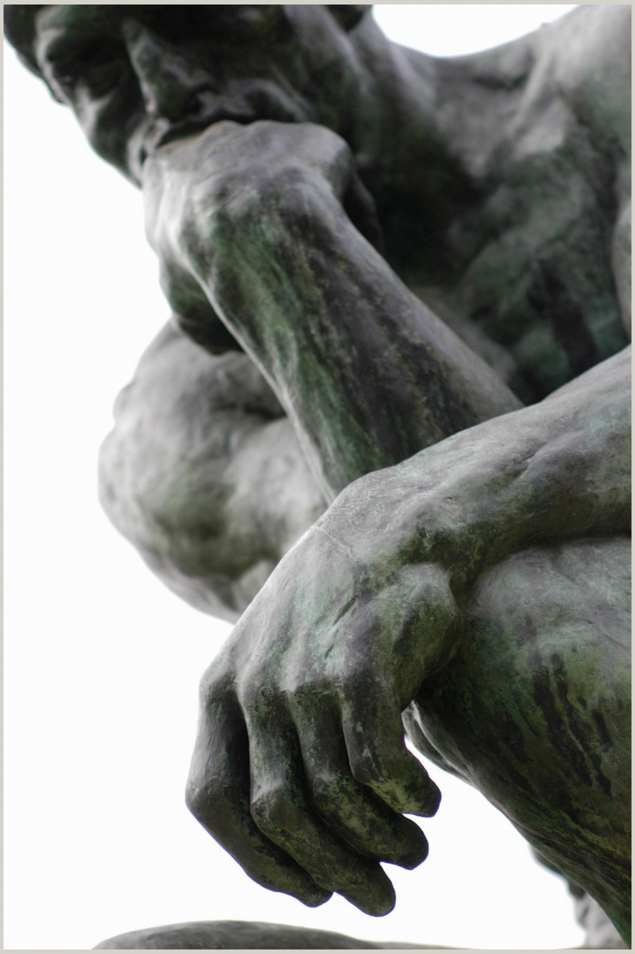
While it is easy to be swept away by the cool things we do, we should not forget that the field also contributes to fundamental scientific knowledge. So let's take a step back from the frenzy and think about the science computing pursues. To help, I pose five deep questions [3]:

P = NP?
What is computable?
What is intelligence?
What is information?[1]
(How) can we build complex systems simply?

There is nothing special about the number five; it is just a way to get a list going. I call them deep because they speak to the foundations of the field, reflecting the kind of far-reaching issues that drive day-to-day research and researchers toward understanding and expanding the frontiers of computing.

The question of whether P equals NP is undeniably the most well-known unsolved problem in the field. A proof in the positive (P = NP) would have profound practical consequences, shaking the founda-

[1]I thank Robert L. Constable, Dean of Faculty of Computing and Information Science at Cornell University, Ithaca, NY, for suggesting this question.



tions of cryptography upon which today's electronic transactions are based. It could give us efficient ways to solve intractable problems (such as the traveling salesman problem and the subgraph isomorphism problem) that arise in mathematics, as well as in every other science and engineering discipline. A proof in the negative ($P \neq NP$) would have profound theoretical consequences for computer science and mathematics, perhaps by discovering a brand-new proof technique.

In order to answer what is computable, we must consider the underlying machine (abstract or physical) that is the computer. Consider the Internet as a computer. What is a universal machine model for the

Is there a complexity theory for analyzing our real-world computing systems as there is for the algorithms we invent?

Internet? Consider a molecular computer, a DNA computer, a nano-scale computer, or even a quantum computer [1]. What problems can and cannot be solved through them? If contemplating these emerging substrates is not mind-bending enough, consider a human and a machine working together as a single computer to solve problems that neither would be able to solve alone [2]. Given that humans and machines have complementary computing capability, now ask: What is computable?

In the 1950s, the founders of artificial intelligence challenged computing researchers with the third question. As our understanding of human speech, vision, language, and motor skills has improved since then, and as we have achieved scientific and technological advances in computer science, neuroscience, cognitive science, and the behavioral sciences, the landscape has changed dramatically. Computer scientists can now probe both deeply and broadly in our quest to understand intelligence, from the neuron to the brain, from a person to a population.

"Information" in the field of computing has seemingly disparate meanings depending on scientific context: information theory, information processing, information retrieval, or information science. Distinguishing signal from noise is relevant in all these contexts, whether we mean transmitting bits over a wire, searching for an answer through the Web, or extracting knowledge from an ocean of sensor data. In essence, there is a chain of representations, from bits to data to information to knowledge. Beyond computing, nature has its own way of encoding information that is not as simplistic as using 0s and 1s. The genetic code is an obvious example. More sweepingly, by interpreting a DNA strand, a cell, or an organism as a reactive system (processing inputs from its environment and producing outputs that affect that environment), it is no longer metaphorical to say biology is an information science. Ditto geosciences. Meanwhile, with quantum computing, it's not bits but qubits.

Our engineering prowess creates computer, communication, and information systems that enhance everyone's daily lives and enable us to do astonishing things: instantaneous access to and sharing of information through palm-size devices with friends in social networks of tens of millions of users; dance with remote partners through 3D tele-immersion [4]; and lead alternative lives through avatars that can even defy the laws of physics in virtual worlds. The complexity of these systems delivers the richness of functionality we enjoy today, with time and space performance that spoil us. Their complexity, however, also makes it difficult for even the original system developers to analyze, model, or predict system behavior, let alone anticipate the emergent behavior of multiple interacting systems.

Can we build systems with simple and elegant designs that are easy to understand, modify, and evolve yet still provide the functionality we might take for granted today and dream of for tomorrow? Is there a complexity theory for analyzing our real-world computing systems as there is for the algorithms we invent? Such a theory would need to consider measures of not just time and space but of energy and cost, as well as dependability, security, and usability, most of which elude quantification today. More ambitiously (or crazily), is there a complexity theory that spans both the theory and practice of computing?

I pose these questions to stimulate deep thinking and further discussion. What deep questions about computing would you want answered? In 50 years, how different will they and their answers be from what we ask and are able to answer today? C

References

1. Reiffel, E. and Polak, W. An introduction to quantum computing for non-physicists. *ACM Computing Surveys 32,* 3 (Sept. 2000), 300–335.
2. von Ahn, L. Games with a purpose. *IEEE Computer* (June 2006), 96–98.
3. Wing, J. Thinking about computing. *Computing Research News 19,* 5 (Nov. 2007).
4. Yang, Z., Yu, B., Wu, W., Diankov, R., and Bajscy, R. Collaborative dancing in tele-immersive environment. In *Proceedings of ACM Multimedia* (Santa Barbara, CA, Oct. 23–27). ACM Press, New York, 2006, 723–726.

Jeannette M. Wing (jwing@nsf.gov) is Assistant Director of the Computer and Information Science and Engineering Directorate at the National Science Foundation, Arlington, VA, and the President's Professor of Computer Science in the Computer Science Department at Carnegie Mellon University, Pittsburgh, PA.