On the Complexity of Biological and Living Systems and the Principle of Computational Equivalence

November 21, 2014

Sahil Loomba

2012CS10114, IIT Delhi

Abstract

The central objective of recent strides taken by biology and biochemistry, has been to *simplify* the complexity of biological systems, and of life itself. While we have developed fields such as systems biology, to better understand molecular interactions that govern metabolic pathways, signal transduction and genetic expression within living cells, we have much older and popular theories such as Darwin's theory of evolution that try to explain the existence of and diversity in life, the way it is. We wish to engineer living and biological systems to suit our own purposes, like increasing longevity of life, treating diseases, developing "synthetic life" for reducing human effort, etc. But in our rush of progress, what we often ignore is the question of pragmatism and computational intractability. The theories and models we propose, and the questions we try to answer, serve no utility if they are impractical for application, subject to the current limits of *our* mathematics and computational abilities. There are, of course, various problems in mathematics and computer science that have been deemed unsolvable (like the Halting Problem), or computationally intensive (like the Maximum Independent Set Problem which is NP hard), but can similar limits be imposed on problems in biology? Our intuition tells us *yes*, and certain recent advances in the theory of computation, cell automaton such as the Game of Life, and the principles of Computational Equivalence and Computational Irreducibility, as coined by physicist Stephen Wolfram, are in overwhelming support of this intuition. We follow these principles, and view systems under the lens of computational pragmatism, to uncover the possible limits imposed on our understanding of Biological Sciences, the way we know it.

Keywords*: biological complexity, principle of computational equivalence, computational pragmatism, determinism, human brain*

Contents

1 Introduction: Characterising Complexity

What defines the complexity of a system? Complexity is, at a high level, a metric for characterising systems with multiple parts, interactions and visibly sophisticated behaviour. It is, indirectly and usually, a measure which is proportional to: the level of difficulty of technique, the amount of time and expense of energy, applied for its analysis. Let us take a closer look at how complexity manifests itself within two systems that are important for establishing the foundation of the arguments that follow.

1.1 Biological and Living Systems

A **biological system** is often regarded as a complicated container of interrelated biological entities, which could be established on different scales: from the molecular, to the genetic, to the organelle, to the cellular, to the tissue, to the organ, to the body and eventually, to the ecosystem itself. Some of these biological systems are special, in the sense that they are *alive*, which is when they are called **living systems**. The definition of the signature of life has itself been a widely researched and debated idea, with biologists and philosophers arguing over it for centuries. The main reason for the difficulty being that *life* is not an embodied object, rather, it is a process that is undergone by certain biological systems, like bacteria, plants and animals. Moreover, a large variety is observed across various lifeforms, which makes it hard to fit a single definition, generalised over all instances of life. Biologists have come up with a convenient criteria for defining lifeforms (convenient, because it aids in being an established starting point from which further analysis, and fields like synthetic biology, can take off), something which they call the three operational functionalities[1]:

1. They exhibit an exchange of energy.
2. They have a chemical realisation of information control that can be passed on in a replicable manner.
3. They possess a boundary that encloses the above two, to create a system segregated from surroundings.

Philosophers might squirm over this definition, and laymen may find it too simple to fit the diversity of life they see. And indeed, one of the challenges that biology faces as a science, is to fit an idea to one and to all. Which is why it is often quoted as a science of exceptions. It is easy to see that if the definition of life itself is so widely debated, where do we even begin to break down biological and living systems? Indeed, biology tries to understand some of the most complex, specialised, and high-information objects in existence. (Erwin Schrödinger went on to define life as a process which decreases its entropy by feeding on negative entropy[2].) However, the fields of physics and chemistry, whose positions have been well established in the realm of science, seem to contain more consistent and predictive theories and laws, describing the behaviour of point particles and molecules. At this low level of abstraction, the entities involved show much simpler and observable behaviour. This led biologists and philosophers alike, to the belief that biological studies could be reduced to these simpler physiochemical phenomena described by physics and chemistry (something called *epistemic reduction*). After all, it is fundamental particles like quarks, fermions (which include electrons), bosons and mesons that build up all matter and energy in the Universe.

The French philosopher Descartes was one of the first people to analyse the tool of reductionism, wherein any system can be analysed by investigating the individual parts of the system in a mechanistic manner. This tool seems useful when one reads about the history of science, particularly physics, and it was adopted by the early Molecular Biologists of the 20^{th} century. So much so that Crick, the co-discoverer of the structure of DNA, went on to say in 1966 that "the ultimate aim of the modern movement in biology is to explain all biology in terms of physics and chemistry"[3]. However, reductionism doesn't seem to hold much significance in contemporary science, simply because expressing an object as a sum-of-parts is too simplified an assumption to be able to define and predict the behaviour of systems that are studied today, accurately. The Greek philosopher Aristotle had described this anti-reductionist belief in his treatise *Metaphysics* as "the whole is more than the sum of its parts"[4]. The main criticism of reductionism is that it blatantly ignores a very important feature of most complicated and biological systems: interdependencies. Biological systems show relationships between parts at every level: molecular, cellular, right up till organisational. Therefore, studying individual parts in isolation would rarely build the whole picture. For example, the signalling pathway in a living cell which leads to apoptosis (programmed cell death), has numerous feedbacks and feedforwards between various classes of proteins that interact with each other to 'decide' if the cell will commit suicide or not. And as these signaling networks get larger and larger, the power of molecular biology weakens further in further in modelling them well.

As one dwells deeper into the complexity of biological systems, it doesn't take much time to realise that the dream of reducing it to physics and chemistry will never see the light of the day. Because the moment we move to

higher levels of abstraction, from the levels of particles and simple molecules with countable types of behaviour, to the levels of interconnected entities and biological networks with seemingly uncountable kinds of behaviour, there seems to be a sudden jump between the levels. Somehow, there is a gap which the objects overcome, across which the objects of that level have become very complex and show often chaotic and unpredictable characteristics. For instance, we have been successful in characterising the 21 essential amino acids that make up all proteins, but predicting protein refolding (sequencing of these amino acids to form protein chains) is a very complex and hard task[5] (in fact it is NP-hard, covered in Section 3.1), for which we have only approximate solutions, that too for the *easier* problem instances[6].

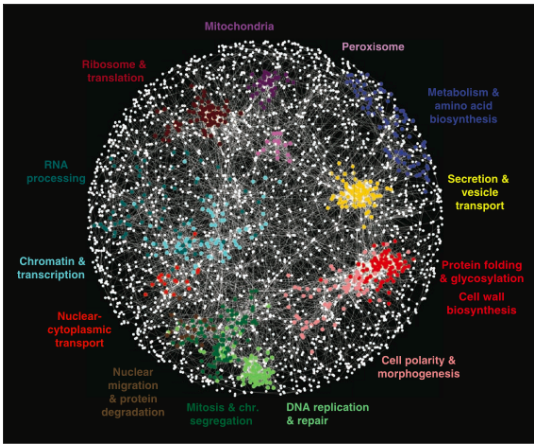

Figure 1: A network representation of pairwise gene interaction in Yeast[7]. This gene network shows the high density of relationships between various nodes (entities), and the reason of inseparability of different parts of this network, for a complete and accurate analysis.

The discussion on biological complexity is incomplete without talking about the most complicated piece of organic machinery known to mankind: the human brain. As one moves from the level of cellular signal networks to the level of a complicated organ such as the brain, the interactions between individual parts become even more intertwined and complicated. The human brain, as per latest and closest estimates, consists of 86 billion neurons[8], close to half the number of stars in the Milky Way galaxy. To appreciate the computing power of this *machine*, even if one encodes the brain as a state machine with every neuron being either ON or OFF, we have $2^{86000000000}$number of states. One may note that there are an estimated 2^{300} number of fundamental particles in the observable universe. The real power of the brain however, doesn't lie in the neurons themselves, but the immensely high number of connections (synapses) they form with one another, 125 trillion by some estimates[9]. The numbers alone are sufficient to show the wondrous capabilities and unpredictabilities of the brain. Unlike practical computers, the brain is highly decentralised and processes information at very fast rates with a lot of parallelism, which is why we are able to do complicated non-linear tasks such as identifying shapes, faces and textures with a lot of ease and non-expense of energy (unlike computers), whereas computers are much quicker in doing seemingly linear tasks like basic arithmetic. We have been able to quote Darwin's theory of evolution and natural selection to attribute to the creation of such a beautiful piece of biological machinery (as the evolutionary development of the cognition controlling neocortex), but unable to fathom the processes of brain functions. This is because the brain keeps moving, seemingly continuously, from one state to another, making new connections and breaking old ones forward in time, reaching the same state never again.

Philosophers have branded this dynamic nature of the human brain, or any other brain for that matter, to the idea of consciousness and free will. Widely ununderstood, consciousness and free will is the black box which encases the human brain's behaviour; a box which appears to be impenetrable to our attempts at modelling it accurately to claim any strong predictions about it. Ramon Cajal, the father of neuroscience, was a pioneer in deciphering the structure of the brain. He studied neurons across species, particularly in a pigeon's cerebellum, to sketch small localised networks in which the neurons were connected. Since then, we have come a long way in describing locations in the brain from where various bodily functions are controlled, thanks to the phenomena of loss-of-function due to neuronal damage, and modern imaging techniques like MRI. However, the brain has never been described in its entirety. The reason is simple: the numbers are too large to be contained within a reasonable model. So the question is, can we break the illusionary black box and decipher the workings of the brain and perhaps even predict its behaviour, through a reduction of the brain to a smaller problem with a lower level of

computational complexity? Like a map legend, where one kilometre is not one kilometre, but one millimetre? The outcome of such a development would be exciting to say the least. In fact, it would be a paradigm shift in modern biology, psychology, sociology, economics and (arguably) one of the biggest achievements of human civilisation. One could even say that it would change our entire perspective of understanding Life and the Universe, the way we know it.

Evidently, we have been unsuccessful to do so till date (otherwise George Orwell's *1984* might have as well been based on a true story!). And as we'll see in Section 3, it is almost certainly unlikely that we'll ever be able to reduce the brain to another system of lesser computational complexity.

1.2 Computational Systems[1]

In the field of theoretical computer science and mathematics, computational systems are essentially equivalent to *programs* (which have the same connotation as the one used in everyday computer science jargon). In the theory of computation, every program can be represented as a bitstring (a string of 0s and 1s), which can be run on a 'computer' to obtain the result for which the program was intended and written. Before we understand how one can define the complexity of a program/system, we must take a closer look at the theoretical foundations of the usual, and now ubiquitous, computer.

The modern computer architecture, often attributed to the polymath John von Neumann, in essence consists of an input device (keyboard, mouse), an output device (hard disk storage, monitor), a central processing unit (the processor), and a memory unit (the RAM). This architecture itself is based on a theoretical abstract idea which was proposed by the computer scientist Alan Turing, called Turing Machines. A Turing Machine has an infinite tape (corresponding to infinite memory), a head (analogous to the input device), a finite table and the state register (consisting of the program instructions, essentially the 'bitstrings', and the state of the machine, both of which are stored in the RAM of a modern computer). Note that a Turing Machine is physically inconceivable, and computationally more powerful than a practical modern computer, because of the presence of infinite amount of memory. A Turing Machine is just one model of computation (also the most powerful), but it is further preceded by another model of computation called Finite State Automata (FSA).

The finite state automata model consists of discrete states, with certain transition functions that govern what the next state will be, depending on the value of the current state. If the transition functions are deterministic, they are called DFAs (Deterministic Finite Automata), else if they are non-deterministic (that is, certain probabilities are associated with the transition function, for example: state A can go to states B or C with equal probabilities of 0.5), then they are called NFAs (Non-deterministic Finite Automata).

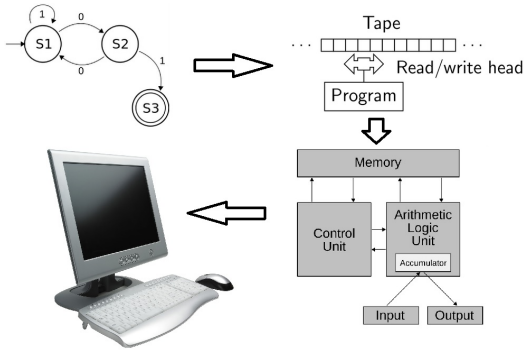

Figure 2: The progress leading to the creation of the modern computer, starting from the theoretical models of FSA and Turing Machines, going up to the von Neumann Architecture.

Armed with the idea of DFAs and NFAs, we can now move to defining complexity of a program. For every instance of a problem, represented as a bitstring of length `n`, its complexity (or more importantly, the time complexity) is a function `f`, which maps `n` to the number of basic computational steps (basic steps could be defined as the elementary operations of add, subtract, load and store from memory, etc.) required for the algorithm of this program to terminate. That is, `f(n)` is a measure of the amount of time it will take for an algorithm to run on a particular problem instance. Usually, to eliminate the variation that can creep in because the selected instance may be easy or difficult, we evaluate asymptotic `f(n)` for the worst-case scenario.

[1]This subsection may be skipped if the reader is comfortable with the ideas and notation of computational complexity classes.

Computational complexity can be used to define the idea of ***efficiency*** of an algorithm. An algorithm is said to be efficient if it runs in polynomial time, that is, if `f(n)` is a polynomial in n. Otherwise, say if `f(n)` is exponential in `n`, then the algorithm is not considered to be efficient. This leads us to the question, does there exist an efficient algorithm for a given program/system? This highly important question remains unsolved for a variety of problems in the field of computer science. If a program has an efficient algorithm, all of its instances can be solved in an *easy* manner using that algorithm, and if a program has a non-efficient algorithm, then at least some of the instances will be solved in a *hard* manner using that algorithm. However, can we somehow relate problems and classify them based on the idea of efficiency and complexity?

Yes, there is a large number of problems (accounted for after conversion into their decision versions, which end up in a yes or no answer), that have been classified into computational classes. Class **P (Polynomial)** is the class of problems which can be solved using an efficient (polynomial time) algorithm. Class **NP (Non-deterministic Polynomial)** is the class of problems which have an efficient certifier, that is, there exists an algorithm **A** which, given the problem (program) and a yes/no instance of the problem, can certify whether it is a yes or a no instance of the problem (by outputting, say, 1/0), in polynomial time. Evidently, class P is a subset of NP, since an efficient algorithm for a program is an efficient certifier itself. Now, we define the class **NP-Hard**. A problem **X** is said to belong to class NP-Hard if every problem in class NP can be reduced, in polynomial time (time $\equiv$ number of computational steps), to that problem. In essence, these are the **hardest problems as far is complexity is concerned.** Such problems are often referred to as being **computationally intractible.** Furthermore**,** the intersection of NP and NP-Hard is called NP-Complete problems; problems which are the hardest of all problems in class NP.

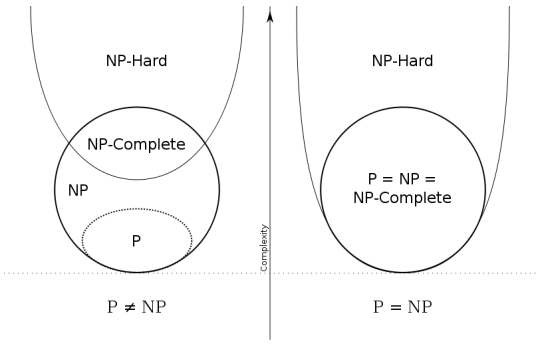

Figure 3: The Euler Diagram describing the Computational Complexity Class if (a) P≠NP and (b) P=NP (Source: Wikipedia)

The implications of such a classification are fairly interesting:

1. Every problem in NP-Hard is at least as hard as all the problems in NP.
2. If a problem in class NP-Hard is solved efficiently, then every problem in NP can be solved efficiently (since every NP problem can be reduced efficiently (in poly time) to that problem in NP-Hard). This would imply that P=NP. Proving, or even disproving this equivalence, remains an open and one of the biggest unsolved problems in the field of computer science! (Also, if proven true, then many of our cryptographic protocols, such as the popular RSA-encryption, would immediately break down.)
3. A problem in P can be solved in polynomial time by a deterministic machine (DFA), while a problem in NP can be solved in polynomial time by a non-deterministic machine (NFA). (Hence, the names of these computational classes.)

The power of this classification lies in the fact that it:

1. provides us the ability to concentrate on solving few problems, which, if successfully solved, would lead to the automatic solving of many other problems.
2. establishes a sense of hierarchy of classes of complexity (where we have sudden jumps in the complexity as one travels from one set to another), wherein the problems belonging to a lower class of complexity can be easily solved, but the ones in higher classes cannot be solved easily in our current capacities of modern computing.

Although it is an unsolved problem, computer scientists largely believe that P≠NP, and thus, we would almost certainly never find efficient algorithms for solving such problems in the current computational paradigm.

2 Principle of Computational Equivalence (PCE)

2.1 Mapping Processes to Computations (Church-Turing Thesis)

Stephen Wolfram, in his magnum opus *A New Kind of Science*, tried to show every natural process as a natural **"computation"**. He claims that "all processes, whether they are produced by human effort or occur spontaneously in nature, can be viewed as computations." This assumption, in part, is based on the overwhelming evidence we have for various natural processes; the fact that we have been able to model various small-scale biological metabolic networks, predict electron behaviour in a crystal lattice, roughly model weather systems, predict market conditions of the world economy, etc. The scope for validity of this assumption also lies in the *hope* that we would be able to model a variety of useful natural processes, which is the very driving force behind modern science and engineering. The assumption of treating processes as computations, in fact, implicitly already exists in every scientific exercise.

Once we have established this mapping of natural processes to computations, we can extend our idea of *natural computations* by invoking the Church-Turing Thesis, given by mathematicians Alan Turing and Alonzo Church in 1936. In their own words, the

Church-Turing Thesis states that every effectively calculable function is a computable function.[10]

By **effectively calculable**, they mean that it is created by *any* effective means, while **computable** refers to the fact that it can be computed by a Turing Machine (TM). This seemingly vague statement is actually a very strong claim and conjecture, that has held up to the test of every single problem till date, and is expected to keep doing so in the future. However, it has not been proven by any means, which is why it remains a thesis, and not a theory. In essence, it means that if a function can be computed, then it can be computed on a Turing Machine. This extension of the argument allows us to bring any natural process under the ambit of Turing Machines, in the light of viewing them as computations. The reader may find this astonishing or unbelievable, since real-world problems seem too complicated to be handled by a simple system such as a TM. However, simplicity of a paradigm doesn't imply that it is powerless. Indeed, the fact that the tape is **infinite** in length, provides enough impetus to this computational model to handle arbitrarily large and complex natural systems. An example to look at complex behaviour arise from the simplest of rules is the subtract-and-branch-if-negative (SBN) machine language instruction. SBN can potentially be combined with itself to create all possible arithmetic, conditional and branching operations of any low-level program, which can eventually be combined amongst each other to create more complicated high-level program instructions like "compute Fourier series for a signal", etc. Therefore theoretically, one instruction is actually sufficient to encode almost every possible programing problem.

2.2 Building the Complexity Pyramid (Computational Equivalence and Computational Irreducibility)

Having understood the above assumption, we can now state the Principle of Computational Equivalence (PCE) in all its simplicity and entirety.

Principle of Computational Equivalence states that almost all processes that are not obviously simple can be viewed as computations of equivalent sophistication.[11]

By **obviously simple**, it is meant that something which is obviously repetitive, or nested, or easily modelled as predictive, stochastic or random. Some examples could be:

- the motion of the Earth around the Sun. It is highly repetitve with near to zero changes in the path followed by the Earth every year.
- the projection of a small body across a given trajectory. This can be easily repeated with the same parameters of distance/velocity/force in every subsequent experiment.
- the replication of DNA within an organism. The DNA replicates with a remarkably low error (as low as 10^{-10} in some bacteria[12]) and thus one can be almost sure about its repetition within the organism.

Many more "natural phenomena" could be added to this list with relative ease. But perhaps, even more number of phenomena could be added to the list of processes with **equivalent sophistication**:

- the weather is very highly non-repetitve. It is evident in the way that we have global temperatures changing every year and deviating from the previous standards, or in how we bear the brunt of hurricanes or droughts or floods, with a differential impact across different years. We seem to never come back to a *previous state* of weather conditions.

- the human (or, for that matter, almost any) brain. As already mentioned before, the sheer number of connections are too high, and a human's life is too short, to exhaust all the possible states the brain can be in. Thus, the brain never returns to the same state (which is good for us, else we would lose all sense of the flow of time!).

- heredity and evolution. Consider the case of human beings, where DNA is the basic unit of heredity, that encodes all the genetic information of the organism. While DNA replication seems to be obviously simple, the fact that sexual reproduction goes via the route of crossovers and mutations, ensures that the DNA being passed to the next generation is different from the parents' DNAs (if one excludes the exception of twins). Of course, not every part of the DNA being passed down within the same species would be different. The human genomes consists of around 32 billion base pairs[13]. Out of these, around 2% of base pairs (640 million base pairs) are considered to be the coding regions[14] (regions which code proteins). If, in the worst case, one assumes that there could be any of the two possible base pairs (A-T and G-C), then we could have a total number of $2^{640000000}$ possible combinations. Again, the numbers are overwhelmingly high for the DNA to be repeated across generations, or to reach the same state, making evolution a sophisticated process.

But in what sense are these complicated processes equivalent to each other? The answer lies in their **universality**.

UNIVERSALITY states that a computation is universal if it can emulate any other computation.[15]

Even for processes seemingly very different from one another, their underlying principles would still show a certain equivalence, in the sense that they are governed by rules that can achieve universality. For example, consider the Cell Automata Systems called Rule 110 and Conway's Game of Life. Both of these cell automata show different characteristics (which, at a high level, could be interpreted as a variety of complex behaviour), that arise form different, but few very simple rules. What makes them equivalent though, is the fact that both of have them have been proven to be universal[16]. They can compute any calculable function, much like a Universal Turing Machine (UTM is a Turing Machine which can simulate any other Turing Machine). Therefore, Rule 110, Game of Life and UTM are all computations of equivalent sophistication, in the sense that they are universal. The brain, including other natural processes (read: computations) of sophistication as described above, also belong to this class of computational sophistication, and are equivalent to TMs and Game of Life in this sense.

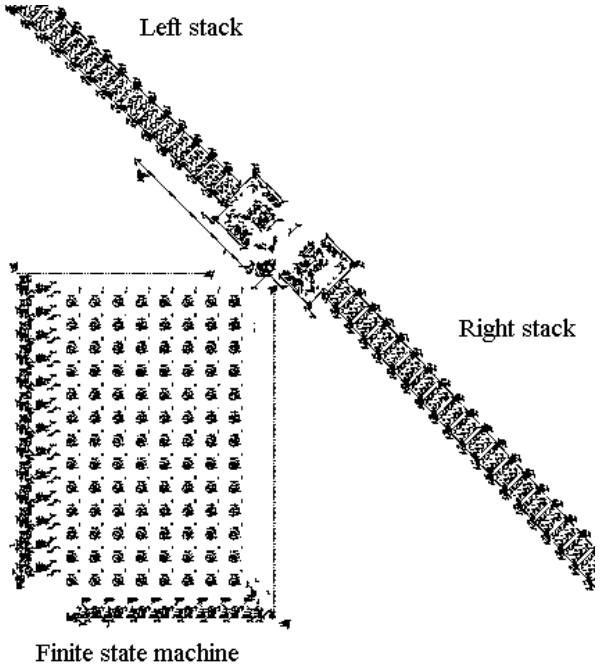

Figure 4: A snapshot showing a Turing Machine simulated on the cell automata Game of Life (Source: Pentadecathalon)

Note that the conditions for universality can be very simple indeed. Complexity of behaviour does not imply complexity of rules. A TM has very simple parts and rules of reading and writing, but its infinite tape (although not just on its own) makes it highly powerful. Similarly, Game of Life has three very simple rules of deciding whether a cell in an infinite array is dead or alive, but the infinite cardinality of the array (again, not just on its own; some cell automata of infinite array size are not universal) makes it highly powerful. [Availability of infinite memory appears to be, logically, a necessary (but not sufficient) condition for universality. Or the size should be at least as large so as to contain any arbitrary practical problem.]

The question that arose earlier in our discussion of complexity, that whether we can reduce highly complicated problems to problems of fathomable complexity, becomes significant here. To find that "map-legend" which gives a good reduction, we need to somehow be able to talk about and describe the problem via a shortcut-of-kinds, without actually executing the problem itself. For instance, computing the sum of integers from 1 to 100 could be done by computing "1 + 2 + 3... + 100", which would be the original problem, or one could reduce the effort, time and energy of computation by simply plugging in 100 in the mathematically derived formula $\frac{n}{2}(n+1)$.

The primary objective of mathematical and numerical analysis has essentially been to create such map-legends for all the problems and processes that we observe in different fields of pure and applied science. But is it possible to create map-legends for all such problems? Or are there some problems that are irreducible to a simpler form? This brings us to the third critical computational concept of this paper:

Computational Irreducibility is the idea that most of the time, we want to derive some mathematical formula which allows us to determine the outcome of the evolution of a system without having to explicitly trace the steps involved[17]. But certain (and certainly, most of the) real-world problems are computationally irreducible, that is, there is no shortcut to determining the fate of the system without having actually simulated through it.

Let us look at this principle in the light of some of the problems discussed above. Conway's Game of Life, for example, has three simple deterministic rules, that given the initial state of the game (a description of which cells are alive and which are dead), will progress deterministically. One might be curious to know if one could predict the behaviour of lifeforms as the Game progresses, or say, what would the condition of the Game be after 50,000 generations. Is there a shortcut formula or algorithm to find out this $50{,}000^{\text{th}}$ state? Turns out, there isn't one[18]. The only way to know the condition of the Game after $\boldsymbol{t}$ time steps is to, well, run the game for $\boldsymbol{t}$ time steps! Therefore, the Game of Life is irreducible. The observant reader would have realised by now, what this means for other problems belonging to the class of sophistication equivalent to that of the Game of Life. Turing Machines are, indeed, irreducible to any other model of computation, which is, unsurprisingly, the reason that they hold so much power and universality in solving other problems! Add to them other natural process like the brain, protein sequencing, weather modelling, etc., and it's easy to conclude that since all of these problems are of equal sophistication, if one of them is irreducible, so are all others.

This analysis of computational classes is analogous to the computational classes of complexity defined in Section 1.2. Just like we have a hierarchy or pyramid of problems, going from NP-Hard at the top to NP-Complete to NP and finally to P at the bottom, we have a hierarchy going from the class of sophisticated at the top to the class of obviously simple problems at the bottom. (There could indeed be more number of classes in between, if a more rigorous and mathematical analysis is done to describe these classes of sophistication concretely.) **The idea being that we can reduce a problem in the direction of going from bottom to up in both of these complexity pyramids**. However, problems are irreducible in the direction of top to bottom.

The above idea has been implicitly in use for as long as computers have existed and have been used for computational analysis of scientific phenomena. The fact that we are able, and we are hoping, to reduce all our (seemingly simple) problems into computer simulations and algorithmic analysis shows the reducibility of simple problems to TMs (a computer is essentially a TM, except that its power is limited by finite memory). Also, as Wolfram states, "if meaningful general predictions are possible, it must at some level be the case that the system making predictions can outrun the system it is trying to predict", a statement which succinctly and intuitively defines the whole idea of PCE and Computational Irreducibility.

3 Implications of PCE

3.1 Ideas of *Hardness* in analysing Biological Systems[2]

As mentioned in Section 1.1, early molecular biology was based on the idea of reductionism. But soon, when the complexity of biological systems was started to be appreciated better and biologists realised that their models were causing oversimplification of biological processes, it gave way to mathematically more rigorous branches of biology. Evolutionary biology was looked at from mathematical persepectives, converted to the field of population genetics[19]. More recently, systems biology has been at the forefront of establishing a mathematical analysis of biological systems.

Systems biology is the study of the behaviour of complex biological organisation and processes, in terms of the molecular constituents[20]. Among the basic tenets of systems biology, the one that deviates furthest from the older idea of reductionism is that of *emergent properties*. These properties are defined as "new properties" which emerge, that are not a part of the individual components that make the whole (cell)[21]. Life is treated as one such emergent property, emerging out of DNA, RNA, proteins, etc. Some of the techniques used in modern systems biology include metabolic flux analysis, metabolic control analysis, physiochemical modelling (which achieves inversely proportional amounts of predictive power and scope), network modelling, etc. However, one of the biggest challenges which this discipline faces is that of the expensive (both in terms of time and resources) computation and data organisation. Moreover, one of the main points of focus of systems biology is on synthetic biology. While branches such as the omics (proteomics, genomics, metabolomics) and bioinformatics are going from top-to-down, by studying macromolecular interactions, studies in reaction kinetics and molecular interactions are trying to cover ground from bottom-to-up. The challenge lies, really, in clearly establishing the murky domain where both of these approaches meet, at the centre of the pyramid: **the jump point of complexity**. As Prof. J. Gomes of the School of Biological Sciences, IIT Delhi suggests, "the success or failure of Synthetic Biology as an engineering discipline depends on where independence approximations become useful in the continuum between the atomic and macroscopic worlds."

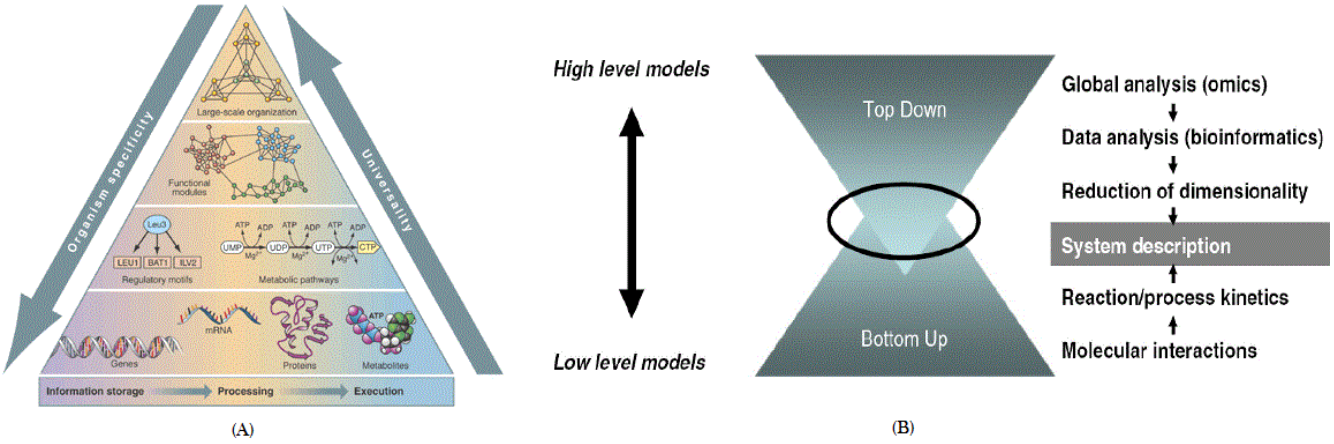

Figure 5: (A) A pyramid showcasing the complexity of life and (B) the top-down & bottom-up approaches being taken by synthetic biology. (Source: (A) ScienceMag (B) Prof. J. Gomes, IIT Delhi)

Can this gap of complexity be uncovered effectively? And how far above in the macromolecular world will we be able to accurately describe and predict biological behaviour? Biology today deals with such challenging questions, and system biologists are working hard to find ways to answer them. However, any conclusive evidence is yet to be seen about the indisputable proving or disproving of these tall claims of systems or synthetic biology being able to predict and model biological elements as complicated as lifeforms. When one refers to the fascinating book by John Wooley and Herbert Lin, titled *Catalyzing Inquiry at the Interface of Computing and Biology*, one realises the plethora of computational challenges which face biologists today. Appendix B[22] of the book lists out a large collection of highly critical and exceptionally challenging problems in computational biology, like protein structure prediction, investigating integral function units, bridging the gap between computationally feasible and functionally relevant time scales, full genome-genome comparisons, rapid topological clustering of proteins, evolutionary model

[2]A major part of this subsection is attributed to the lectures and discussions with Prof. James Gomes, of the School of Biological Sciences, IIT Delhi.

building, genetic circuits, the brain and artificial thinking machines, model brain function, model simplification, exhaustive discovery and analysis of cancer genes, etc. (the list contains close to 200 such computationally intensive and grand challenge problems).

As briefly mentioned in Section 1.1, the protein folding problem using the Hydrophobic-Hydrophilic (HP) Model was proven to be NP-complete[23]. Now, for such a problem, all instances of it may not be hard, but some of these instances are definitely hard to solve for (refer to Section 1.2). And indeed, system biologists have been trying to sequence amino acids to study protein formation, but have been unable to do it efficiently beyond even 4 or 5 amino acids! Even for studying biological boolean networks (BBNs), used to study metabolic pathways in cells, analysis of identifying attractors, and finding control strategies is NP-Hard[21]. This direct and established correspondence between algorithms of the highest computational complexity classes and processes (computations) belonging to highest classes of sophistication, as described by PCE, is further in overwhelming support of these two pyramids of biological and computational complexity, as described in Section 2.2, being analogous to one another. If this analogy is explored and strengthened further by coherent and collaborative studies between theoretical computer scientists and biologists, we could potentially intertwine these two pyramids to build a single pyramid of Complex Systems in both the Abstract (computational complexity) and the Real (biological complexity) and work forward in this unified domain where *everything* is a computation, and some problems can be reduced to other harder problems that lie at the top tier of the pyramid.

3.2 Determinism, Prediction and Free Will (Newcomb's Paradox)

The ideas of determinism, prediction and free will have been central to philosophical thinking for time immemorial. Determinists hold on to the belief that every state (or condition) is completely determined by the previous states. Or, in other words, for every event we see, no other prior states could have led to this event to occur. Different thinkers have held different degrees of belief in this concept, with some being hard determinists while other compatibilists. Hard determinism directly contradicts with the idea of free will, which is the belief that any living entity, and in particular human agents, have the ability to make their own choices, independent of the preceding conditions or states. Libertarianism is the other end of the spectrum, which has full faith in free will but holds no truth in determinism. In between these two extremities, lies other varieties of thinkers and believers, who are either too confused or too fearful to take a strong stance, or perhaps intelligently cautious enough to not pass a judgement on an unsolved problem with open fire on both sides of the argument.

Some psychologists, like Skinner, believe that the idea of free will is nothing but a label for a black box of consciousness that we don't yet understand. He says that "it is my strong belief that the basic mechanisms of human thinking will, in the end, turn out to correspond to rather simple computational processes." If the idea of free will is indeed an apparent illusion, perhaps PCE and irreducibility hold the power to show how this illusion comes into being, and if we will ever be able to break out of it.

By the end of Section 3.2, we have established how highly sophisticated biological systems, which include the human brain, are computationally equivalent to models of computation such as UTMs. They both lie in the top-most tier of complexity, and cannot be reduced to a shortcut problem which might lie somewhere below in this spectrum of complexity. Thus, the best we can do to "predict" the human brain is to really, just simulate it, or run it, on its own. Since even our best computers, which are mere limited versions of UTMs, lie in the same equivalent class as the brain, the brain cannot be "run" on them. The intuitive reason is similar to what Wolfram says about the necessity of the predictor computation to "outrun" the predicted computation. Thus, since both posses equivalent levels of sophistication, one of these can never outrun the other. The question that arises then is, *what can?*

A beautiful analogy to understand the path to answer this question, is the English novelist Edwin Abbott's 1884 novella, *Flatland: A Romance of Many Dimensions*. The story revolves around a two-dimensional world called Flatland, wherein the characters are 2D geometric shapes, and the protagonist is A Square (In fact, "A Square" is the pseudonym under which Abbott wrote this book!). The inhabitants of Flatland are aware of the existence of a world called Lineland, where everybody is a line in one dimension, and everybody in Lineland is aware only of the world Pointland, wherein everybody is nothing but one point. Now, the Flatlanders have never wondered about the existence of a world of higher dimensions, having never encountered anything beyond 2D. For them, two is the highest one can ever go. Just like for the Linelanders, this magical limit is 1. But then one day, a Sphere appears from Spaceland to educate the Square about the 3D world. But the Square is not able to comprehend anything, having never stepped out of his frame-of-reference: the 2D world. However, when the Sphere transports Square to Spaceland, then by distancing him (on an interesting sidenote, men are polygons while women are lines in Flatland) from the lower 2D world to an outside 3D existence, the Square realises the truth of what the Sphere had to say: that Flatland was, in reality, enclosed within the container of Spaceland. However, when

the Square reports back to Flatland, nobody pays heed to his claims of the existence of a higher world, since all Flatlanders are constrained by the two dimensions of their world to understand what he's talking about. And well, he is imprisoned on accounts of spreading false and foolish fallacies.

In essence, what Flatland teaches us is that many times, to be able to study a system which is at an equivalent footing, we cannot do so if we are also at the same footing as the system being analysed. We must move to a higher level, and thus distance ourselves from the system-to-be-analysed, after which we *might* be able to study it accurately and objectively. If one applies this principle to the highest class of sophistication which consists of both UTMs and the human brain, then a strong claim would be to suggest that one could study and predict the human brain, and thus debunk the "myth" of free will, only through a computation (read: method) which lies in a class of sophistication above the brain and the UTM. And as long as we stick to the conventional model of computation, we will never be able to break out of this illusion.

Therefore, we seem to have proved that Free Will is just a black box concept, since it is impossible for us to predict the human brain's behaviour given our current computational model. Does this idea seem to suggest that the brain is then, deterministic? None of the above claims seem to contradict with this new idea. The human brain and UTMs are, after all, expressions of same amount of complexity. If a UTM can be expressed through deterministic rules, why can't the brain? The catch lies in, of course, the fact that we know the deterministic rules by which the UTM runs, since we created it, but we don't understand the rules governing our brains, since it was nature and evolution that created it! The reader might be confused by the two claims that:

1. The Brain is deterministic.

2. The Brain cannot be predicted.

But, on a closer look, one would realise that determinism doesn't necessarily imply prediction. If simply knowing the rules of a system were sufficient to predict its behaviour, then we should be able to predict the output of Conway's Game of Life! (As established in Section 2.2, we cannot.) For the purpose of closure, take the example of the famous Halting Problem in theoretical computer science. The halting problem is the problem of determining whether there exists an algorithm which, given any arbitrary computer program and an instance of this program, outputs whether the program would halt or not. This problem is deemed unsolvable by a Turing Machine, thus, we can never predict if the input problem will halt or not, despite everything (from the rules of the TM to the definition of the program and the instance) being deterministic.

Newcomb's Paradox Suggested by William Newcomb, of the University of California, this paradox tells the reader to consider the following. Say there is a supercomputer X, which is aware of all the deterministic physical laws of the Universe, which given all the current conditions, predicts the future. Now, a human player A asks X to pop the prediction for the output of a coin toss, which A would be executing immediately after popping up of the prediction. Now consider whichever prediction X pops: (a) Heads (b) Tails (c) the coin stands on its sides (d) the Player doesn't flip the coin, etc., A defies this prediction after hearing it. [If the prediction is a, b, or c, then the player doesn't flip the coin, and if it is d, then the player flips it anyway.] Why was the prediction defied, despite X being fully capable and aware of all the deterministic laws of the universe?

Franz Kiekeben, of the Ohio State University, suggests a way around this paradox. The only plausible solution being that determinism does not imply prediction. And why? Because during the popping up of the prediction, new information was added to the supercomputer-human system, which was not accounted for at the initial state which was inputted in the supercomputer for its calculation.

One could feel tempted to further extend Kiekeben's argument, and claim that this paradox of new information within the system can be resolved, if the entity predicting the coin toss is sitting outside the system, much like what was suggested from the lessons of Flatland and the idea of a computational class that lies higher above than the class containing the human and the supercomputer[24].

3.3 Uniqueness of Human Intelligence: The Travails of Triviality

In his widely popular paper titled *Computing Machinery and Intelligence*, Alan Turing considered the question of "can machines think?". However, owing to the vague definition and connotations of the word "think", he replaced this question by another question of "can machines pass the Turing Test?"[25] The Turing Test was essentially to examine whether a computer program could fool an isolated investigator, to believe that it was human, by imitating human-like behaviour, via a series of exchanges of questions and answers between the investigator and the computer. If the computer passed the test, it would be considered *intelligent*, in some sense of the word.

Its practical utility brushed aside, the Turing Test has been criticised for having used the idea of human-like behaviour as being the only form of intelligent behaviour. It considers humans to be in a privileged and unique position, without any particular reason provided for this assumption. However, if one were to base intelligence on the ability to understand complexity, and to solve problems of sophistication, then invoking the Principle of Computational Equivalence would cause both human intelligence (the brain) and machine intelligence (TM) to fall in the same category of equivalent intelligence. As a matter of fact, one could even include computations like the weather and heredity to be as intelligent as humans. Coming to think of it, most natural systems that we observe seem to fall into this category of high sophistication, which would imply that the level of human intelligence is a work of mere triviality, and nothing unique to our species.

One might argue against this, by iterating and reiterating the enormous contributions (both good, and bad) of mankind to the world. Which is why we need to re-evaluate our idea about intelligence. At the end of the day, claiming that the weather is as intelligent as us would to the least, increase our respect for nature's other creations and remind us of the other beautiful and complicated things of natural occurrence, and perhaps even reduce the root cause of our vanity and false sense of superiority as a species.

4 Computational Pragmatism

In the light of computational and biological complexity, and having studied the equivalence of classes, can we still hope to describe and explain some of the biological systems using our current level of computational technology, and within the limits of intractability imposed by theoretical computer science, or not? This section addresses this important question, to reveal which scientifically relevant questions need due consideration in the face of this argument.

4.1 Algorithmic Evolution

Daniel Dennett, an American Philosopher, spends a lot of time in his popular book *Darwin's Dangerous Idea* on describing evolution as an algorithmic process. He claims Darwin's theory of natural selection to be like an algorithm, and he goes on to define an algorithm on three characteristics[26]:

1. **Substrate Neutrality:** The procedure must be independent of the material on which it is executed, and its power lies in its logical structure.
2. **Underlying Mindlessness:** Each of the fundamental steps that build the procedure must be utterly simple.
3. **Guaranteed Results**: It must be a foolproof recipe.

What Dennett completely misses out in his discussion of evolution as an algorithm, however, is two of the most important properties of an algorithm as seen by computer scientists: the proof of correctness and the time complexity. Dennett claims that the outcome of this algorithm could be the creation of "complex structures" that arise. However, what is the means of describing the correctness of the working of this algorithm? How do you even talk about whether it is giving out any "guaranteed results"? As seen in Section 2.2, evolution and heredity are more like computations of high sophistication, with potential algorithms of very high complexity. Which means it is unlikely we would ever be able to comment on what exactly this computation is computing, in all of its entirety. Darwin's theory of evolution gives a macro perspective on this computation, hinting at the prospective objective of this computation to be the acquisition of levels of higher fitness. Advancements in the field of genetic algorithms is looking to model small modules of the idea of evolution, looking at small sections of the phylogenetic tree, but to be able to look at the big picture which lies at the highest level of sophistication using an algorithm, appears to be a long shot. After all, we will greatly benefit from any such model if only we are able to make somewhat accurate predictions from it.

4.2 Quantum Computing: An Incomplete Answer to the Biggest Questions of Human Existence?

The reader might recall from the discussion in Section 1.2, that in the theory of computational complexity, computer scientists have more reasons to believe that P≠NP. Under the assumption of this inequality having maximum likelihood of being true, we know that there is no polynomial time algorithm for problems belonging to class NP-Hard. What are we to make of this result? That most instances of NP-Hard problems may be treated as essentially unsolvable, given that we have finite number of resources and time in practical studies, experiments and simulations. However, if one were to change the model of computation itself, from a deterministic one (DFA) to a non-deterministic one (NFA), the problems in NP-Hard could then potentially be solved in polynomial time by that new model.

This is where Quantum Computers (QCs) come to our rescue. While traditional computers are based on DFAs, a quantum computer works on NFAs. It exploits the quantum mechanical nature and the idea of a spin of an electron, to take probabilistic decisions to reach from one state to another. This "non-determinism", in a certain sense, allows quantum computers to solve certain problems in quick time. How? Consider there are 300 electrons in the quantum computer system, and each electron can be in two states (up spin or down spin). This means that this computer can achieve as many as 2^{300} states, which is almost same as the number of atoms in the visible Universe! This possibility of strong vasts amount of information in fewer number of units is what grants quantum computers their power (mathematically, an exponential amount of speedup!) So can quantum computers solve every NP-hard problem efficiently?

You might recollect from past discussions that if even a single problem from the class NP-hard is solved efficiently, it would solve all problems in class NP in poly time. Quantum computers have been shown to execute the NP-Complete problem of integer factoring (which is what cryptographic protocol RSA is based on) in poly time. However, unfortunately, most computer scientists have been able to find few quantum algorithms for other problems in that class[27]. We cannot even comment if we can have a quantum algorithm for all NP-Complete problems, which, considering that we haven't proven that no poly time algorithms exist for NP-Complete problems, seems believable.

In the August of 2012, the quantum computer D-Wave was able to find the lowest configuration of amino acids and interactions, and thus "solve" the protein folding problem[28]. Although the algorithm didn't work perfectly, since out of 10,000 runs, it gave the right solution only 13 times. Still, this recent breakthrough gives hope to all fields that deal with complex systems, particularly biology, that there could after all be a way to create and exploit a piece of computation which is of sophistication higher than a standard deterministic UTM, and more sophisticated than even the human brain or the natural wonder of evolution. They could potentially shed some light on the eternal questions of existentialism and the meaning of being and life. Where some scientists might be conjecturing the possibility that quantum computers cannot solve all NP-Complete problems, one must remember that the doors of revision and correction are never shut in the scientific world. And neither is the window of imagination. Some physicists are already considering the possibility of a computation even more sophisticated than a QC: the Ubër Computer[27]. Based on speculative physical processes born from exotic physics, like the idea of time travel, Ubër Computers are being speculated to be powerful enough to solve problems as complicated as PSPACE, which are higher in complexity than even the class of NP-Complete, including problems such as Chess!

5 In Conclusion

"The world is stranger than we think," is in essence, what the French philosopher Sartre meant as one of the strong points of his ideas of existentialism. As humans advance with scaling acceleration in different fields of science and technology, we keep hitting bumps at more and more problems. Just like the further away a candle shines, the larger we realise is the region of unknown darkness. Sometimes when we reach such barriers, like unexplained phenomena, we try and modify our current scientific theories to fit the observed phenomena. Other times, we disband old theories and come up with new ones. In our haste of reaching a conclusion, and wiggling out results that make human life better, what we often ignore is the fact that we too have limits to our abilities. The world, which we had come to know so well, could suddenly appear more stranger than ever.

This paper has tried to expose such inevitable gaps of knowledge by establishing principles of correspondence between classes of computational complexity, and classes of biological complexity. Biological systems and life, exist in the form of a complexity pyramid, where jumping from one lower class to a higher class could seem implausible within our current models of computation. Also, it strengthens the notions of when problems can be reduced to simpler ones, and when problems are just irreducible. It motivates biologists and computer scientists to join hands on founding a new paradigm in which it would be possible to study and predict the behaviour of systems of high sophistication, like the brain, weather, evolution, etc. This would require for us to be able to create a computational model which resides higher than the current UTM model, and even higher than us, in the pyramid of the classes of sophistication/complexity.

Besides the ideas of computational pragmatism in treatment of biological systems, this paper also sheds some light on the nature of human thought, intelligence and brain. That the black box of free will is a mere illusion which can be explained by the principles of computational equivalence and irreducibility. And that determinism of rules may not necessarily imply prediction of a system governed by those rules. The central position that humans appear to hold among all the creations of nature, seems to turn small and trivial when one realises that there is nothing unique about the computational complexity of the human brain. And that we are blips in this cosmos of complexity, yet unique in our endeavours to dream of creating systems of complexities unfathomably larger than our own.

References

[1] Definition of signature of life taken from the discussion with Prof. Aditya Mittal, IIT Delhi.

[2] Erwin Schrödinger (1944). What is Life – the Physical Aspect of the Living Cell. Cambridge University Press.

[3] Crick (1966). Of Molecules and Man. Seattle, USA, University of Washington Press.

[4] Fulvio Mazzochhi (2008). Complexity in biology. European Molecular Biology Organization, Vol 1, No 1.

[5] D. Searls (1998). Grand Challenges in Computational Biology. Computational Methods in Molecular Biology, Elsevier Science.

[6] Wikipedia page: http://en.wikipedia.org/wiki/Protein_structure_prediction#Historical_perspective

[7] Costanzo et al. (2010). Science, 327: 425.

[8] Azevedo et al. (2009). Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. J Comp Neurol., 513(5):532-41.

[9] CNET page: http://www.cnet.com/news/human-brain-has-more-switches-than-all-computers-on-earth/

[10] Gandy (1980). Barwise 1980:123

[11] Stephen Wolfram (2002). A New Kind of Science. Champaign, IL, Wolfram Media, Page 717.

[12] Pray (2008). DNA replication and causes of mutation. Nature Education 1(1):214.

[13] International Human Genome Sequencing Consortium (2004). Finishing the euchromatic sequence of the human genome. Nature 431 (7011): 931–45.

[14] Claverie JM (2005). Fewer genes, more noncoding RNA. Science 309 (5740): 1529–30.

[15] Hector Zenil (2012). Irreducibility and Computational Equivalence: 10 Years After Wolfram's A New Kind of Science. Springer Science & Business Media, Page 188.

[16] Conway et al. (2001 2004). Winning Ways for your Mathematical Plays, (2nd ed.). A K Peters Ltd.

[17] Stephen Wolfram (2002). A New Kind of Science. Champaign, IL, Wolfram Media, Page 737.

[18] YouTube video featuring John Conway: https://www.youtube.com/watch?v=R9Plq-D1gEk

[19] Wikipedia page: http://en.wikipedia.org/wiki/Mathematical_and_theoretical_biology

[20] Kirschner (2005). Cell, Vol. 121, 503-504.

[21] Elements of Systems Biology as taken from discussions with Prof. James Gomes, IIT Delhi.

[22] Wooley et al. (2005). Catalyzing Inquiry at the Interface of Computing and Biology. National Academies Press: 429-435.

[23] Berger et al. (1998). Protein Folding in the Hydrophobic-Hydrophilic (HP) Model is NP-Complete. RECOMB: 30-40.

[24] Kiekeben's page: http://www.franzkiekeben.com/predict.html

[25] Turing (1950). Computing Machinery and Intelligence. Mind LIX (236): 433–460.

[26] Dennett (1995). Darwin's Dangerous Idea. Penguin Books.

[27] Aaronson (2008). The Limits of Quantum. Scientific American, Inc.

[28] Nature blog: http://blogs.nature.com/news/2012/08/d-wave-quantum-computer-solves-protein-folding-problem.html