CS480	Computational Theory

Computer Musings

Reading for the Final — *By Robert B. Laughlin*

A Nobel-Prize-winning physicist, from his book, *A Different Universe*:

Driving to work one morning I heard a fascinating allegation on the radio that women understand computers better than men do. The speaker did this only indirectly and was careful to be politically correct, but the point of her remark was nonetheless clear. After she explained her position I could see that she was probably right. Men always want to tinker with the computer, she said, take it apart, add memory and peripherals, and so forth, while the women concentrate on more important things like sending out hundreds of email invitations to a wedding shower. This is fully consistent with my own experiences with technological things. When our car breaks down I obsess on figuring out what happened while my wife just wants to spend a lot of money to get it fixed and go to the movies. Women just seem to intuitively understand better than men that how a thing works matters much less than what one uses it for.

Computers are an especially helpful instance of this technological fact of life because they are so transparently hierarchical. At the highest level they are tools that store and process email, manipulate more formal written communications, and allow one to search for deals at Internet auctions. (There are less practical uses, such as tasteless video games, secret pornography downloads, and trading copyrighted songs and movies, but these waste time and do not count.) At the next level down one has the processor, motherboard, and expansion slots containing wonderful things with names like voodoo and rage, so powerful that they require extra fans. Below this one finds silicon microchips with their fabulous webs of wires and diffused transistors, and below this the orderly lattice of silicon atoms through which electrons and holes propagate. It is possible to send out all those shower invitations without thinking about it too carefully because of the reliability of an enormous tower of functionalities, each resting on the one below and supporting the one above. How each level works is immaterial. The invitations could

just as easily have been sent out by little gnomes with pads of paper and miniature telephones, although they would probably have demanded more money.

Computers are machines. Like any other machine, such as a lawn mower or steam engine, a computer works by moving matter from place to place. Because the matter in question is composed of electrons solely, it can be made to move easily and with blazing speed, but it is still conceptually the same as a piston rod or crankshaft in a car. In the end, the objective of computer engineering is still to get an assemblage of mechanical linkages to cause some physical thing to occur, such as deposition of ink on a page, motion of a loudspeaker cone, or twisting of the liquid crystal in a display pixel. Computers are often touted as the magical technology of the twenty-first century, but they are actually the crowning achievement of the nineteenth.

A key difference between computers and other kinds of machines is the ease with which their mechanical linkages can be modified. The modification process is called programming, and it has the genteel appearance of a term paper being typed, except that there is more coffee consumed and more swearing. But looks can be deceiving. This activity is not term paper composition at all but auto shop. It involves construction of complex mechanical relationships between simple parts that then either function or don't depending on the workmanship. One has simply traded in lathes and torque wrenches for pencils and keyboards.

There are several qualitative differences between computers and cars brought about by the ease of modification. For example, the economics of engineering is fundamentally altered by driving the cost of making the physical microchip way below the labor cost of programming. This is why software costs so much, and why its monopolization is so different from that of steel, railroads, or oil. Programming is also sufficiently similar to day-to-day use of the computer that the two become mixed up in people's minds as a kind of super abstraction of thinking. In the world of computers one begins to confuse play with work, work with play, and business activity with fundamental meaning. Computation, as most people experience it, is separated by complex layers of economic activity from the basics of the machines them-

selves and is in this sense a classic case of emergence. Modern computer programs are constructed by enormous teams of people, each of whom understands only a small fraction of the task, and these programs often wind up interacting with each other in ways their creators could not have imagined. This sociological phenomenon is a logical implication of the simple fact of cheap programmability made possible by the agency of electricity.

[. . .]

One of the more interesting trends of the computer age is that physical science students are increasingly unwilling or unable to write computer code. I was very upset when I first observed this and took stern measures in my department to counteract it, much to the students' chagrin, for I myself am very good at coding and consider it something any self-respecting technologist should know how to do. Eventually, however, I realized that the students were right and I was wrong, and stopped the crusade. Computer programming is one of those things in life, like fixing one's own car, that is fascinating, fun, useful—and unacceptably time-consuming. The truth is that it is no longer cost-effective for most well-educated people to program their own computers, or even to learn how to do so. The wise use of time is to spend a few bucks to buy a program that does what one wants or, in extreme cases, search the internet for free software.

When I was a graduate student, in the early 1970s, the economics were exactly reversed. Student labor was cheap and computers were hideously expensive mainframes that occupied entire floors of university computer centers. They were pampered affairs, with legions of attendants working in shifts around the clock and special air conditioning with power backups. We wrote computer programs for these behemoths late at night on grey metal machines about the size of bears. One of these machines would hum away with its motor running until one struck a key, at which point it would tremble a bit, go chunkh, and put a crisp new hole in the card one was punching. After one was done with a card, one would hit the feed key, and the machine would rotate the card klicka-ka-chunkha-chunkh to the bottom of the growing stack and feed in a new blank one. The computer programs we wrote were realized as decks of cards punched in this way, held together with rub-

ber bands and stored in cardboard boxes. Running the program involved submitting one's deck to an attendant, who would feed it into a card reader, an apparatus that looked like a gasoline-powered wood plane and sounded rather like a vacuum cleaner with a leaf caught in its fan. The printer would all the time be whining away in the background under its metallic sound hood and frantically throwing off page after enormous page of white computer paper as though it had gone berserk. Every few minutes an attendant would retrieve this output, which would require opening the sound hood briefly, thereby filling the room with unbearable screeching. The attendant would then tear the output at appropriate seams and stuff it into bins for pickup by students. This output consisted mostly of incomprehensible operating system mumbo-jumbo with the stuff one had actually calculated on the last page—unless the code had a mistake, in which case one would get either a thin nothing or an inch of meaningless core dump output, depending on the severity of the mistake. The expense of all this was incomprehensible. I remember talking with one of my fellow students just after he had submitted a deck in three boxes and seeing his hands shake. Ah, those were the days.

The most famous deck story of all time is the box containing an enormous hydrodynamic simulation code that somebody dropped, causing cards to fly everywhere. The program in question was promptly christened Nixon because it would clearly never run again. But happily it did run again and became the nucleus of the classified program Lasnex, the current workhorse of laser fusion simulation.

The joke about gender bias in computer skills thus has at its core the more important observation that we owe the existence, reliability, and utility of computers to principles of organization—including economic ones. That women have an easier time understanding the supremacy of organizations than men is not news, for this was known to the ancients and recorded in numerous places, notably the *I Ching*. According to Taoist philosophy, the universe is impelled forward by the conflict of two opposing principles, yin and yang, that constantly produce and supplant each other. Yang represents maleness, the sun, heat, light, dominance, and so forth. Yin represents fe-

maleness, the moon, material forms, cold, submission, and so forth. Yang, the sunny southern side of the mountain, creates, while yin, the shady northern side, completes the created thing. One might say that we are presently in an age of yin, and even though computers were brought into existence by yang, they have reached their full potential only under the dominance of yin. A more direct western way of saying this same thing is that computers were originally conceived as dogs but now have become cats. The machine one brings home from the store is clever, self-serving, constantly underfoot, and always scheming how to get you to do what it wants. But when you lobotomize the thing, strip away its sophistication, and reach down past the facade to the wires, transistors, and algorithms underneath you find unquestioning obedience, steadfast loyalty, straightforwardness, and simplicity—i.e., a dog.