Herr kansler, Herr rektor, Prorektorer, värderade kolleger, mina damer och herrar:

Computer Science today appears as a highly successful discipline. Within about half a century since it has been recognized as a scientific discipline, it has helped transform our lives in profound ways. The most direct way one can witness the success of Computer Science is in the technology that it has enabled: for example, our computers today are far beyond being just calculating machines; we demand of our phones much more than just to be able to call somebody; our cars are assisting and anticipating our driving in sophisticated ways. At the same time, one can see the success of Computer Science through the impact it had on modern society: in the way we educate ourselves; in the way we communicate; in the way we keep contact with each other; in the way we entertain ourselves; in the way we get informed; in the way we organize ourselves politically, socially and professionally. We can see it in our modern culture in the way we create art; in the new types of literature or music; in the new types of exhibitions. We can see the success and the impact of Computer Science in the great advances in medicine: in new molecular screening tests; in the intelligent body implants; in the way surgeries are performed today; in all the advances towards personalized medicine. Despite its newcomer status among sciences, Computer Science has changed profoundly the nature of many disciplines; we have today many new fields of science, technology and humanities that are using “computational” as a prefix to name themselves. Here are just a few examples: computational biology, computational chemistry, computational physics, computational mathematics, but also computational linguistics, computational and algorithmic music, computational intelligence, computational philosophy, computational neuroscience, and computational economics.

In fact, many people argue that Computer Science and the technologies it has enabled have led to such a profound transformation in the way science is conducted nowadays that the computational approach is today the third pillar of science, on equal footing with its other two pillars: theory and experiment. In 2005, the U.S. Presidential Information Technology Advisory Committee issued a report titled "Computational Science: Ensuring America's Competitiveness," that was stating:
"Together with theory and experimentation, computational science now constitutes the “third pillar” of scientific inquiry, enabling researchers to build and test models of complex phenomena – such as multi-century climate shifts, multidimensional flight stresses on aircraft, and stellar explosions – that cannot be replicated in the laboratory, and to manage huge volumes of data rapidly and economically. Computational science’s models and visualizations – of, for example, the microbiological basis of disease or the dynamics of a hurricane – are generating fresh knowledge that crosses traditional disciplinary boundaries. In industry, computational science provides a competitive edge by transforming business and engineering practices."

On the other hand, computer scientist Moshe Vardi (Rice University) argues in a recent editorial in Communications of the Association for Computing Machinery (vol. 53, no. 9, p. 5, 2010) that the computational approach has profoundly changed the nature of both theory and experiment, the two traditional legs of science. The theory in systems biology for example can well take the form of Petri nets, statechart diagrams, timed automata, or pi-calculus programs, which were developed originally in the context of computer science. The nature of experimentation as a fundamental pillar of science also has transformed both through complementing physical experiments with computational ones, as well as through the scale of its computational challenges; for example, the Compact Muon Solenoid experiment at CERN’s Large Hadron Collider generates 40 terabytes of raw data per second, a flow that gives a great challenge even in terms of storing it, not to mention analyzing it.

While answering to all these challenges with advances in modeling, simulation, and other computational technologies, Computer Science became established as a fundamental discipline in itself. Seen from this point view, the European term “Informatics” (“Datavetenskap” in Swedish, “Tietojenkäsittelytiede” in Finnish) is much better than the American term “Computer Science” to describe the nature of this scientific discipline. Whereas the latter term refers to a particular device of computing, the term “Informatics” defines my discipline more accurately as the general science of information and information processing. As a matter of fact, Computer Science has been most successful as a fundamental discipline in enabling reasoning about computing, independently of its physical implementation, in other words independently of what kind of processors, memory, or communication protocols one uses to implement computing. In its quest to understand the nature of the various ways of processing information and doing computations, Computer Science has developed its own scientific language and concepts, and it has identified a number of scientific challenges stemming from within itself. Some of these challenges have received great recognition and visibility in the scientific community independently of their implications in technologies or impact in other sciences, but rather for their fundamental importance for Computer Science itself. One such challenge is the question whether the computational complexity classes of P and NP are equal; the problem has been listed as one of the 7 Millennium Prize Problems by the Clay
Mathematics Institute, each of them carrying an award of 1 million dollars for solving it. To put this list of problems into perspective, one should note that one of the problems, the Riemann hypothesis, formulated in 1859, also appeared in the list of problems formulated as great challenges by mathematician David Hilbert on August 9, 1900. The P vs NP problem is much younger, having been proposed in 1971; it can be formulated as whether there exist questions whose answer could be quickly checked for correctness, but which require an impossibly long time to solve by any direct procedure. To claim the 1 million dollar award one only has to choose his favorite difficult problem, show that a candidate solution can be checked efficiently, but a solution cannot be searched for efficiently. That is to say, the challenge is not to show that the prize applicant could not find a solution, but rather that one cannot be found efficiently, independently of any current or future computing technology. In other words, this is a problem about the very nature of information processing, a true scientific jewel of Computer Science.

Computer Science has enabled a fundamentally new way of doing science and understanding the world in terms of computations. One of the areas where this approach has been very fruitful is in biology. Aviv Regev (MIT) and Ehud Shapiro (Weizman Institute of Science) have described the paradigm “Cells as computation” (Nature 419 (343), 2002) as an approach to understand biological and biochemical mechanisms as systems of interacting computational entities. The same concept was proposed in our 2002 book “Computational processes in living cell. Gene assembly in ciliates”, co-authored with Andrzej Ehrenfeucht (Boulder, Colorado), Tero Harju (Turku), David M. Prescott (Boulder, Colorado) and Grzegorz Rozenberg (Leiden, the Netherlands). At the core of this approach is the idea that biology can be described through the logical interactions of various types of molecular entities such as genes, RNA molecules, proteins, and others. When looking at biology, what a computer scientist typically does is to abstract from the biological details of the mechanism at hand and replace them with formal statements about the type of information they carry, about the types of interactions they are capable of, the types of resources they need in their interactions, etc. What the computer scientist does in fact is to replace the biological system of interest with a computational system described in a high-level language very similar to a computer programming language. On that level, the computer scientist can perform formal reasoning about the properties of the computational system and relate them to the properties of the biological system in the background of his investigations; he can do static and dynamic analysis, numerical simulations, approximations, predictions, sensitivity analysis, estimates of robustness, etc. This type of research, called computational systems biology, is a beautiful example of genuinely multi-disciplinary research that is already giving new insights into the inner mechanisms of complex biological systems.

But the concept of “life as computation” can in fact be taken one step further. Starting from the vision of computational systems biology of huge numbers of molecular machines interacting according to precise logical rules, one may turn the tables and ask whether we can write ourselves molecular programs that, when executed inside a cell-like environment give rise to interesting nanoscale patterns and behavior. The goal here is to take the basic primitives of biochemical interactions and use them to design new types of algorithms that run in suitable biochemical solutions, leading
to new types of computing. This way of computing is drastically different than that based on electronic computers. One starts by specifying the product requirements in a high-level language and then compiles the specifications to obtain a molecular program, for example in terms of a number of DNA strands; the precise sequence of nucleotides of those strands is as crucial for the molecular program as the precise sequence of 0/1 is for the binary code of a computer program. One orders then those strands from a biotech company or from a university core facility and obtains a small test tube with them floating in a solution. Taking good care of them, for example adding some salted solution and cooling them slowly from almost boiling point will yield something almost magical: the strands start to self-assemble into a nano-scale object that follows the specifications that the computer scientist made in the beginning of the project. The end result could be a nano-scale electronic circuit, a molecular motor, or even a small computer capable to permeate cellular membranes and make logical decisions depending on the environment. In other words, non-electronic, unconventional computing.

The world seen through the eyes of a computer scientist is a kingdom dominated by concepts such as logic, abstraction, transitions, transformations, interactions, concurrency, competition, synchronization, robustness, efficiency, etc. When looking at other fields of science through the glasses of Computer Science, those fields may sometimes appear drastically different than the conventional way. Representatives of other sciences may have sometimes difficulties identifying themselves with the image of their field projected by Computer Science. However, they may also play the game of abstraction and offer, half-jokingly, the following unusual definition of computer scientists, paraphrasing Goethe:

"Computer Scientists are like Frenchmen: whatever you say to them they translate into their own language, and worthwith it is something entirely different."

To conclude, I will tell you that Computer Science for me is about computing in all of its forms (electronic, molecular, quantum, or other forms); it is about designing and building efficient computing, and applying it to any application domain in need of computational solutions; it is about understanding the limits of computing and of efficient computing; and it is about a general scientific approach in terms of information and information processing.

Turku, September 30, 2011