Genesis Machines: Computing with the Code of Life

MARTYN AMOS

Edited transcript of talk given on 21 May 2008

Manchester Memoirs: The Memoirs and Proceedings of the Manchester Literary and Philosophical Society 146, p.p. 91-96 (2010).

In 1959, a great personal hero of mine, the Nobel Prize-winning physicist Richard Feynman, gave a ground-breaking speech[1] titled “There's Plenty of Room at the Bottom”. In his speech, Feynman outlined the possibility of individual molecules - even individual *atoms* - making up the component parts of computers in the future. With that one talk, Feynman laid the foundations of the field we now know as nanotechnology. At the time, computers filled entire rooms and were tended by large teams of lab-coated technicians, so the idea that one could compute with individual *molecules* was considered fairly outlandish.

However, around the same time the microbiologist A. J. Kluyver argued that “The most fundamental character of the living state is the occurrence in parts of the cell of a continuous and directed movement of electrons.” At their most basic level, computers work in exactly the same way; by funnelling electrons around silicon circuits, and the links between computers and living organisms have been well-documented and explored by von Neumann, Turing and others.

It wasn't until 1994, however, that the feasibility of *actually building* computers from molecular-scale, organic components was demonstrated. Feynman's vision had waited, not only for the technology to catch up, but for a person with the required breadth of understanding and insight. In 2002, Len Adleman, of the University of Southern California, won a share of the computer science equivalent of the Nobel Prize for his role in the development of the public-key encryption. Adleman has a long-standing interest in biology; when Fred Cohen, one of his students, showed him a program that could take over other programs and force them to replicate it, Adleman observed the similarities between the program's behaviour and that of a biological virus. Cohen subsequently made the first ever reference to a "computer virus" in an academic article.

In the early 1990s, Adleman read the classic textbook "The Molecular Biology of the Gene" co-authored by Jim Watson, who, together with Crick, discovered the structure of DNA. He came across the section describing a particular enzyme inside the cell that reads and copies DNA, and he was struck by its similarity with an abstract device in computer science known as the Turing Machine. By bringing together two seemingly disparate concepts, Adleman knew at once that, in his own words, “Geez, these things could compute.”

He founded a lab at USC and began building a molecular computer. He knew that DNA, the molecule of life that contains the instructions needed to build every organism on the planet - from a giant redwood to a blue whale - can be thought of, in abstract terms, as a series of characters from the set A, G, C and T (the letters representing the initial characters of the different chemical bases of DNA). Molecular biology has always been about the transformation of *information*, usually inside the living cell. This information is coded in the AGCT sequences of genes and in the proteins that these genes represent. Adleman immediately saw how this mechanism could be harnessed, not to represent proteins, but to store digital data, just like a computer encodes a file as a long sequence of zeroes and ones.

Adleman decided to use this fact to solve a small computational problem. Readers may be familiar with the Travelling Salesman Problem, and Adleman's was a variant of that; given a set of cities connected by flights, does there exist a sequence of flights that starts and ends at particular cities, and which visits every other city only once? This problem is easy to describe, but fiendishly difficult to solve for even a relatively small number of cities. This inherent difficulty is what made the problem interesting in Adleman's eyes, “interesting” being, to a mathematician, a synonym for “hard”.

Adleman decided to build his computer using the simplest possible algorithm; generate all possible answers (right or wrong), and then throw away the wrong ones. He would build a molecular haystack of answers, and then throw away huge swathes of hay encoding "bad" answers until he was left with the needle encoding the correct solution (of which there may be just a single copy). For Adleman, the key to his approach was that it is possible to *synthesise* DNA in the laboratory. A machine the size of a microwave oven will sit in a lab connected to four pots, each containing either A, G, C or T. Type in the sequence you require, and the machine gets to work, threading the letters together like molecular beads on a necklace, making trillions of copies of your desired sequence.

Adleman ordered DNA strands representing each city and each flight for his particular problem. Because DNA sticks together to form the famous double helix in a very well-defined way, he chose his sequences carefully, such that city and flight strands would glue together like Lego blocks to form long chains, each chain encoding a sequence of flights. Because of the sheer numbers involved, he was confident that a chain encoding the single correct answer would self-assemble. The problem then was to get it *out*. In a way, Adleman had built a molecular memory, containing a huge file of lines of text. What he then had to do was sort the file, removing lines that were too long or too short, that started or ended with the wrong words, or which contained duplication. He used various standard lab techniques to achieve this, and, after about a week of molecular cutting and sorting, he was left with the correct solution to his problem.

The example that he solved could be figured out in a minute by a bright 10-year-old using a pen and paper. But that wasn't the point. Adleman had realised, for the first time, Feynman's vision of computing using molecules. After he published his paper[2], there was a flood of interest in the new field of DNA computing, a tide on which I was personally carried. The potential benefits were huge, since we can fit a vast amount of data into a very small volume of DNA. If we consider that every cell with a nucleus in a human body contains a copy of that individual's genome - 3 *gigabytes* of data, corresponding to 200 copies of the Manhattan phone book – we begin to understand just how advanced nature is in terms of information compression.

After a few years, though, people began to wonder if molecular computing would ever be used for anything important. They were looking for the “killer application”, the thing that people are willing to pay serious money for, like the spreadsheet, that persuaded small businesses to buy their first ever computer. The fundamental issue with Adleman's approach is tied to the difficulty of the problem; as the number of cities grows only slightly, the amount of DNA required to store all possible sequences of flights grows much more quickly; a small increase in the number of cities quickly leads to a requirement for bathtubs full of DNA. Indeed, it was estimated that if Adleman's algorithm were to be applied to a map with 200 cities in it (rather than seven), the DNA memory required to store all possible routes would weigh *more than the Earth*.

It would appear, then, that DNA-based computing has reached the end of the line, *if* we insist on applying it to computational problems in a head-to-head battle against traditional silicon-based computers. When DNA computing first emerged as a discipline, I was dismayed to see a rash

of papers making claims that within a few years we would be cracking military codes using DNA computers and building artificial molecular memories vastly larger than the human brain. I was dismayed because I knew what had happened 30 years previously to the embryonic field of artificial intelligence. Again, hubristic claims were made for their discipline, ranging from personal robot butlers to automated international diplomacy. When the promised benefits failed to materialise, AI suffered a savage backlash in terms of credibility and funding, from which it is only just beginning to recover. I was very keen to avoid the same thing happening to molecular computing, but I, like many others, knew that we needed to look beyond simply using DNA as a tiny memory storage device.

The next key breakthrough was in realising that, far from being simply a very miniaturised storage medium that can be manipulated in a test tube, within its natural environment – the cell – DNA carries *meaning*. As the novelist Richard Powers observes in The Gold Bug Variations, "The punched tape running along the inner seam of the double helix is much more than a repository of enzyme stencils. It packs itself with regulators, suppressors, promoters, case-statements, if-thens." *Computational structures*, that is. DNA encodes a program that controls its own execution. DNA, and the cellular machinery that operates on it, pre-dates electronic computers by billions of years. By re-programming the code of life, we may finally be able to take full advantage of the wonderful opportunities offered by biological wetware.

As Oliver Morton[3] observes in his book *Eating the Sun*, "The world is not just a set of places. It is also a set of processes." This nicely illustrates the shift in thinking that has occurred in the last few years since the human genome has been sequenced. The notion of a human "blueprint" is outdated. A blueprint encodes specific locational information for the various components of whatever it is intended to represent, whether it be a car or a skyscraper. Nowhere in the human genome will you find a section that reads "place two ears, on on either side of head" or "note to self: must fix design for appendix." Instead, genes talk to one another, turning each other (and often themselves) on and off in a complex molecular dance. The genome is an electrician's worst nightmare, a tangle of wiring and switches, where turning down a dimmer switch in Hull can switch off the Manhattan underground system.

The human genome project (and the many other projects that are sequencing other organisms, from the orang-utan to the onion) is effectively generating a biological "parts catalogue"; a list of well-understood genes, whose behaviour we can predict in particular circumstances. This is the *reductionist* way of doing science; break things down, in a top-down fashion, into smaller and smaller parts, through a series of levels of description (for example, organism, molecule, atom). The epitome of this approach is the very well-funded physicists smashing together bits of nature in their accelerators in an attempt to discover what some call the God Particle.

Of course, smashing together two cats and seeing what flies off is only going to give us a very limited understanding of how cats work, so the reductionist approach is of limited use to biologists. *Systems biology* has emerged in recent years to address this, by integrating information from many different levels of complexity. By studying how different biological components interact, rather then just looking at their structure, as before, systems biologists try to understand biological systems from the bottom up.

An even more recent extension of systems biology is synthetic biology. When a chemist discovers a new compound, the first thing they do is break it down into bits, and the next thing they do is try to synthesise it. As Richard Feynman said just before his death, "What I cannot build I cannot understand." Synthetic biologists play, not with chemicals, but with the genetic components being placed daily in the catalogue. It's where top down meets bottom up – break things down into

their genetic parts, and then put them back together in new and interesting ways. By stripping down and rebuilding microbial machines, synthetic biologists hope to better understand their basic biology, as well as getting them to do weird and wonderful things. It's the ultimate scrapheap challenge.

As we all must recognise, the planet is facing a very real energy crisis. A major effort is now under way to engineer microbes that degrade plant waste to produce clean, cheap fuel. Other important challenges revolve around health - Jay Keasling, a colleague in California, has recently been awarded 43 million dollars by the Bill and Melinda Gates Foundation to persuade *E. coli* to make substances that are alien to them, but which provide the raw ingredients for antimalarial drugs. The drug is found naturally in the wormwood plant, but it is not cheap – providing it to 70 per cent of the malaria victims in Africa would cost $1 billion, and they can be repeatedly infected. Drug companies would need to cover an area the size of the entire state of Rhode Island in order to grow enough wormwood to satisfy global demand, so Keasling wants to produce it in vats, eventually at half the cost.

If we told someone in the field of nanotechnology that we had a man-made device that doesn't need batteries, can move around, talk to its friends and even make copies of itself – and all of this in a package the size of a bacterium – they would sell their grandmother for a glimpse. Of course, we already have such devices available to us, but we know them better as *microbes*. "Biology is the nanotechnology that works", according to MIT's Tom Knight. By modelling and building new genetic circuits, synthetic biologists are ushering in a new era of biological engineering, where microbial devices are built to solve very pressing human problems.

REFERENCES

1. Feynman, Richard P.: “There's plenty of room at the bottom.” *The Pleasure of Finding Things Out* (Allen Lane, 1999) 117-140.

2. Adleman, Leonard M.: Computing with DNA. *Scientific American*, (August 1998) 34-41.

3. Morton, Oliver : *Eating the Sun: How Plants Power the Planet* (Harper, 2008).

4. Amos, Martyn: *Genesis Machines: The New Science of Biocomputing* (Atlantic, 2006).

5. Endy, Drew: Foundations for engineering biology. *Nature*, **436** (2005) 449-453

6. Jones, Richard: *Soft Machines: Nanotechnology and Life* (Oxford University Press, 2004).

Dr Martyn Amos is in the Department of Computing and Mathematics, Manchester Metropolitan University

correspondence to M.Amos@mmu.ac.uk