

Such boundary regulations raise questions that are simultaneously philosophical, technical, and, in the end, bioethical: Is a long-term in vivo biosensor part of the patient's biophysiology? How does a patient's own DNA affect how that DNA samples itself in microarrays? Examples such as in vivo biosensors and DNA chips suggest that device design should proceed not exclusively from the engineering perspective of tools (because part of the tool is composed of living biological components), nor exclusively from the biomedical engineering perspective of functional replacement (because the purpose of the device is not biological but bidiagnostic). Rather, we might think about the status of bioMEMS devices from the perspective of integrated systems that may cross several boundaries: ICs fused with living cells that analyze biological samples that are then uploaded into a database. From a theoretical perspective, bioMEMS are illustrative of the need for biotech research to maintain an emphasis on the embodied, situated quality of biotechnical hybrid systems. Both from the perspective of research and design, and from the perspective of critical contextualizing of biotech, there is a need for a means of working beyond the irresolvable biology–technology boundary, a need materialized in the very workings of bioMEMS themselves.

## CHAPTER FOUR

### Biocomputing

#### *Is the Genome a Computer?*

#### The Life of a Computer

What if “life” turned out to be a form of computation? “Wang’s Carpets,” a short story by Greg Egan, explores the question of how the boundary between biology and computers may be negotiated in the future.<sup>1</sup> Revised as a section of the novel *Diaspora*, “Wang’s Carpets” replays a familiar trope in science fiction: the search for alien life, which is also a search for the criteria for alien life. Paolo Venetti, a sentient software “citizen,” comes across a unique type of life-form on the planet Orpheus, some twenty light-years from Earth. The life-forms appear to be immensely large, planar, kelplike “carpets” covering the planet. The description that Paolo’s computer gives of the life-form suggests that it is far from being living, let alone sentient:

The carpet was not a colony of single-celled creatures. Nor was it a multicellular organism. It was a single molecule, a two-dimensional polymer weighing twenty-five thousand tons. A giant sheet of folded polysaccharide, a complex mesh of interlinked pentose and hexose sugars hung with alkyl and amide side chains.<sup>2</sup>

Upon further analysis, Paolo discovers that the carpets—called Wang’s Carpets—display both computational and biological characteristics. Not only do they function like autocatalytic enzymes, ceaselessly generating and regulating themselves, but their planar structure also makes them a kind of Turing machine, able to perform simple calculations based on the combinations of the different tiles in the carpets. As Hermann Karpal, another character in the novel, notes to Paolo, “they [the Wang’s Carpets] can calculate anything at all—depending on the data they start with. Every daughter fragment is like a program being fed to a chemical computer. Growth executes the program.”<sup>3</sup>

However, the Wang’s Carpets do not simply carry out their “computations” arbitrarily. Paolo and Karpal discover that their view of the large, planar structures is but a

fraction of a polydimensional space encoded by the Wang's Carpets, in which organisms they refer to as "squids" interact and communicate with each other through physical contact. Eventually, Paolo and the others with him must confront a question both highly abstract and immediately political. Not only do the Wang's Carpets demonstrate both computational and complex biological characteristics, but the recursive way in which they do this raises the issue of sentience:

"All the creatures here gather information about each other by contact alone—which is actually quite a rich means of exchanging data, with so many dimensions. What you're seeing is communication by touch."

"Communication about what?"

"Just gossip, I expect. Social relationships."

Paolo stared at the writhing mass of tentacles.

"You think they're conscious?"

Karpal, point-like, grinned broadly. "They have a central control structure, with more connectivity than a citizen's brain, which correlates data gathered from the skin... Right or wrong, it certainly tries to know what the others are thinking about. And"—he pointed out another set of links, leading to another, less crude, miniature squid mind—"it thinks about its own thoughts as well. I'd call that consciousness, wouldn't you?"<sup>4</sup>

Aside from the difficulties of picturing polydimensional organisms encoded by kelp-like structures the size of continents, the very concept of the Wang's Carpets takes on a different tone in the ensuing chapters. Both Paolo and Karpal must decide how to present this discovery to the "polis," or governmental body, which will make decisions concerning whether or not to further explore and possibly colonize other planets. At the center of this issue is therefore the question of designating the Wang's Carpets as living or nonliving, as biological or computational, or as something both and neither.

The very idea of creating a computer out of biology seems at once an invention of science fiction and a curiously old idea. As Egan's short story illustrates in detail, the question of whether the "living" excludes the "computational" tends to become vague in the discourses of connectionist cognitive science, artificial intelligence, and related fields in which the brain is technically related to microelectronic computing machinery. However, histories of the modern computer point to the fact that, during the era of business industrialism, a "computer" was indeed a person, most often a person sitting in front of a mechanical tabulator, manually performing statistical calculations for the census or for a company's books of production and sales.<sup>5</sup> We can even extend this nominalism further and suggest that the very idea of technology, especially when aligned with labor, implies a reconceptualization of the living human body as a machine—we work to think/calculate and think/calculate to work. The notion of the "biocomputer" is therefore a case of something forever arriving in the future, as well as something that has always existed in principle. It elicits a range of heterogeneous images, from a new type of "bioport" technology to the image of outsourcing our brain-ROM for hire.<sup>6</sup>

However, within the discourses of discrete mathematics and computer science, "biocomputing" is becoming an increasingly specific set of both theoretical and practical procedures, with specific aims and questions raised within its cross-disciplinary context. One way of beginning to discuss biocomputing (or biological computing) is to differentiate it from a seemingly related field, that of computational biology or bioinformatics.<sup>7</sup> On the surface, there would seem to be little difference between biological computing and computational biology. There are two important differences, however. The first has to do with the chosen medium of each. In biological computing (or biocomputing), the chosen medium is biological, with particular emphasis on the properties of DNA and protein molecules and interaction processes. In computational biology (or bioinformatics), the medium is the computer, with particular emphasis on the use of computer technologies to simulate, model, and analyze data that is biological in its origin.

If that were the only difference, then both biological computing and computational biology would simply be two approaches to doing biology, akin to the difference between traditional "wet lab" techniques (e.g., cell culturing) and current "dry lab" approaches (e.g., computational modeling). However, a second difference can be added to the first, and that is the difference in conceptual aims. If computational biology (bioinformatics) has as its aim the use of computers to extend biology and biomedical research, biological computing (biocomputing) has as its aim the use of biology to extend computer science research. In short, we can say that computational biology/bioinformatics makes a biological use of computers, while biological computing/biocomputing makes a computational use of biology. For the sake of clarity, we will from here on out refer to biological computing as "biocomputing," in order to avoid terminological confusion. We will refer to biocomputing as the use of biological components and processes toward nonbiological, computational ends.

From this basic distinction (a distinction of disciplines as well), we already see two central characteristics of biocomputing: its overall conceptual aim of developing novel computational technologies, and its means of doing so through the use of biological components and processes. It will be helpful to contextualize these characteristics by reference to two rather well-known developments in computer science that have served as the main motivations of biocomputing. These are computer storage capacity and parallel processing. The former refers to the limits often placed on current silicon-based computers in terms of their storage capacity. The increasing miniaturization of computer storage technologies has led some to speculate on the possibilities of building nano-scale storage devices. However, it has been known for some time that one of greatest data storage devices exists in each of our cells—the intricately compacted, densely coiled, and elaborately indexed molecule of DNA. From a computer science perspective, the ability of DNA to act as an ultraspecific, dynamic database, as well as its large storage capacity, makes it an ideal model for studying computer storage in biological media. This interest is bolstered by another development, which is in computer

processing technologies. The ever-present shadow of Moore's Law, which refers to the progressive shrinking—and limit—of integrated-circuit (IC) technology, has prompted many to speculate as to whether other media might serve as inheritors of the microelectronics in current computers.<sup>8</sup> Researchers in biocomputing have proposed that DNA's base-pair complementarity (A-T; C-G) offers not one but two binary pairs, which could be used together for massively parallel processing applications.<sup>9</sup> In fact, the first proof-of-concept experiments in biocomputing were applied to types of computational problems that have, in the past, provide difficult, if not impossible, for traditional silicon-based computers.<sup>10</sup>

Therefore, the impetus for biocomputing emerges not from biology or biotechnology, but rather from the computer industry. The basic goal of biocomputing would then be to make use of biological components and processes (e.g., DNA base pair binding, protein folding, cell signaling) in an explicitly "nonbiological" manner. What would be the applications of such biocomputers? For one, the majority of researchers and spokespeople for biocomputing are clear in that biocomputers will most likely not replace our familiar desktop computers. Most foresee the possibility of either hybrid systems (with two types of processors, a silicon one for linear computations, a biological one for parallel computations) or fully biological computing systems geared toward highly specific problem application. The candidates tested thus far include cryptography and mathematical problems for which an exponentially large search field exists (we will turn to these examples later).<sup>11</sup> As may be expected, both the military and the IT industries have expressed a limited, cautious interest in biocomputing, though mostly at the level of R&D.<sup>12</sup> The majority of researchers working in the field are thus academic researchers, predominantly with backgrounds in either computer science or discrete mathematics. Rare are the biocomputing initiatives with biologists or within biology or biomedical departments.

Again, the notion of biocomputing seems vague—exactly how does one make a computer out of biology? Before addressing the details of biocomputing, it is important to assert that biocomputing is first and foremost an approach to the idea of "computability." In this, biocomputing is as much a conceptual practice as it is a technical one; or rather, we will want to stress the intertwined nature of the conceptual and the pragmatic in biocomputing research. In this way, we can continue by pointing out one of the central principles of biocomputing: that there is an equivalency between the notion of the "biological" and the "computational" that is based on a dissociation of medium and process. In other words, the motto of biocomputing may be paraphrased as follows: the genome is a computer. By this we mean that, for biocomputing, there are characteristics of computing that are seen to inhere in biological components (such as DNA) and processes (such as protein binding to cell membranes). This raises one of the fundamental—and still debated—issues in the very idea of biocomputing: how "computability" is defined, and, by implication, how "biological" is defined; for,

as we will see, biocomputing puts forth a set of material and pragmatic claims concerning the ways in which the biological and computational domains can be reconfigured and redesigned.

Biocomputing, as of this writing, is generally divided along three lines, according to the type of biological components and processes utilized.<sup>13</sup> The first biocomputing experiments were performed using simple DNA fragments and their base pair binding characteristics. This area of DNA computing is, as we will see, based on the parallel processing capacities of DNA's double binary set. Researchers encode elements of a problem to be solved into the DNA (for instance, a problem concerning the best route between multiple destination points), and, using standard molecular biology laboratory techniques, allow the DNA fragments to selectively bind to each other. From this biologically enabled combinatorics, DNA base pair binding "selects" and solves the problem.<sup>14</sup>

A second area, alongside DNA computing, is that of membrane computing. While DNA computing is based on the linearized matching of strings (DNA strands), membrane computing makes use of the ultraprecise molecular "fit" between specific protein transit molecules and protein membrane molecules. This lock-and-key structure ensures the passage of molecules into the cell (such as nutrient molecules) and prepares the delivery of molecules out of the cell (such as enzymes produced inside the cell). The cell membrane therefore has a kind of cascading mechanics to it, in which molecular fit between receptor regions on a protein and a membrane initiate a series of chain-reaction events that result in the membrane's changing shape and the entry or exit of the triggering molecule.<sup>15</sup>

A third and more recent area is that of cellular computing, a broadly defined area that includes processes such as cell signaling (akin to the process in membrane computing), protein-protein interactions (folding and binding between protein structures), and cell metabolism (the particular "pathways" of a given chemical reaction). As the most theoretically oriented of the biocomputing fields, cellular computing is unique in that it places emphasis less on components (DNA, proteins, membranes, cells), and more on the inherent network properties of cellular processes. If enough is known about a given biochemical reaction in the cell, then, theoretically, that reaction pathway can be used as a kind of distributed processing network within (and between) cells<sup>16</sup>—if, however, enough is known about such complex reactions, for this "systems" approach to biochemistry is in many ways an area that is just being defined within molecular biology itself.

To get an idea of biocomputing in action, we can consider one of its founding, proof-of-concept experiments as our starting point: Leonard Adleman's 1994 experiment involving the use of DNA to solve a standard directed Hamiltonian path problem. In doing this, we will pay particular attention to the techniques and technologies employed in the articulation of this DNA computer. This attention to detail will be a way



for us to highlight the implicit claims that are materialized in biocomputing practices. From there we can then go on to consider some of the more theoretical implications of biocomputing, especially in relation to the issue of "computability" and the notion of distributed networks.

### Biological Solutions, Computational Problems

Leonard Adleman's 1994 paper "Molecular Computation of Solutions to Combinatorial Problems" is widely regarded as a seminal proof-of-concept paper on the possibility of computing with biomolecules. Although speculation as to the computational aspects of DNA and other biological components was not uncommon prior to Adleman's experiment, it was this paper that translated or "ported" the informatic models of the biomolecular body into computational terms.<sup>17</sup> Adleman's DNA computer is far from a fanciful mathematical exercise; its concerns are simultaneously biological, computational, and mathematical. Indeed, its relevance, from a philosophical perspective, is that it renders computational concerns inseparable from biological concerns.

In his paper, Adleman shows how a particularly difficult type of mathematical problem can be solved using DNA and the standard tools of molecular biology. Consider a transportation problem such as the "traveling-salesman problem." You are a salesman, and must pass through seven cities on your itinerary. You do not want to waste money on plane tickets going back and forth to cities you have already visited, so you want to find the most efficient means of hitting each of the seven cities. In addition, you are beginning at the first city ( $O_1$  or  $O_{start}$ ), and must end up at the seventh city ( $O_7$  or  $O_{end}$ ). Your problem is thus: what is the most direct, efficient path that begins at the first city, ends at the seventh city, and passes through each city only once? (Figure 11).

Such problems are often found in the field of mathematics known as "graph theory." Graph theory, as a branch of geometry, is primarily concerned with the quantitative properties of networks. A "graph" ( $G$ ) is therefore a set of "nodes" or "vertices" ( $V$ ), some of which may be connected by "links" or "edges" ( $E$ ). Nodes can be people, cities, or viruses, and edges can be social interaction, transportation, or infection. A given set of nodes can have a radically different structure (or network topology), depending on how the edges are configured, just as the linking capability of edges is directly related to the configuration of nodes.<sup>18</sup>

The traveling-salesman problem is easily solvable on the average personal computer, but with one important requirement: that the input data is very small. A network such as the one described here, with a mere seven cities, is even solvable by visual inspection and pencil and paper. However, as the input data size increases, the possible solutions to the problem also increase. If we take a simplified version of three cities (nodes), there are six possible routes from any given node:  $3 \times 2 \times 1 = 6$  (or, in mathematical terms,  $3!$ ). What about ten cities? The mere addition of seven cities to the network would seem to matter little. But the possible number of routes increases

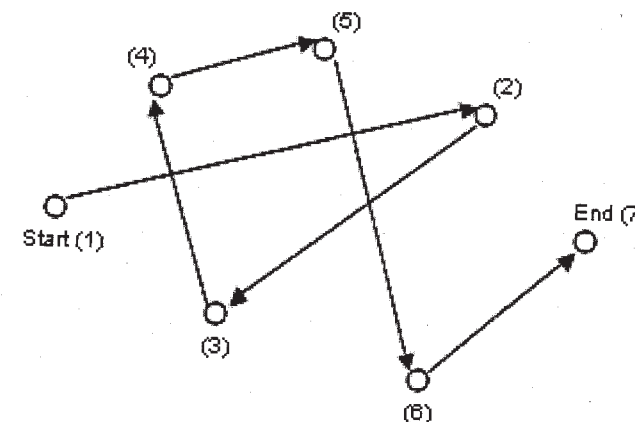


Figure 11. Diagram of solution to directed Hamiltonian path problem. Nodes represent cities, lines represent paths (adapted from Adleman).

to 3,628,800 (or  $10!$ ). The math is the same, but, as the input data size increases linearly, the search field for a solution increases exponentially.

This class of problems, known as "NP-complete," is characterized by its exponentially large search field. This means that, in some cases, there is no way to know if a solution to a problem exists, other than simply trying every possible solution. Note that the issue with such NP-complete network optimization problems is not finding a solution, it is merely deciding whether or not a solution exists at all. The key threshold in NP-complete problems, such as the traveling-salesman problem, is therefore in the efficiency of a given algorithm for finding a solution. In computational terms, this means considering whether a computer can find a solution in a reasonable amount of time. As the search field increases exponentially, so does the time required to find a solution.

The problem, though it deals with transportation (cities connected by highways or flight paths), could just as easily be considered in terms of communications (data routed through computers) or even epidemiology (infection and spread); that is, the traveling-salesman problem is an example of a network optimization problem, in which, for a given network, a certain value must be minimized. The network exists less as a spatial configuration, and more as a time-based actualization of a path through the network. In graph theory terms, the traveling-salesman problem is a version of a "directed Hamiltonian path" problem. It is "directed" because the edges of the graph are unidirectional (no backtracking; having one-way streets). It is "Hamiltonian" because it is named after the Irish mathematician William Hamilton, who devised a mathematical game involving tracing single paths along complex geometrical shapes. And it is a "path" because it does not repeat any node (in contrast to a "walk," which

may repeat nodes, and a "trail," which does not repeat any edge).<sup>19</sup> The directed Hamiltonian path specifies a very particular way in which a network is realized, by actualizing some edges/links, while leaving others in an unactualized state.

The important thing to relate concerning graphs/networks and graph theory is that the node/edge distinction often follows a space/time distinction; that is, nodes are often taken as being relatively static (e.g., cities on a transportation route), whereas edges are often taken as being dynamic (e.g., moving from one city to another via plane). We will return to this observation later, but for the time being it serves to denote a particular type of ontology concerning physical systems (be they computer networks or biological networks).

The problem Adleman sets out to solve using DNA is thus a directed Hamiltonian path, or a specified version of the traveling-salesman problem. Because Adleman chose a small network (seven nodes), the solution is easily verifiable. However, Adleman's experiment is not intended to be mathematical research, but rather to demonstrate the feasibility of computation at the biomolecular level. Adleman's basic algorithm for solving the traveling-salesman problem follows four basic steps: (1) randomly generate a series of paths through the network; (2) keep only those paths that have the correct starting and ending points ( $V_{in}$  and  $V_{out}$ ); (3) keep only those paths with the correct number of nodes (seven nodes or cities); (4) keep only those paths that hit each node once. As can be seen, the algorithm begins by generating a large group of possible solutions, then proceeds by a series of filtering processes, by first identifying starting/ending points, then the number of nodes in the path, and finally the identification of nodes in the solution paths.

The key biomolecular process that enables Adleman's experiment is the basic binding characteristics of DNA. As is well known, DNA, or deoxyribonucleic acid, is a double-helical structure, whose strands are composed of a sugar, a phosphate, and one of four nitrogenous bases (adenine, cytosine, thymine, guanine). Although DNA is as structural as a molecule as it is informatic, it has long been a tradition to "read" DNA as a linear sequence, thus making it amenable to treatment as a data string (where the beads composing the string are the variations of the bases). The way in which these variations of molecules bind to each other is widely acknowledged to follow regular rules. Known as "Watson-Crick base pair complementarity," it simply states that the binding affinities of DNA follow a regular pattern, in which adenine (A) always binds to thymine (T), and cytosine (C) always binds to guanine (G).

Under certain conditions of heating and cooling, the double helix of DNA can be broken apart ("denaturing"), synthesized ("replication"), and glued together ("annealing"), a process that has in fact been automated in the technology of PCR (polymerase chain reaction). From this basic principle of base pair complementarity, DNA contains two elements crucial to any computer: a processing unit (the enzymes that denature, replicate, and anneal DNA), and a storage unit (the regulatory "instructions" encoded in DNA strings). Not only does DNA form a highly efficient storage system

(an estimated one bit per cubic nanometer), but, in the living cell, instructions are carried out in a massively parallel fashion (in contrast to the sequential processing of instructions in many computers). This combination of massive parallelism and storage capacity makes DNA an ideal "computer" for instructions that require simple calculations on a massive scale—problems such as the traveling-salesman or directed Hamiltonian path problem.

To illustrate the way in which Adleman's experiment layers computational and biological concerns, it will be helpful to describe the way in which the individual steps of the algorithm were carried out.

The first step was to generate random paths through the network. Because the Adleman experiment works with DNA as its "hardware," the first step involved encoding each of the seven nodes (cities) into DNA. Adleman chose to encode each node as a short segment of DNA (20 base pairs, or 20bp). Each node can therefore be represented by  $O_n$ , where  $n$  is between one and seven. Then each edge or link was encoded into DNA as well. However, because the "edge" DNA would connect two "node" DNA, the edge DNA would therefore be half of each node DNA. For a node  $O_3$  and a node  $O_4$ , the edge connecting them would be half of each, or  $O_{3-4}$  (Figure 12).

Taking advantage of DNA's base pair complementarity, Adleman could then produce equal parts of single-stranded nodes (actually, the complementary strand of the nodes) and single-stranded edges. These "sticky" single strands of DNA could then be mixed together and allowed to bind accordingly. This mixture produces, through the combinations of possible binding between DNA strands, a large pool of DNA strands that represent possible solutions to the traveling-salesman problem (Figure 13).

The second step is to begin filtering this data set by keeping only those paths that begin and end with  $O_1$  and  $O_7$ , respectively. This can be achieved using PCR, a kind of DNA Xerox machine, which repeatedly heats and cools DNA while adding extra DNA fragments, or "primers," to synthesize copies of DNA in the process.<sup>20</sup> In this case, the selection of primers is key. Primers that contain  $O_1$  and  $O_7$  will therefore replicate only those paths that either begin or end with  $O_1$  or  $O_7$  (note: this does not yet concern the length of the paths, or their order, only their beginning and ending sequences).

The third step in the algorithm is to filter again the paths retained from step 2. In this filtering step, only those paths that contain the exact number of nodes in the path are kept. Because each node was encoded with a DNA strand of 20bp (twenty base pairs long), any possible solution to the traveling-salesman problem will be exactly 140bp (20bp  $\times$  7 cities). Such a filtering process can be carried out using a technique known as gel electrophoresis, commonly used to sequence DNA.<sup>21</sup> In short, DNA strands to be analyzed are passed through an agarose gel using electric polarization (because DNA has a negative charge, it will be pulled by the application of a positive charge). The gel forms a thick mesh through which short strands of DNA will pass faster than longer strands. Using this technique, researchers can measure the length of individual DNA strands, using DNA of known lengths as "controls."

| Node             | Code                  |
|------------------|-----------------------|
| O <sub>2</sub>   | TATCGGATCG GTATATCCGA |
| O <sub>3</sub>   | GCTATTCGAG CTTAAAGCTA |
| O <sub>4</sub>   | GGCTAGGTAC CAGCATGCTT |
| O <sub>3</sub>   | CGATAAGCTC GAATTTCGAT |
| Edge             | Code                  |
| O <sub>2-3</sub> | GTATATCCGA GCTATTCGAG |
| O <sub>3-4</sub> | CTTAAAGCTA GGCTAGGTAC |

Figure 12. Each node (city) and edge (path between cities) encoded as a DNA strand (adapted from Adleman).

$O_{2-3}$                        $O_{3-4}$   
 GTATATCCGA GCTATTCGAG CTTAAAGCTA GGCTAGGTAC  
                                  CGATAAGCTC GAATTTCGAT  
                                   $O_3$

Figure 13. Two "links" from a single node, showing overlapping binding by base-pair complementarity (adapted from Adleman).

The results from the gel electrophoresis will therefore keep those paths of exactly 140bp (seven nodes). From this, the fourth step provides the last filter, that is to keep those paths that hit each node only once. This is where the directed Hamiltonian path qualification comes in, requiring that the path is not only directed, but that it does not repeat itself. A version of PCR is here used ("graduated PCR") in which selected DNA fragments are attached to a magnetic bead, which "tags" the selected DNA fragment (e.g., the DNA strand for node O<sub>4</sub>). Using this technique, individual nodes along the path can be identified (e.g., with graduated PCR with O<sub>4</sub> as the primer, one would look for results that are 4 × 20bp or 80bp long). Repeating this process for each node will eventually filter out any paths that visit any single node more than once. If there are paths retained after this, they will constitute a solution to the traveling-salesman problem, because they will be directed (starting at O<sub>1</sub> and ending at O<sub>7</sub>) and will visit each node only once (will constitute a "path").

Working backwards, we can see how each step of the Adleman experiment combines computational and biological elements in a common way. At each step, a common technique in molecular lab biology is used to carry out a "calculation": DNA synthesis performs encoding procedures for the input data set, gel electrophoresis performs data analysis based on the physical and electrical properties of DNA, and PCR performs data analysis based on identifying nodes in a combinatorial fashion. Again, the key element that enables calculation to occur at all is the principle of base pair complementarity in DNA.

We should note the obvious: though Adleman's primary concern is computational, his materials are almost totally biological. One would at first be tempted to state that

this is a paradoxical notion of computer technology, because there is no technology present, only the biology of DNA and enzymatic reactions. However, it is just this initial presupposition that makes biocomputing worth noting as an instance of "biomedia." Certainly, there is no "technology" in this computer, if by this we mean electronic digital computers based on integrated-circuit technology. But in every other sense, Adleman's biocomputer is thoroughly technological, its "natural" hardware notwithstanding. It is a kind of technologization of biology, in which biology is technically recontextualized in specific nonbiological ways.

Thus, Adleman's DNA computer makes computing and biology inseparable, but in a certain way. It integrates the logic of the modern computer (input/output, memory, processor, logic blocks) with the structural properties of biological components and processes. This is more than a simple grafting of Boolean operators onto the physical medium of DNA, for, as Adleman and other biocomputing researchers note, DNA is a unique "computer" in its own right. Its parallel processing, dual binary logic, and immense storage capacity make for an entirely novel computing system.

### "So please just send in the machines"

As Adleman's experiment makes clear, the biocomputer is not simply a microelectronic black box; rather, the "object" of the biocomputer is a clustering of techniques from genetic engineering and molecular biology, discrete mathematics (particularly graph theory and combinatorics), theoretical computer science (particularly concerning the question of "computability"), and the still-vague area of molecular or nanocomputing. Biocomputing, in the Adleman model, is therefore as much biological as it is computational.

In this sense, several historical marking points are worth pointing out in relation to biocomputing. Using Adleman's experiment as our paradigm, we can outline a series of conditions that conceptually enable the field of biocomputing.

There is, first, a series of statements that are taken either as assumptions or as statements actually implicitly made by biocomputing research. As a particular intersection between molecular biology and computer science, biocomputing begins from an assumption concerning the equivalency between genetic and computer "codes." This relationship itself emerges from the historical backdrop of postwar developments in both cybernetics/information theory and molecular biology. The discursive transactions between these fields in part informs Francis Crick's concern throughout the 1950s and 1960s with "the coding problem" in DNA.<sup>22</sup> The research of Crick and others into "cracking the genetic code" therefore supplies an interdisciplinary backdrop against which the questions of biocomputing are posed.

Yet, despite the basic tenet of this backdrop—that the DNA molecule in the living organism is organized and even functions like a "code"—biocomputing still needs another statement to make available the question of computation with respect to biological systems. For molecular biology, the trope of the genetic "code" is in itself enough



to generate research into later efforts in genome mapping and genetic engineering. For biocomputing, by contrast, a further statement is required, and that is supplied by the studies in molecular biology on genetic regulation. François Jacob and Jacques Monod's well-known work on "genetic regulatory mechanisms" in the cell—resulting in the identification of "repressor" and "promoter" proteins and genes that turn other genes off or on—establishes a view of the cell that in many ways moves beyond the trope of the genetic "code."<sup>23</sup> Jacob and Monod's work suggests, using their own words, that the genetic material functions as a kind of "genetic program," regulating and modulating the states of other genes, which in turn dictates which proteins are produced and which are not. The trope of the genetic program obviously suggests that the extension of concepts from cybernetics into molecular biology will reconfigure the living cell as a cybernetics system, complete with feedback loops, regulatory mechanisms, and even subroutines. The reason Jacob and Monod's work is relevant to biocomputing is that it provides an early instance in which the living, functioning cell is possibly regarded as more than just a biological system; it is also a program, a computer in its own right, with its own type of formal logic (which remained elusive to Jacob and Monod, despite their use of terms from computer science and cybernetics). The work of Jacob and Monod on genetic regulation is an important statement concerning the nontropic quality of the notion that DNA is a computer. Although this is not explicitly stated by Jacob or Monod, the issue of the computability of DNA is raised here in an early form.

These two statements—that DNA is information, a "code," and that DNA is a computer, a "program"—can be seen to have created a fork in the road for future developments in biotechnology generally. One path is a familiar one, grounded in molecular biology, medical biotechnology, and genetic engineering. It is the path that extends the tropic quality of molecular biology, and from that begins to inquire into the biological applications of computing, leading to contemporary fields such as bioinformatics and genomics. It broadly goes under the rubric "computational biology". The other path is less familiar, and more recent. Instead of extending the tropic qualities of molecular biology, it takes molecular biology quite literally, and approaches biomolecules, cells, and interactions as being computers in their own right, related to but different from silicon-based, microelectronic computers. This "biological computing" (or biocomputing for short) is less concerned with the biological, biomedical implications of the informatic tropes in molecular biology, and is much more concerned with the computational, mathematical implications in biological components and processes. It is important to note that in neither case is informatics simply taken as a "metaphor" for something else that is material. Rather, both computational biology (the biological strand) and biological computing (the computer strand) ceaselessly materialize the relationship between biology and technology, genetics and informatics. However, they do so in different ways, and that difference is dictated in part by their approach to the organism. From the perspective of computational biology (i.e., bioinformatics), the

organism is an information-processing system that constantly correlates sequence and structure (DNA and protein) in the ongoing functioning of the living cell. From the perspective of biological computing (i.e., Adleman's DNA computer), the organism is not the question; the question is to what degree isolated biological components and processes may form highly specialized, "wet" computers. The question that biocomputing puts to the organism is, "Does the question of 'computability' apply to the organism?"

One of the primary propositions that biocomputing makes is that biological components and processes contain within them a set of characteristics that lends them to problems of "computability." In short, the basic proposition of experiments such as Adleman's is that DNA is a computer. Again, it is worth restating the difference between this statement and the statements made by molecular biology (including genetics and biotech generally), that DNA functions in an informatic manner. In the latter statement (DNA operates informatically), the reference point is the notion of "information" derived from cybernetics and information theory. DNA is seen to operate biologically, but, in doing so, it is also seen to fulfill the requirements of technical systems (i.e., communications and control systems in engineering). By contrast, the former statement (DNA is a computer), made by biocomputing, takes as its reference point the notion of a computer derived from the theoretical computer science work of John von Neumann and Alan Turing. It does not take cybernetics/information theory as its standard, against which to analyze molecular biological interactions; rather, it adopts the theoretical approach of computer science, asking not whether DNA fulfills the definitions of information, but whether DNA might be a specific, unique type of computer in its own right. Certainly, there is a technical notion of information involved in this inquiry, but rather than grafting a series of technical requirements upon DNA, the biocomputing approach—arguably the approach of theoretical computer science—poses a set of challenges to what may possibly be a feasible, totally biological computer system.

Turing's work in theoretical computer science is especially interesting in this context, for his basic concerns lie at the heart of the contemporary inquiries into biocomputing. Turing's early work in the mathematical problems of computability begins from an interest in the paradoxes inherent in formal logic (some might say the paradoxes produced by formal logic). Spurred on by the work of Kurt Gödel (and later, David Hilbert) into the issue of mathematical validity, Turing's paper "On Computable Numbers," published in 1936, set out to articulate the mathematical limitations or domains of activity for calculation.<sup>24</sup> The challenges put forth by mathematicians such as Gödel were based on the internal contradictions that formal logic sometimes generated. For example, take the statement "This statement is false." If it is proved to be true, then the statement is no longer false (it leads to contradiction). If, on the other hand, it is proved to be false, then the statement is no longer true (in that it denies its own validity). Gödel suggested that such logical conundrums posed a challenge to

mathematics, for they showed that such inconsistent problems were in some way representative of mathematics itself. Even taken modestly, the implication of Gödel's work was that an axiomatic system such as mathematics in some senses cannot help but to make statements that are arbitrary (that cannot be proved or disproved). Although Gödel's work generated much controversy within the world of mathematics, critics often pointed out that, despite such inconsistencies, buildings and bridges still stood, implying that Gödel's work was of only theoretical interest, and that such inconsistencies were anomalous by-products of a self-consistent essence of mathematics.<sup>25</sup>

For many, it was Turing who showed that the work of Gödel was of both theoretical and practical interest. Turing posed the question not of mathematical validity or certitude, but rather of the possibility of identification (or nonidentification) of such arbitrary statements. Rather than pore over the aporias of mathematical certitude, Turing focused on what is often called "computability": how can the difference between mathematically certain and mathematically arbitrary statements be identified? Turing also framed his question in terms of reflexivity: could arbitrary statements be identified from within the system itself? If a set of basic rules for differentiating between mathematically certain and mathematically arbitrary statements could be identified, then, in theory, this algorithm could be carried out by a machine, so as to automate the filtering process for practical applications in engineering.

Turing devised a scenario involving a "universal Turing machine," a machine capable of functioning like any other machine, once the instructions for that machine are fed into it. This universal computer could therefore simulate any other computer, whether its task was to calculate tables, send data, or play chess. The challenge for this machine was now to ask what would happen when it was fed its own instructions—that is, when it was asked to simulate itself. Such a problem would not be computable for Turing, for the computer would end up in a kind of digital psychosis, endlessly turning in on itself with no verifiable output. In other words, the Turing computer would be performing the kind of logical paradox articulated by Gödel earlier. From Turing's thought experiment, "computability" came to be defined in relatively precise terms: the situation in which it was possible to formulate a set of rules to decide whether or not a statement was susceptible to proof using only the rules of that system itself.

What does Turing's work on computability have to contribute to biocomputing? For one, it raises the question Turing and others posed at the early stages of electronic digital computing: how can a physical system be designed that would fulfill the mathematical requirements for computability? That is, how can you build a computer machine that would consistently function in a mathematically valid manner? The experiments of biocomputing researchers such as Adleman have suggested that biological computer systems can only be verified by the kinds of proof-of-concept experiments that rely on a presumed solvability of a given problem (i.e., the limited, easily solvable scope of the seven-node Hamiltonian path problem).

If we shift our perspective for a moment from computer science to molecular biology, we can put Turing's question another way: if Turing's computability thesis does establish a foundation for the level of validity in a calculating or computing system, and, if such systems can be constructed from biological as well as electrical components, then it follows that biological systems—as computing systems—inherently contain a set of logical inconsistencies that may be termed "noncomputable." In other words, if we accept the feasibility of biological computers, then we are also impelled to ask whether biological systems display some level of paradox, arbitrariness, or "unsolvability." This is tantamount to asking, in molecular biology, whether biological systems can, in certain cases, continue to function in an anomalous manner. Note that this is not asking whether or not a mechanical computer or biological system can malfunction or "break down" (systems crash or death). Rather, it is asking whether computers or biological systems can take on anomalous modes of functioning that, in their own right, are perfectly consistent, but that, from the vantage point of the computer or biological system, are identifiable as anomalous.<sup>26</sup>

### DNA Dualisms

While the very idea of DNA computers raises the questions Turing posed concerning "computability," they also lead to a set of more intricate questions concerning the status of DNA computers as living organisms. The question that many designs in biocomputing elicit is this: if a computer can be constructed from biological components and processes, to what degree is this computer "living"?

To address this question, we can briefly consider the correlative within computer science: the notion of "intelligent machines" and its related discourses (AI [artificial intelligence] and branches of philosophy of mind).<sup>27</sup> As a way of comparing organisms and machines, as well as humans and computers, the discourse of intelligent machines attempts to discover the limits of computers, as well as the illusions of anthropocentrism, or the intractable uniqueness of the human being-as-organism. Although this tendency to compare the organism and machine is not new, it takes a specific form in the development of the modern digital computer during World War II and the Cold War era. In particular, the work in theoretical computer science of Alan Turing and John von Neumann provides us with two paradigmatic ways in which organisms and machines have been related in terms of the computer.

The first approach is illustrated by the theoretical and mathematical work of von Neumann, elaborated in a series of lectures given at Yale University in the mid-1950s. However, as an adviser to many of the military-sponsored mainframe computing projects (ENIAC and EDVAC), von Neumann's interest in the relationships between humans and computers dates back to the mid-1940s and the development of the "stored-program computer."<sup>28</sup> Von Neumann was an adviser to the building of the ENIAC, perhaps the most well known of the room-sized mainframe computers. In his collaboration with ENIAC researchers, von Neumann helped to develop the notion of



a computer that had inputs, outputs, a processor, and a central memory unit. This logical structure, or "von Neumann architecture," remains, with several modifications, in the computers of today, and the very language of digital computers—processor, memory, and bits—derives in large part from von Neumann's design. Rather than separating program and data, the computer's memory could treat both as data, sidestepping the laborious process of plugging and unplugging wires for each procedure.

The von Neumann architecture contributes two important elements to our consideration of the relation between humans and computers. The first is the design itself. In brief, the von Neumann architecture is made of five components, each with a particular function in the whole: input and output units, a control unit, an arithmetic unit, and a memory. The program instructions and data on which the program operates are stored in the memory, while the control unit interprets the instructions, and the arithmetic unit completes the actual calculations. The inputs allow program instructions and program data to be fed into the memory, while the outputs display the results of the control and arithmetic units (one important change in current digital computers is that the control and arithmetic units are often considered together as the processor). The second contribution relevant to human-computer discourse is von Neumann's suggestion that binary rather than decimal encoding schemes could be used to represent data. Whereas the convention at the time was to use ten "flip-flops" of a switch to represent a single digit, the use of binary encoding schemes would enable three digits to be represented, owing to the combinatory possibilities (ten flip-flops of the switch, each flip encoding a different combination of values).

Taken together, these two innovations in modern computer design not only have had a technical impact on future computer science thinking, but they have also, in part, set the terms in which a comparison between humans and computers could take place. The articulation of "computability" into distinct sectors of the computer has an effect on how we think about the "processing" of data into knowledge, and in this sense the von Neumann architecture is perhaps more revealing as a mirror of human cognition than of the operation of a computer. This segmenting of the computer is therefore not unrelated to a segmenting of the processes of cognition, a comparison made more explicit in the field of cybernetics. This, combined with the implementation of binary coding and the use of "bits," makes for a view of the computer not only as segmented, but as combinatoric. The use of binary coding schemes allows for a greater flexibility in the storage capacity of the computer, while also allowing the encoding of all data—including programs themselves—into the computer's memory. In a sense, then, the first aspect of the von Neumann architecture (the five components) serves a segmentary, canalizing function, a kind of bureaucratic division of labor among interrelated component parts (first input into memory, then instructions to the control unit, etc.). If this is the case, the second aspect of the von Neumann architecture (binary coding) works against the first, allowing for a diversification of how much and

what kinds of data can be stored (data in will be transformed into data out). While the first aspect "stratifies" or segments computer function, the second aspect "smooths" out the plane of what can be encoded (including the very description of the computer itself, which leads to Turing's notion of a "universal machine").

In addition, it is noteworthy that, owing largely to von Neumann's interest in neuroscience, terms such as *memory* replaced traditional terms such as *storage*. Von Neumann's lectures on the computer and the brain explicitly explore the technical significations of this use of biological language to name computational processes. This interest was specifically a technical, functional interest. Rarely does von Neumann speculate on the capacity for computers to obtain intelligence, sentience, or anything related to "mind." Rather, his interest lies in the low-level processes of how data is input, interpreted, operated upon (calculated), and output toward certain action. Von Neumann systematically provides technical descriptions of modern digital computers alongside findings in neuroscience and studies of the brain. His language pushes the possible resonance between computer and brain through the use of physiological terms:

The organization of large digital machines are [*sic*] more complex [than analog computers]. They are made up of "active" organs and of organs serving "memory" functions—I will include among the latter the "input" and "output" organs.<sup>29</sup>

In this analysis, von Neumann notes that while modern digital computers utilize the segmentation and binary coding of the von Neumann architecture, the brain contains a combination of both analog and digital components. Interestingly enough, von Neumann locates the primary example of this in the activity of neurons, as well as in the genetic basis of nerve cells. Neurons contracting a muscle, or genes directing the synthesis of proteins, are examples of hybrid analog-digital systems in the body: "processes which go through the nervous system may... change their character from digital to analog, and back to digital, etc., repeatedly."<sup>30</sup> While the firing of a neuron or expression of a gene may be viewed as digital, their outputs—muscle contraction or protein folding—may be understood as analog.

However, in spite of these differences, von Neumann's low-level, materialist approach tends more toward the correspondences between the computer and the brain. In a series of comparisons that were to prove influential for a certain branch of AI, von Neumann essentially views the brain—and neuroscience—through the lens of modern digital computers:

systems of nerve cells, which stimulate each other in various possible cyclical ways, also constitute memories... In our computing machine technology such memories are in frequent and significant use... In vacuum-tube machines the "flip-flops," i.e. pairs of vacuum tubes that are mutually gating and controlling each other, represent this type. Transistor technology, as well as practically every other form of high-speed electronic technology, permit and indeed call for the use of flip-flop like subassemblies, and these can be used as memory elements in the same way.<sup>31</sup>

In other words, it is highly relevant that von Neumann's lectures were titled "the computer and the brain" and not "the computer and the mind." Despite this, we can detect in von Neumann's analysis an ambient interest in mind as well as brain. The use of terms borrowed from biology, as well as the direct comparison of computer and brain (through the lens of computers), indicates that such a comparison is not without its at least implicit interest in the kind of "mind" that would arise from the brain—whether this brain is a neurobiological one, a vacuum-tube one, or a transistor one. Certainly, terms such as *memory* become quite contentious here, and von Neumann's analysis can be seen to consistently border the line between an emphasis on lower-level brain functions and higher-level manifestations of mind. The questions implicit in von Neumann's analysis are twofold: First, on what level can the computer and the brain be seen as functionally analogous, and how might such a correspondence affect the design of technical systems? Second, if brain is not unrelated to mind, what "higher-level" manifestations might arise from the lower-level functions of computers?

To unpack the implications of these questions, we can turn to our second example of thinking concerning the relation between humans and computers: the work of Alan Turing and the famous "Turing test." Between his work for the British government's National Physics Laboratory in the late 1940s, and his lead on the MADAM computer project at Manchester University in the 1950s, Turing's interest in the theory of computation turned increasingly to questions of cognition and communication.<sup>32</sup> Turing's famous 1950 paper "Computing Machinery and Intelligence" set the terms for a still-ongoing debate about the feasibility of intelligent machines, as well as articulating the theoretical questions for the field of artificial intelligence.<sup>33</sup> One of Turing's insights was that the question of free will and determination in computers is, in a sense, a moot question. What counts is not whether or not a computer really is intelligent (and whether or not we can deduce valid criteria for assessing this), but whether or not a computer behaves as if it were intelligent. The implication here is that our own criteria for intelligence in humans (and other species, for that matter) is largely dependent on the ways in which we act and interact with others, and less dependent on any universal set of measurable criteria.<sup>34</sup> It is this precondition of interaction that gives the impression that the computer-as-player is somehow "intelligent" or "alive."

To demonstrate this further, Turing hypothesizes an experiment in which the "intelligence" of a computer can be assessed, but assessed in a way that would bypass the unending philosophical arguments of free will, sentience, and consciousness. This experiment—the "Turing test"—takes as its starting point communication between individuals separated physically from each other in separate rooms or by dividing walls.<sup>35</sup> Suppose that person A in one room communicates through a keyboard and terminal with person B in another room. Their communication is mediated by the monitors (that is, language as a cultural practice is the mediator). Depending on the conversation, person A may make assumptions about person B, about gender, ethnic-

ity, age, personality type, and so on.<sup>36</sup> Now, Turing proposes, suppose we replace person B with a computer, programmed such that it is able to answer typed questions and, based on those questions, return responses and pose its own questions. As Turing states, "may not machines carry out something which ought to be described as thinking but which is very different from what a man does?"<sup>37</sup> Turing's point here is that if person A is in any way convinced or in any way assumes that there is a human being on the other end, and not a computer, then the computer will have effectively demonstrated a certain level of "intelligence." Why? Because—and this is Turing's most controversial point—this is how human beings assess "intelligence" in each other.

In the context of biocomputing, the Turing test is noteworthy, not because it presages anything in biocomputing, but precisely because it rules out the possibility of biocomputers. Note that Turing's question is not "is biology computational?" but rather "is intelligence computational?"<sup>38</sup> In other words, it might be more accurate to say that Turing relates organisms and machines in the specific terms of humans and computers. This is done on the level of a certain type of cognitive performance; humans and computers are compared via the notion of "mind." By contrast, von Neumann's interest in the functional correspondences between the computer and the brain expresses a different sort of question, which also forecloses the possibility of biocomputers. If Turing's question concerns "intelligence," von Neumann's question is, "Is memory-based cognition in the brain a stored-program computer?"

To simplify greatly, we can suggest that while Turing emphasizes intelligence through the performance of the Turing test, von Neumann emphasizes memory as data processing through the architecture of the stored-program computer. However, these terms need to be understood in very particular ways. For Turing, "intelligence" is not an *a priori* quality exclusive to human beings, but is rooted in communication, performance, and practical assessment of behavior. Likewise, for von Neumann, "memory" is not so much a mysterious set of impressions specific to human beings or other organisms, but, when seen functionally as a pattern arising from switches (flip-flops), it can be considered a mechanical property of biological or computational systems. At no point in their respective texts do Turing or von Neumann question the existence of intelligence or memory in human beings as we commonly use the terms. Although Turing, not without some irony, asks "why shouldn't I be considered a computer?" it is important to note that both Turing and von Neumann begin by analyzing cognition in the organism through the lens of computer science—for both, the data processing involved in mind, brain, and computers proceeds through a series of discrete, finite-state machines. This is evident in the Turing test (series of questions and answers) as well as in the von Neumann architecture (correlation of control unit and memory unit, of arithmetic unit and control unit). In the process, the terms themselves (intelligence, memory) become transformed, not only quantitatively, but also qualitatively. We might say that for Turing, intelligence is a "state" of mind, whereas for von Neumann, the brain is just the "memory" of the computer.<sup>39</sup>



The field of biocomputing offers a corrective to the assumptions in the intelligent machine discourse, but it also raises its own set of problematics not explored by either Turing or von Neumann. If Turing and von Neumann see the human-computer relationship in predominantly cognitive terms, gauging "the human" in terms of higher-level processes such as learning, memory retrieval, or communication, biocomputing sees the human-computer relationship in predominantly biomolecular terms, displacing any interest in the human with an interest in biomolecular process. Biocomputing inverts the intelligent-machine discourse's interest in cognition, and places higher priority on the seemingly secondary, lower-level processes of the organism at the biomolecular level. Biocomputing keeps the relationship of organism and machine at the level of organism and machine, and resists the analogous comparison of human and computer that both Turing and von Neumann carry out. The key to this difference is that for Turing and von Neumann, the differentiation between organism and machine takes place at the level of human cognition: intelligence/learning and memory/data processing are the limits of what computers can do.

By contrast, biocomputing suggests that the difference between organisms and machines is not anything human, but rather a difference between living and nonliving systems: cell metabolism, gene regulation, and cell membrane signaling are the limits of what computers can do. Again, we can detect not only a biologism but an anthropocentrism in Turing and von Neumann, in the sense that the human is the standard against which computer performance is judged. Biocomputing does not necessarily assume this; it looks rather to the complex, "parallel" processes in the living cell as the threshold of computability. For Turing and von Neumann, what is at stake is essentially mind, with the human as its most sophisticated manifestation (one that is nevertheless amenable to computation). For biocomputing, what is at stake is "life," by this meaning the ability of biomolecular systems to carry out exceedingly complex calculations "naturally." In a strange way, neither Turing nor von Neumann is really interested in computation, but rather the computational explanation of human-centered attributes such as intelligence, learning, or memory access. Biocomputing researchers, in contrast, are centrally concerned with computation, with the understanding that computation in the 1990s comes to take on more than it had in the 1950s. For biocomputing, computation becomes, in part, synonymous with complexity and parallelism. In this context, "life" is both nonhuman and "intelligent."

We can therefore see the changes in the way that the computer is related to the human as a shift from an emphasis on "mind" (or cognition) to an emphasis on "life" (or complex networks). The key link between this emphasis on mind versus life is the changing artifact of the computer itself. From a historical perspective, it is obvious that the computer shifts from a room-sized, military-funded "electronic brain" to a microelectronic, industry-marketed "personal" computer. Although the computer as an artifact plays many roles and takes on many meanings, the point to be made here is that, from the perspective of computer science, the modern digital computer of Turing

and von Neumann conceives of computation as a cognitive function, whereas in the PC era of biocomputing research, computation is seen as inherently nonconscious, distributed, and in parallel.<sup>40</sup>

### When Computers Were Human (Again)

Before concluding with a consideration of the role of networks in biocomputing, it will be helpful to provide a summary of our concerns thus far. As we have seen, the burgeoning field of biocomputing raises a number of issues pertaining to our discussion of the relationships between biology and computers. Looking at Leonard Adleman's initial DNA computing experiments, we saw how the algorithm for DNA computing of a computationally complex problem (such as NP-complete graph problems) made use of techniques and tools from molecular biology. In general, we can say that Adleman's DNA computing experiment shows us three things about the concept of the biocomputer. First, it provides a possible set of criteria for assessing what may count as "computational" for biocomputer systems. Second, it demonstrates the technical feasibility of designing biological systems that function in "nonbiological" ways. And third, the DNA computer materializes a network, both computationally and mathematically, as well as biochemically.

Thus, biocomputing not only rearticulates "technology," but in its procedures it also brings a unique perspective to the question of "computation." As a number of biocomputing researchers have suggested, Adleman's DNA computer is important because it poses the question that Turing posed in relation to mainframe computing: what qualifies as an adequate (solvable) problem for a computing machine? On the one hand, the historical-biological understanding of biological components (the cell, the genome, DNA) as functioning in an informatic way has been instrumental in creating the conditions for conceiving of DNA as a computer in itself. On the other hand, the creation of the DNA computer has also run counter to the common habit of aligning computers with either brain (in the von Neumann architecture) or "mind" (in the Turing test). We have suggested that one result of this rethinking of computational paradigms is the shift from "intelligence" to "life," or, to put it another way, from the neuronal to the metabolic model of network behavior.

This shift is noteworthy precisely because the assessment of computability in biocomputing takes place in the context of problems that are inherently networking problems. The graph-theoretical problem of the traveling salesman (directed Hamiltonian path) not only provides a test problem for DNA computing, but it also instantiates a network layering of biological and informatic networks. Thus, these first two characteristics of biocomputers—nonbiological instrumentalization and the question of computability—come together in a third characteristic, which has to do with the network dynamics of biocomputers.

As Adleman's research suggests, the application of DNA computers to NP-complete problems is a form of network computing. Not only does the DNA computer "solve"



the Hamiltonian path problem through the analysis of DNA sequences, but it does so through a biological process in which multiple DNA "nodes" become linked or not linked to each other via a process of base pair binding (or an "edge"). There are, therefore, two network layers that are formed in Adleman's biocomputing algorithm. The first network layer is a biological one. This is evident from the very concept of the DNA computer as proposed by Adleman: a basic property of DNA in the living organism is its specificity in the binding of its base pairs (A-T; C-G). This biological property forms the "processor" of the DNA computer, in that it is the component that actually does the work of computing. The precomputing procedures of encoding (sample preparation input), as well as the postcomputing procedures of decoding (DNA sequencing output), are both dependent on the central processing unit (CPU) of the biological property of Watson-Crick base pair complementarity.

The second network layer is a computational one, a network not present in the DNA itself, but selectively extracted from a technique of decoding the double-stranded DNA sequences. The difference between the computational and biological layers is clearer if we consider two contexts. For the molecular biologist, the sequence would be decoded into a linear string of As, Ts, Cs, and Gs, and then imported into a range of software tools and analyzed against biological databases for possible homologies or polymorphisms; that is, sequence is first and foremost what counts for the bioinformatician and biologist. By contrast, the research in biocomputing takes the DNA sequence not as a string of (biological) data, but as a linearized arrangement of a graph (or network) of distributed nodes and edges composed of DNA; that is, in contrast to the biologist, the computer scientist's goal is to extract a graph from a string, to redeploy the network from out of the sequence, to decode the distributed from the linear. The network of this computational layer is therefore derived from the biological functioning of the biocomputer, *though that biological functionality has no biological function*. It is worth pausing over this statement: the biocomputer displays biological functionality because it creates a context in which selected biological components and processes may occur. In the case of the Adleman experiment, these are DNA base pair binding properties (Watson-Crick complementarity). However, this functionality serves no biological function, directly or indirectly, and is therefore quite different from DNA in the living cell or in the molecular biology lab. The DNA computer does nothing to the DNA itself that might be seen as beneficial from a biological, organismic, or medical perspective. However, it still "works," and that is perhaps the central insight of biocomputing.

This division between biological and computational layers is, of course, not a decisive one; it is understood from the beginning that the use of DNA's properties are being recontextualized radically for, in this case, computer science (and not molecular biotechnology). Likewise, the networks that emerge from such computations are in no way inherent in the DNA computer itself; they are a result of the recontextualization that occurs in the biological layer of the network, as well as a result of the decoding procedures specific to the particular mathematical problem being computed.

That being said, it is worth pointing out that, from the vantage point of biocomputing, there is an isomorphism between the two layers. Between the multiple DNA fragments (or nodes) placed in a PCR thermal cyclor to facilitate their binding (or edges) and the resultant directed Hamiltonian graph there is a correspondence based on the specific relationships between discrete nodes. On the biological layer of the network (DNA binding via PCR), we have a number of DNA fragments, some which code for nodes (i.e., "cities" on the Hamiltonian path) and some of which code for edges (i.e., "roads" between cities that make up the directed path). If we view these DNA fragments (nodes and edges) spatially, what we are presented with is a tangled mass of points and lines, some of which will connect, and some of which will not. The task of the DNA computer is to perform the "computation," which in this case is the routine binding of base pairs. The task after this, however, is to extract from this mass the particular set of points and lines that represent the most efficient solution to the problem. This is done, as we have seen, by a particular means of decoding the sequence so that it produces a network (from string/sequence to graph/network). The resultant graph therefore bears some relationship, however direct or indirect, to the particular set of interrelated DNA fragments in the PCR cyclor. In other words, the way in which the DNA computer's processor works is to move along a three-step process: from a distributed set of DNA fragments (nodes and edges), to a linear DNA sequence (node-edge-node), to a graph or pathway with nodes distributed in space (the directed Hamiltonian graph). In this three-step "processing" of data, we go from a graph, to a string, to a graph again (or from network to sequence to network). It is this first and third state that are isomorphic, and in this sense biocomputing can be seen as a means of backtracking to find out what has happened in the interactions of biological components and processes.

One way of understanding this difference between these two networks is to pay more attention to the role of time. Studies of computer processors are all concerned with time, especially in "clocking" a processor's ability to handle computationally intensive tasks (such as image processing). However, by time we mean something different than processor clock time, though not unrelated to the notion of time as a quantifiable unit. Our question is, in the potential relationships that exist in the DNA computer before processing occurs, what kind of a network exists? One way of approaching such a question is to pursue what we mean by "potential" relationships that constitute a network. Surely, from a more technical point of view, a network only exists by being materialized in some way; otherwise the network "in potential" is simply a totally connected network (because every node is potentially connected to every other node). From the vantage point of DNA computing, a particular type of network is being searched for—in Adleman's case, one that fulfills certain requirements of the Hamiltonian path problem (being directed or one-way, beginning and ending nodes, passing through each node only once).

But this computational, mathematical network that is searched for must somehow pass through a set of biological relationships, relationships that constitute the "proces-

sor" of the DNA computer. There are actually two types of processing, two types of temporality, in the DNA computer. Just as we have two types of network layers in the DNA computer, we also have two correlative time-based modes as part of the DNA computer processor. The first type of time is the "binding time" of the molecular interactions in the PCR cyler, a biological time configured by base pair complementarity (in the Adleman example). Actually, we need to be cautious in calling this "biological time," because the PCR machine does not aim to replicate or simulate *in vivo* conditions. Rather, the PCR machine isolates a set of particular biomolecular interactions (annealing and denaturing of DNA strands), and, through cycles of heating and cooling, intensifies and speeds up the biomolecular interactions that participate in the replication of DNA strands. (It is for this reason that PCR is a standard laboratory technique, for it enables the mass replication of any DNA sample for analysis and experiment.) However, we can provisionally refer to the DNA computer's processor as operating through biological time in the sense that the PCR cyler isolates and intensifies biomolecular interactions that regularly take place in the living cell.

The second type of time is the "polynomial time" of the particular computation to be solved, a computational time that is a set of limitations placed on the biological binding time of the DNA computer processor (the PCR cyler). In Adleman's experiment, this polynomial time is set by the class of mathematical problem to be solved, a problem class that is defined by its exponentially large search field. As a mathematical class, the NP-complete problems thus place certain sets of computational constraints on the way in which the binding time is configured. In one sense, this is obvious, in that the PCR cyler will be configured so that it optimally accommodates the kinds of results that the polynomial time requires (i.e., minimal cycling to enable a path sequence to anneal that matches the predicted outcome for the five "cities" or nodes of the problem). In this way, the polynomial time is not a separate process from the binding time of the PCR cyler. The processor of the DNA computer encompasses both, though in different ways. While the binding time dictates a set of possible biomolecular interactions that constitute the "engine" of the processor, the polynomial time nuances the binding time by setting a series of generalized result parameters (i.e., a path made of seven nodes, with a degree of two, with specific beginning and endpoint nodes). Just as the computer network layer is grafted onto the biological network layer, so the computational polynomial time is enfolded onto the biological binding time in the DNA computer processor. As time-based processes, one does not happen before or after the other; rather, their concurrence (of processor capacity and computability) defines an integrated temporality that is defined by the twofold constraint of biology and computability.

In a sense, all models for biocomputing—membrane, tiling, signaling—configure their processors according to this twofold, enfolded process of biological binding time and computational polynomial time. For Adleman's experiment—and arguably for biocomputing generally—the relation between the network preprocessing (binding

time) and the network postprocessing (polynomial time) is that between the possible and real. A possible set of networks exist preprocessing, a possibility instantiated by the biological process of base pair binding. This possible network is therefore a network of combinatoric relationships. This network is negated in the postprocessing network in that, of those combinatorial possibilities, only one will be selected as the solution to the problem. The graph at the end of the experiment therefore simultaneously realizes a potential in the preprocessing network and negates that network as potential (because there can be only one).

### Molecular Molecules?

If the biocomputer's processor functions via a binding time and a polynomial time, we can return to the question of network "layering" and ask: how are the computational and biological network "layers" coordinated in the DNA computer? That is, what might the isomorphisms between the biological and computational layers of the biocomputer tell us about the possible network dynamics common to both? This question pertains equally to philosophy and to computer science, for it brings together the questions concerning self-organization in biological systems with the questions concerning computability in biocomputers. What is needed here is a way of understanding the heterogeneity of networks in biological-computational systems such as biocomputers.

In a discussion of the common characteristics in packs of wolves, swarms of rats, transubstantiations in sorcery, the processes of memory, mineral "life," and epidemic contagion, Gilles Deleuze and Félix Guattari use the term *molecular* to describe dynamic changes in systems that exist above and below the level of the individual.<sup>41</sup> The concept of the molecular is defined not by its scale, but rather by its dimensions. The difference between a single wolf and a pack, between a single cell and a network of cells, is not simply number, but rather the modes of interaction that constitute a pack and an immune system as a network. The molecular is the phenomenon of intensity (aggregate dynamics) as opposed to extensity (movements of particles), qualitative transformations as opposed to quantitative analysis.

Central to the concept of the molecular are the concepts of becoming, difference, and multiplicity. As Deleuze and Guattari state, "the molecular is becoming," or the concept of the molecular implies dynamic transformation in organization. However, this notion of becoming is not be opposed to a notion of static "being." Rather than saying that a set of individual wolves becomes a pack, or a set of individual cells becomes an immunity network, we should say that there is becoming in each of the wolves, in each of the cells.<sup>42</sup>

Becoming is connected to two other concepts, difference and multiplicity. By difference, Deleuze and Guattari do not mean a negative notion of difference ("A is not B"), but rather a positive notion of difference that proceeds via a continuous internal restructuration in time ("A is not A"). This generative notion of difference creates



novelty not by making anew, but by repeating itself, by repeating its internal differentiation ("I is another"). This principle of differentiation contributes to the third concept related to the molecular, that of multiplicity. As Deleuze and Guattari note, multiplicity can be both qualitative (a cluster, a bunch, a group, a pack, a swarm) and quantitative (clustering coefficients, power laws, graph topologies). Multiplicities are both singular ("a" pack, "a" swarm, "a" network) and plural (a pack of wolves, a swarm of insects, a network of cells).

Taken together, the concepts of becoming, difference, and multiplicity articulate the ways in which the molecular manifests itself in any network. The molecular can be described as a phenomenon that expresses the transformative dynamics of becoming, which is enabled by a principle of internal differentiation, which proliferates differences over time to create multiplicities.

In our consideration of biocomputing, we have seen several networks in action: the networks of biological components and processes in the living cell, the networks of interactions of those components in lab technologies (synthesizers, PCR, electrophoresis), the networks described by mathematical problems in graph theory (directed Hamiltonian path problems), and the abstract networks modeled by such mathematical problems (e.g., geographical networks of cities in the traveling-salesman problem, or information networks in routing Internet data). When we look at biocomputing as an instance of "the molecular," how does our understanding of these different networks change?

From a philosophical perspective, the preprocessor network of the DNA computer (the mass of DNA fragments in the PCR cyclor) is an instance of a virtual network, in that the combinatoric characteristics of the network exist not to solve any one particular problem, but rather to continuously differentiate, and thereby establish connections between nodes. Furthermore, if we recall that the coding scheme of nodes and edges has been embedded into the DNA fragments (with no functional alteration to the DNA), then the type of network formed in the preprocessing state is one of "edges without nodes."

This network of edges-without-nodes is at first nonsensical, but also, in another way, rigorously empirical. How can a network exist without nodes? Don't the nodes precede the edges? Aren't they in fact a necessary condition for the action or movement of edges to exist? In any given network—the Internet, a social network, a metabolic network, an economic network—it seems to make sense that the discrete objects we call nodes (computers, people, cells, corporations) exist prior to the actions they effect (data transfer, enacting of dialogue, enzymatic reactions, fluctuations in value). If there is no subject, how can there be any action, any intentionality, any "directed" links between subjects?

On the other hand, it is equally clear that a static network is not a network at all. A group of computers can have cables attached between them, but if there is no activity on individual computers or between them, a network exists only hypothetically. Net-

works are materialized by their actions, by the edges that create proximities, alliances, and condensations between discrete nodes. In a sense, we can say that, although networks seem to depend on the preexistence of nodes, they also are constituted as networks by the edges between nodes. In the cell, a network without edges is death, just as a network with only edges is metastasis. Networks are always active and activated; they are always relations whose terms may very well change over time.

If this is the case, then it would make sense to reconsider biocomputing in light of a concept of edges-without-nodes. As already noted, biocomputing constructs two layers to its unique bioinformatic protocol: a computational layer, represented by the directed Hamiltonian path, and a biological layer, represented by Watson-Crick base pair complementarity. The molecular biology techniques of DNA synthesis, PCR, and gel electrophoresis provide the media through which one layer touches the other. What is the difference between these two layers in biocomputing? We can begin by suggesting that the computational layer, informed as it is by modern graph theory, is a network of discrete dynamics. This is illustrated in the graphical representation of the traveling-salesman problem (directed Hamiltonian path) (Figure 11), as it is illustrated in other graph-theoretical depictions that utilize a diagrammatic language of nodes and edges. In other words, the computational layer, working from a basis in graph theory, necessitates a topology of clearly demarcated nodes (static entities) and edges (effected actions). In this way, the computational layer is "computational" according to a digital, binary logic of Boolean operators (AND, OR, XOR).<sup>43</sup>

By contrast, the biological layer is "computational" in an analog manner, not unlike models of physical computers based on water, mercury, or mechanical parts. The various biological models for biocomputing—from DNA base pair binding to protein-protein structural specificity—are all based on processes that are continuous. Note that such processes can be interpreted as being discrete (e.g., genes being switched "on" or "off" in gene expression), but there is no separation between the medium and the message in the living cell. Indeed, this has been the primary technical challenge for both bioinformatics and biocomputing: to integrate the digital/computational and analog/biological layers of computability into a single whole.

If, from a philosophical standpoint, the main challenge for biocomputing is to functionally correlate its two protocological layers (the computational and biological layers), then we can suggest that Deleuze and Guattari's notion of "the molecular" can serve to prompt new modes of thinking in the understanding of how computers and biology mutually implicate and transform each other. The model of biological process in traditional molecular biology and biochemistry is one in which molecules are nodes, and the physical-chemical interactions between them constitute edges.<sup>44</sup> In other words, the view of biomolecular networks in biology and biochemistry is one based on the graph-theoretical model of discrete nodes that precede the effectuation of edges between them—the same digital model of computation seen in biocomputing's computational layer. But we also see that there are important differences between digital



computers and living cells from a network perspective. Although one can be reduced to the other, it would also seem important to consider a notion of networks—in particular, biomolecular, biocomputational networks—that begins from the dynamic perspective of edges, rather than the static perspective of nodes. Experiments in biocomputing such as Adleman's show that computational problems can be solved, but that such novel, hybrid systems also generate networks at other levels as well. In the case of biocomputing, "molecules" are not necessarily "molecular."

## CHAPTER FIVE

### Nanomedicine

#### *Molecules That Matter*

#### Small Body Problems

To begin with, we can discuss the large effects of the very small. Linda Nagata's science-fiction novel *The Bohr Maker* envisions a near future in which the ability to control individual atoms has become a reality, a primary theme in the emerging scientific field of nanotechnology.<sup>1</sup> Although a number of science-fiction works take up this nanotechnological theme of control on the atomic scale, *The Bohr Maker* is notable because of the way in which it provides a specific meditation on how the human body may be transformed by nanotechnology.<sup>2</sup>

*The Bohr Maker* takes place in a world defined (and redefined) through nanotechnologies and socioeconomic divisions enhanced by the access to the design of those technologies. Although the future world of *The Bohr Maker* is divided into a first-world Commonwealth and a predominantly Southeast Asian third world, "citizenship" and even subjectivity itself are defined by nanotechnologies at the level of the everyday. In the novel, Phousita, a young Indian woman living in the slums of Sunda, comes into possession of a device called a "Maker." Unbeknownst to her, this device has been smuggled out of government research labs, where it has been kept under lock and key. The Maker is a nanotechnology device (a kind of nanotech PDA) invented by the scientist Leander Bohr. When injected into a person, it enables that person to modify certain biochemical features of the body, including genetically engineering one's own genome, enabling cellular and molecular regeneration (biological life extension), and enhancing neurological capacity (intelligence boosting). When Phousita first feels the effects of the Maker, she and her partner Arif attribute it to either a mysterious illness or even spirit possession. At this stage of the novel, Phousita's body is approached as an infected body.