Stochastic Finite Automata for the Translation of DNA to Protein

Tsau-Young Lin[1] and Asmi H. Shah[1,2,3]
[1]San Jose State University (SJSU), USA
[2]Masssachussetts General Hospital (MGH), Harvard Medical School, USA
[3]Lets Nurture Inc., India
tylin@cs.sjsu.edu and asmi.capri@gmail.com

***Abstract*—The use of Statistical Finite Automata (SFA) has been explored in the field of understanding the DNA sequences; many focus on local patterns, namely partial representations of DNA sequences. In this paper, we focus on global and complete representations to understand the patterns in whole DNA sequences. Obviously, DNA sequences are not random. Based on Kolmogorov complexity theory, there should be some simple Turing machines that write out such sequences; here simple means the complexity of the Turing machine is simpler than the data. The primary goal of this paper is to approximate such simple Turing machines by SFA. We use SFA, via ALERGIA algorithm, to capture and analyze the translation process (DNA to protein) based on amino acids' chemical property *viz*., polarity. This, in turn, enables the understanding of interspecies DNA comparisons and the creation of phylogeny – the 'tree of life'.**

Keywords—DNA, pattern, stochastic finite automata, machine learning, proteins, bigdata

I. INTRODUCTION

Essentially we - humans or, as a matter of fact, any living being known to the nature are made up of proteins. Proteins not only fabricate the physical body of the living creature, but also control almost all the bodily chemical processes carried out in the cells [1]. Some 37 trillion cells make up an average human body. The code by which all such cells take birth, carry out their functions, and die out when their time is done, is carried in a complete set of chromosomes (made up of DNA) in each such cell. Proteins are made up of chains of amino acids lying around in the cells' nuclei. These amino acid chains i.e., proteins, are synthesized after reading the instructions provided by the DNA sequence. DNA (DeoxyriboNucleic Acid) is a double helix three dimensional binding form of four nucleotides (called bases) represented by four alphabets i.e., A, C, G, and T. Human genome has 23 chromosomes which are made up of DNA. These strings of DNA contain around 20 to 25 thousand genes and non-coding regions of DNA; in total of some 3000 mega base pairs (Mb). The translation process of DNA to proteins is explained by the phenomena of central dogma of molecular biology [2].

DNA is nonlinear and complex. It has unique mechanism to store information. It is nature's most amazing example for the level of compression it offers. Researchers are still trying to decode and understand the behavior of the codes significantly packed in these microscopic chromosomes. If we find some success there, we can replicate these ideas into the artificial intelligent (AI) systems, which will take AI into a different level of sophistication. But one thing is clear that DNA sequence is not random. Considering the whole phylogenic tree as well as the resemblance between the genotype of species, phenotypes of the siblings and their parents and grandparents, etc. tell a lot about the common patterns the DNA and genes would carry and their correlations. Also, they say mutations themselves have some pattern as well. Moreover, apart from genes, there is a lot of non-coding DNA, which consists of some essential regulatory elements such as promoters and enhancers. For example, promoters at specific locations on the DNA signal the presence of next gene but are not part of any gene. These elements also do have patterns. A classic example is a TATA box [3], a promoter which is a mere repetition of nucleotides 'T' and 'A' (as TAATAATATA…) Tandem repeats in genes have roots in some genetic diseases like Huntington's disease (repetition of 'CAG' as CAGCAGCAG… with more than 40 repeats) [4]. The repeats' number can be in tens, hundreds, or thousands as well.

Thus, for so many reasons it is necessary to have mechanisms to identify and predict the types of patterns our DNA holds. The ability of predicting such genetic disorder patterns in DNA can lead us to tremendous opportunities in the betterment of the living animal kingdom including us in not-so-far-away-future. This could only happen when all the interdisciplinary fields come together and their efforts take place hand-in-hand. For example, relevant information are available in the fields of clinical data, gene expression data, DNA sequence databases, data about the interactions of genes with the environment, genotype-to-phenotype data, etc. The greatest challenge we have here is of humongous DNA sequence datasets itself, where some 183 eukaryotes (fungi, plants, and animals) are completely sequenced. The human genome differs by 0.1% from human to human. This 0.1% makes us look so different from each other considering our diversity! Now, consider the current world population of 7 billion and let say, we are all sequenced; then, the collective DNA data would result to be around 21 Exabytes, just for us humans as of now! Thus, considering the heterogeneous data resources and efforts done here and there in machine learning approaches such as *deep learning* with the evolution in multimedia, video, and graphics as well, we have to move towards the bigdata and deepdata solutions for understanding our DNA code better.

- This work is based on AHS's master's thesis carried out at SJSU. No update was carried out during her PhD research work. She re-visited the study and started working on it again immediately after her doctoral graduation.

II. Global and Local Patterns and Turing Machines

Kolmogorov introduced the concept of random in terms of his complexity, by simple twisting we proposed the following concept of patterns: a sequence is said to have a pattern, if the sequence is not random. An obvious consequence is the pattern can be written out by a simple Turing Machine, where simple means the complexity of this Turing machine is simpler than the input data [5] [6] [29]. A good example is the prime number sequence; the algorithm is well known, however, it is not satisfactory to human being; different types of patterns have been feverishly searched by mathematicians. For symbolic patterns, we need to search for simpler primitives – we choose DFA. Let us examine the approaches in "standard" mathematics.

A. Local and Global Patterns in numerical sequences?

The idea of this section is taken mostly from [6]. Given a sequence

(1) "1, 3, 5, 7, ...", what would be the next number?

Almost a uniform answer will be 9. Why? Implicitly, everyone consciously or unconsciously knows that the sequence is generated by ascending odd numbers (global pattern).

(2) $f(n) = 2n-1$, for any n.

Next, let us examine some "harder" sequence.

(3) "1, 3, 5, 7, 15, 41...", what would be the general term?

Observe that we are capturing the global patterns of the whole sequence: We are not looking at local patterns that are valid only for first few steps; for example polynomial (length = 3).

(4) $f(n) = 2n-1$, for $n \leq 3$ are *local* patterns for both sequences (1) and (2), but is not a *global* pattern of sequence (3).

B. Global Patterns in Symbolic sequences?

Observe that a pattern can be represented by any function, including an unknown function. The reason that all the patterns found above are polynomials is due to the fact that there is a hidden mathematical theorem, Stone Weistrass Approximation Theorem [9], that says, every continuous function defined on a compact domain (bounded and closed) can be approximated by polynomials. What should be the Symbolic Analogy? We have conjectured that analogous to polynomials are finite automata and there are some suitable version of Stone Weistrass Approximation Theorm in the world of symbolic sequences. Though we have not formulated such a theorem precisely yet, our group has been working on exploring the evidences. We have found some successful applications in intrusion detection of some malware [7], identification of (content independent) writing patterns of authors [8], etc. Past several years, we have been trying to understand the patterns of DNA sequences by use of this same finite automata theory, which has been proved to be working for the two challenges mentioned above. Though, DNA problem is more complex and deeper from the biological point of view as well as the machine learning point of view, we also have found positive evidences in this work. We conjecture that automata theory is one of the best mechanisms to unlock the patterns hidden in DNA sequences. In this work, the approximation theory of Granular Computing [31] [32] was one of the key factors responsible for the success of its applications.

The general theme of approximation is: In a given universal set U (of discourse), is there a family β, termed "known knowledge" of appropriate subsets of U so that an unknown concept (some type of subset of U) can be approximated by "(known) knowledge" ? In the theory of topological spaces, β is the topology (the family of open sets) or the (topological) neighborhood systems [34, Chapter 1 and Exercise B]. In the theory of neighborhood systems (generalized topological spaces) β is the neighborhood system [31] [32]. And in rough set theory β is a partition. In automata theory, it goes as follows: Let U be the universe that consists of all sentences from a fixed set of alphabets Σ (that is, $U = \Sigma^*$ as in [30]). In such a view, a Turing machine is a recursive enumerable subset and a finite automaton a regular subset. So the main theme becomes: *Could each Turing machine be approximated by a collection of finite automata?* Our research tells "yes" *stochastically*. From a standard text e. g., [30], one sees that a Turing machine can easily be approximated by a set of finite automata from above, namely each sentence accepted by a Turing machine can also be accepted a finite automaton; we will use stochastic finite automata, since such automata allow probabilistic uncertainty, and thus, we often can get more flexible approximations.

III. Method of modeling the DNA to protein translation process

A lot of ligands, enzymes, and other cell machineries are needed to be around in the cell to create the convenient environment for promoting transcription (DNA to mRNA) or for blocking the transcription process. After transcription, these orchestrated factors give the instruction to the necessary cell units to get ready for the translation process (mRNA to proteins). Some gene unwinds and one strand of DNA behaves as the template and gets transcribed into mRNA and further gets translated into protein. This is the basic process followed in molecular biology of the cell; to describe it in more detail is beyond the scope of this paper (please refer to the central dogma of molecular biology [2]).

A. DNA sequence representation with respect to amino acids

DNA consists of nucleotides A (Adenine), C (Cytosine), G (Guanine), and T (Thymine). mRNA consists of nucleotides A, C, G, and U (Uracil). The simple transcription process reads the template strand of DNA and synthesizes mRNA with exactly the same sequence as DNA, only difference is that Thymine is replaced with Uracil. Once mRNA is created, the protein synthesis starts. With help of other cytoplasmic RNAs, ribosomes start reading the mRNA sequence three base pairs at a time, which is called a codon. For each codon, there is an

amino acid mapped in the genetic code shown in fig. 1 [10]. Protein synthesis always starts with methionine amino acid, which is 'AUG' for its codon. From there on the amino acids are linked with each other with polypeptide bonds until the stop codon is encountered (UAA, UAG, or UGA).

Each such codon, standing for a unique amino acid, is a permutation of any 3 nucleotides out of the available 4. This makes a total of 64 codons, which code for 20 amino acids, where more than one codon can represent the same amino acid; refer the genetic code. Thus, taking the amino acids into consideration, we can have a base to analyze the coding gene DNA sequences.

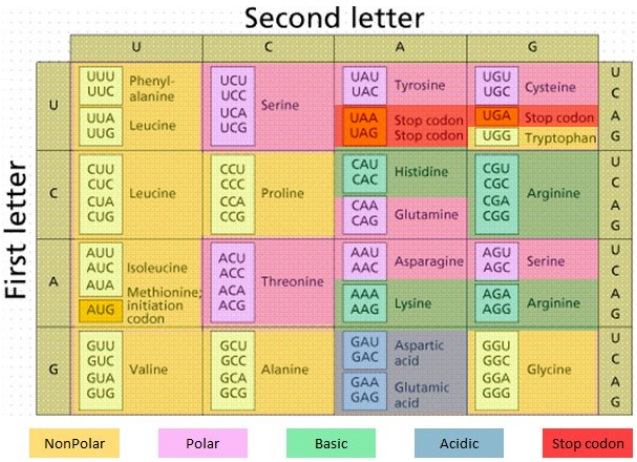

Figure 1. Genetic code - mapping of codons to respective amino acids. Different color coding shows the classification of the amino acid used here

Now, these 20 amino acids can be categorized further depending upon their chemical property – polarity. Each amino acid has an amino group and a side chain which varies in polarity, depending on its structure. Some are hydrophobic being non-polar and the rest are hydrophilic being polar, which again can be subcategorized in neutral, acidic (being negatively charged) and basic (being positively charged), depending on their acidic-basic behavior. It has been proved that during the translation process, replacement of one amino acid by another can result in silent replacement if they fall in the same category due to their similar chemical and structural properties [11] [12]. This indicates that 'intra' amino acid group replacement results in no major harm and the protein still results in the supposed one with no major dysfunction in its properties.

Hence, we have four main amino acid groups and the mapping of the amino acids with their codons is shown below. We enumerate the groups from 0 to 3 for the amino acid groups and -1 for the stop codons (please refer the genetic code in fig. 1).

- **0 – NonPolar:**
  Glycine, Alanine, Valine, Leucine, Isoleucine, Proline, Methionine, Phenylalanine, Tryptophan
- **1 – Polar (neutral):**
  Serine, Threonine, Cysteine, Asparagine, Glutamine, Tyrosine
- **2 – Acidic (polar acidic):**
  Aspartic acid, Glutamic acid
- **3 – Basic (polar basic):**
  Lysine, Arginine, Histidine
- **-1 – Stop codons**

With these codes we can think of DNA as a regular expression of numbers 0, 1, 2, 3 and -1 as the signal of the end of it with encountered stop codon on mRNA.

The phenomena of probable genetic mutation have its types too. Mutations are possible in the cellular process itself, which may result in a silent mutant protein (due to the intra amino acid replacements), totally deviated (missense), or senseless protein (nonsense) or no protein at all [11]. For example, in a DNA sequence we encountered 'GGU' (which is Glycine falling under the group of nonpolar amino acid – 0) as a codon instead of 'GCU' (which is a different amino acid – Alanine falling under the same group – 0) due to mutation where 'G' got replaced by the base 'C', then it would translate into and represent the same amino acid group even though they code for different amino acids.

B. Deterministic finite automata for DNA representation

To represent and understand the coding DNA sequences translated into proteins, their patterns, and in order to tackle the silent point mutations, we need some fault tolerance acceptance probability. Regular expressions can compress the repetition and represent them in lesser space [13], and these regular expressions can be studied well by use of Stochastic Finite Automata (SFA) with its state merging techniques described in [14]. Stochastic deterministic finite automata can not only learn the sequence pattern, but also have the capability of fault tolerance, which is integrated using the Alergia Algorithm [14] with acceptable confidence probability. For more details on the application of alergia algorithm on DNA sequences in the form of text as well as the derivations of Alergia, please refer to [14], [7], and [8].

The procedural theory of applying the Alergia algorithm can be explained with an example of a DNA "Homo sapiens nebulin (NEB), transcript variant 1, mRNA | NM_001164507.1 (NCBI DNA database reference number)". The DNA is taken and parsed by the designed algorithm to find the training set strings, which actually would code for proteins. Each string starting from 0 (mapped from AUG) and ending by -1 (mapped from the stop codons) as described. Then, a prefix tree acceptor is created using this training set. The PTA generated from the training set by itself is going to be too rigid. To see some patterns we will need to generalize the automata to "learn" to accept some probable little deviation in the test strings, for example, to compensate for the mutations happening intra-species. This can be achieved by Alergia algorithm, a state merging method discussed in detail in [15], which at every node of PTA evaluates the probability of the transitions from that node. The equivalent nodes are merged and the transitions through the previous states are adjusted as well. If the nodes are not different, the destination nodes are approached and checked to see the compatibility. If they are compatible then we can merge them. In the end, the tree is compressed by merging the states which are compatible and the properties of

each node are recalculated and changed such as their parent node, child nodes, and frequencies of transitions.

C. Algorithm

The algorithm proposed here for modeling the DNA sequence pattern recognition and comparison by SFA with its state merging technique can be shown by the system flow chart shown in fig. 2.

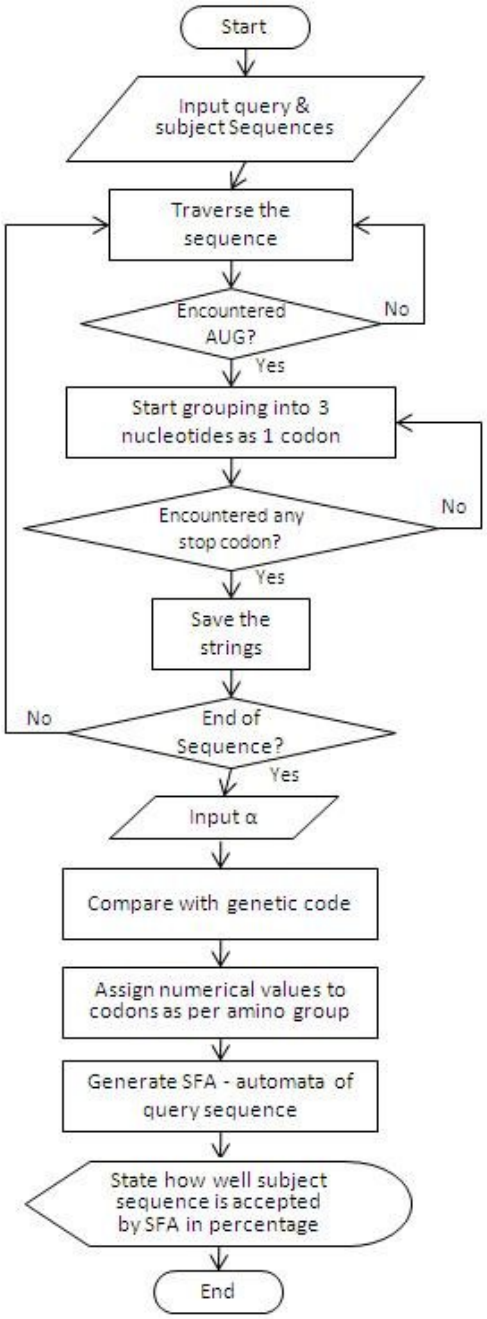

Figure 2. System flow chart of the SFA software

After creating a general PTA from the strings found in the input DNA sequence, a reasonable confidence level α is selected and applied to the PTA to merge some states and create an SFA. Nodes are checked for their equivalence depending on their probabilities. The equivalent ones are merged and the non-compatible ones are left unmerged. Once the SFA is created from the training DNA, the test DNA sequences from other species are checked against the SFA for cross comparisons with the training DNA, depending on the chosen confidence level α.

D. Complexity in dealing with DNA sequences

To understand this with an example, let us take a set of strings of DNA (very short and dummy sequences, as actual gene sequences are almost 100s or 1000s of characters long, but to understand the whole process clearly, we would take very small dummy strings) as given below as the training set, to create the automata:

Str1: AUG GCG AGA CAG CCA UGA
Str2: AUG CUC AGG GAU UAA
Str3: AUG GCC CAC UGA
Str4: AUG AAA UGC UGG UAG
Str5: AUG CGU ACG CAU ACC UAA
Str6: AUG GCC UGC ACG UAA
Str7: AUG UGG GAU UAG
Str8: AUG UUU AAG UGA

Mapping them to our amino acid groups as described:
Str1: 0 0 3 1 0 -1
Str2: 0 0 3 1 -1
Str3: 0 0 3 -1
Str4: 0 3 1 0 -1
Str5: 0 3 1 3 1 -1
Str6: 0 0 1 1 -1
Str7: 0 0 1 -1
Str8: 0 0 3 -1

Thus, the set S is as following:
S = {00310, 0031, 003, 0310, 03131, 0011, 001, 003}

TABLE I. FREQUENCY STATISTICS FOR PTA OF SET S

i	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
n_i	8	8	6	4	2	1	2	2	1	1	1	2	1
$f_i(\#)$	0	0	0	2	1	1	0	0	1	0	1	1	1
$f_i(0)$	8	6	0	0	1	0	0	1	0	1	0	0	0
$f_i(1)$	0	0	0	2	0	0	2	0	0	0	0	1	0
$f_i(3)$	0	2	4	0	0	0	0	1	0	0	0	0	0

Fig. 3 shows the PTA generated out of set *S* and table I shows the frequency statistics for set *S*. By applying the alergia algorithm with $\alpha = 0.7$ and checking the calculations for states' equvivalence, it comes out that nodes *5, 8, 10, & 12* are compatible as their frequencies are equivalent; this is shown in fig. 4. Further calculations and compatibility checks

are shown in fig. 5 and fig. 6. At each step of the merging process, the frequency statistics are calculated and updated.

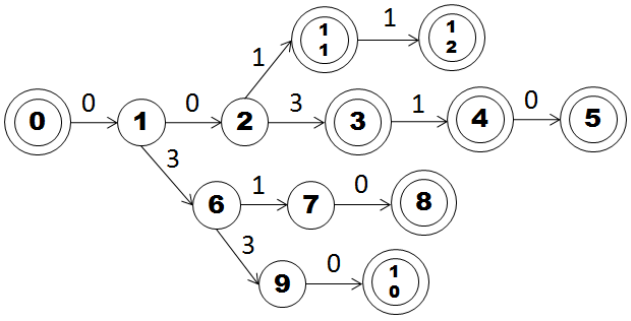

Figure 3. PTA of set *S*

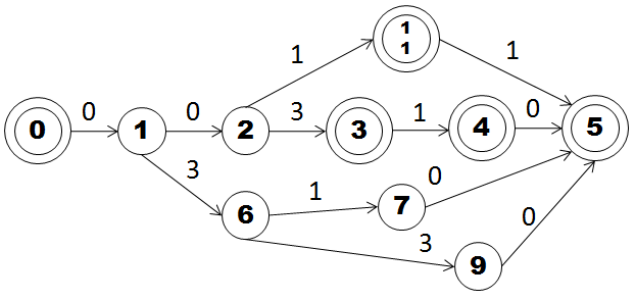

Figure 4. Merging states *5, 8, 10, & 12*

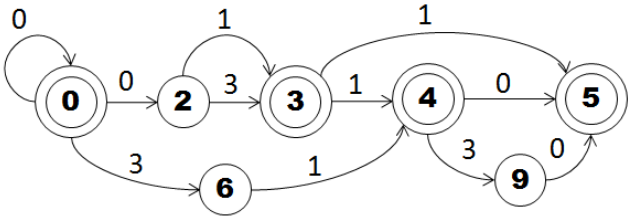

Figure 5. Merging states *0 & 1 and 3 & 11 and 4 & 7*

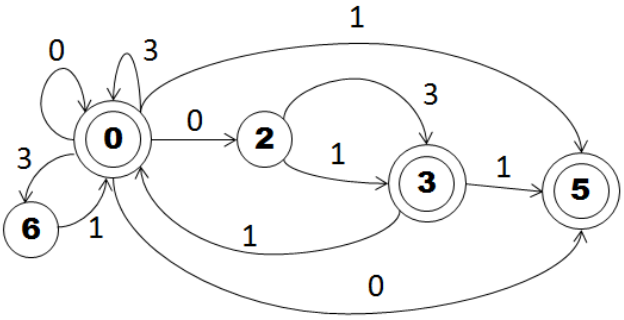

Figure 6. Merging states *0, 4, & 9*

As we can see in fig. 6 above, the automaton defines a language by use of strings in the set *S*, which happens to be a subset of the set represented by the SFA in fig. 6. Now, if we have a set

Q = {003131011, 0310033, 1111010, 100101, 2312100}

then strings *003131011*and 0310033 are accepted by the SFA shown. Thus, the confidence (acceptance) probability will be 0.40 or 40% as each string carries 20% equal probability. We apply the same concept here in the case of actual DNA sequences where, set S contains the training DNA sequence strings and set Q contains the testing DNA sequence strings.

What really is to be taken point of here is that in our example we deal with very short sequences. To get a glimpse of actual DNA sequence size and their strings size, we look at the complete genome of mycobacterium canettii (a type of bacterium) which is in total 4525948 base pairs long, has 3861 genes meaning if we have it as our training DNA sequence then there would be 3861 strings which would create a PTA. Here, after assigning them the amino acid group numbers, the strings would consist of 507, 402, 385, 187, 675,… numbers (numbers coming from NCBI Nucleotide database) and thus, approximately 1,544,400 nodes (3861 strings * 400 numbers per string) in creating the first step PTA.

IV. Related Work

The analysis of DNA sequences to find patterns as repetition of nucleotides, to find important characteristic sites such as the TATA box, and to compare the DNA sequences with each other for evolutionary results have been done since decades. This has been one of the most important research areas in bioinformatics. But using machine learning approaches onto DNA sequences is not that common. It has not been explored to its fullest yet.

Each method of study has its own particular reason of invention and success; we will discuss some of them here.

E. 3D technique of DNA pattern matching

When the textual DNA sequences (the mere sequence of letters (nucleotides)) are aligned and matched against each other, they fail to answer many of the characteristic information about the species involved in the analysis. This happens because DNA is something more than just the sequence of characters; they are stored in the cells in form of double helical structure, where hydrogen bonds are responsible for their build. So the 3D model of the DNAs is considered and matched to solve the problem. The dimensions and the double helix structure of DNA sequences are compared [15].

This approach is towards the biological studies for the species behavioral characteristics as included in protein folding, whereas, our method works on the primary structure of DNA.

F. DNA pattern matching using FPGA

Here, FPGAs (Field Programmable Gate Arrays) are used to analyze DNA sequences and to match patterns between them. This approach uses the hardware directly, and so it happens to be efficient enough and expensive too. “The novel aspect of this approach is the technique of converting a matching problem into a boolean satisfiability problem and then to a circuit, exploiting the configurability of FPGAs” [16]. But, the limitation to the solution that this method can address is of the DNA size, that is, the string capacity that FPGAs can support.

In our approach, hardware is not used, the computational time increases in order of the size of DNA, which is very nominal as compared to the cost of hardware used here.

G. DNA pattern understanding using machine learning approaches

One of the preliminary efforts done towards understanding the DNA sequence using automata is discussed Burks and Farmer in [17]. Mainly, the bases A, C, G, and T are quantified to 1, 2, 3, and 4 respectively and the automaton is generated. Similar approaches are discussed in [18] and [19]. Fault tolerance and generalization is a biggest question while translating the DNA sequence into automata. Researchers have tried to use stochastic regular grammar and language to predict splice sites in DNA [20] and to model RNA [21]. Researchers have invested in understanding the exons and introns patterns with hidden markov models as well [22]. The other related studies to mention here are DNA sequence analysis based on neural networks [23], DNA analysis by optical pattern recognition [24], DNA image representation by cellular automata [25], etc.

The novelty in our approach is to apply the automata machine learning technique onto the whole genome of the organisms or rather any huge actual DNA sequences that code for proteins. Thus, already the data that we address is large scale as compared to the other approaches listed here. For example, concentrating on promoters or splice sites are just a few fractions of the whole DNA data. Moreover, we concentrate more on the biochemical aspect of the biological substance - DNA and try to characterize the DNA sequences in response to their ultimate protein functions and their salient properties. Moreover, the other approaches show their results for a partial DNA sequence or they work on only short original sequences in their study. While, in our study we have tested our software on whole chromosome sequences as well, for example, chromosome 3 of Arabidopsis thaliana – 23,459,830 base pairs.

V. RESULTS AND VALIDATION

Here, while converting the DNA sequence into the state machine, each string would code for a protein. Average length of human written sentences is 15-20 words. But, the DNA sequences in question are quite large. Each protein normally has 400-1000s of amino acids. The largest protein, *titin*, has around 27000 amino acids.

The software developed for this study asks the user to point to the directory folder, where DNA sequences, to be compared with its human counterpart, are sitting. All the files that are placed in that folder are taken into consideration and are checked against the SFA created out of the human DNA sequence. To vary the rigidity of the SFA and make it more general a range of numbers are tested for different values of α, where $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$.

A. Nebulin DNA sequences from various species compared

Nebulin is very important for muscle heath; it has been evolved in vertebrate skeletal muscles [26]. We took a human nebulin sequence to create the automata. Nebulin from the other species are extracted from NCBI's nucleotide database. The species considered in this run of experiment and their DNA sequence information is as following:

- Human - Homo sapiens nebulin (NEB), transcript variant 1, mRNA (locus: NM_001164507) which has 26202 base pairs -- HumanNEB
- HouseMouse - Mus musculus nebulin (Neb), mRNA (locus: NM_010889.1), length = 22489 bp -- MouseNEB
- Rat - Rattus norvegicus nebulin (Neb), transcript variant X21, mRNA (locus: XM_006224436.1), length = 21702 bp -- RatNEB
- Chimpanzee - Pan troglodytes nebulin (NEB), mRNA (locus: XM_515832.4), length = 20634 bp -- ChimpNEB
- DomesticCow - Bos taurus nebulin (NEB), mRNA (locus: XM_613028.6), length = 22869 bp -- CowNEB
- Zebrafish - Danio rerio nebulin (neb), transcript variant X1, mRNA (locus: XM_002663368.3), length = 19870 bp -- ZebrafishNEB
- RhesusMonkey - Macaca mulatta nebulin-like (LOC100430324), partial miscRNA (locus: XR_091505.1), length = 13835 bp -- MonkeyNEB
- Pufferfish - Takifugu rubripes nebulin-like (LOC101064677), mRNA (locus: XM_003962214.1), length = 9585 bp -- PufferfishNEB

TABLE II. SPECIES NEBULIN COMPARISON WITH SFA CREATED FROM HUMAN-NEB ALONG WITH COMPARISON TO NCBI'S BLAST RESULTS

α	Chimp-NEB	Cow-NEB	Mouse-NEB	Rat-NEB
0.2	97.8	92.6	89.5	84.2
0.5	92.7	85.6	73.3	70.4
0.7	65.3	56.2	65.2	42.5
0.9	50.7	43.5	52.6	34.1
BLAST	93 o 100	90 o 98	87 o 98	84 o 95
owSeq	91%	87%	75%	69%

α	Zebrafish-NEB	Monkey-NEB	Pufferfish-NEB
0.2	42.6	78.1	70.3
0.5	26.3	69.9	43.8
0.7	16.4	55.9	20.4
0.9	05.0	41.5	19.8
BLAST	65 o 89	97 o 67	71 o 34
owSeq	10%	68%	42%

The first step PTA without applying the state merging method yet had some 11,759 total nodes from the training DNA strings set. The acceptance by the SFA or rather similarity score for each species created from Human-NEB is shown in table II. The row "BLAST" shows the results of alignment done between human and the respective species-NEB sequences by the BLAST (Basic Local Alignment Search Tool) algorithm [27] available at blast.ncbi.nlm.nih.gov. When

it comes to comparing the DNA or protein sequences, or to search homologous species or to map evolutionary relations, BLAST is a defacto standard for DNA sequence alignment. It is a heuristic approach where it finds a few short matches between two sequences and then starts to look for more similar alignments locally, which means it does not cover the entire sequence space. To match a 'query' and 'subject' sequence, it starts with the matching words of size seven to eleven bases. From there, it starts comparing the nearby sequence areas to find more matches. It only aligns locally, considers few gaps and substitutions. For example, for Pufferfish-NEB in "BLAST" has the value as '71 o 34' (short for - 71% over 34%), which means the following. 71% similarity is seen between HumanNEB and PufferfishNEB over just 34% of the humanNEB sequence. The rest of 66% of the sequence does not match at all, even with the gaps and substitutions, and is not considered while aligning the sequences. The column "owSeq" (short for - over whole sequence) takes the whole sequence into consideration and not only the local alignment and gives the score of similarity over the whole sequence derived from BLAST.

VI. Disscussion and Outlook

Here, if we analyze the results in table II, we can say that when $\alpha <= 0.7$, the SFA is too rigid with lesser merges of the states. More rigid the SFA, lesser the number of strings accepted by it. With decrease in α at 0.5, we can see comparable similarities to the results of column "owSeq". From this example and the other test runs done with this program, we can say that for DNA sequences,

$$0.4 < \alpha <= 0.55$$

gives comparable satiable results. Moreover, as discussed already, we only take care of substitution mutations and not the insertion and deletion (with those kind of mutations there is a frame shift in the reading frame in coding DNA sequence and thus, translated into completely different amino acid chain). Even, for this factor, we can loosen up the automata a bit to handle little repetitions and loops occurring in the DNA sequences (though not so frequently).

The runtime of the program is linear – *O(n),* even with large datasets. We have tried to compare whole chromosomes of the species against each other, where for example, Arabidopsis chromosome 3 is of 23,459,830 base pairs. But the program sticks to its linear runtime. If we consider the bigdata that we would be flooded with, when more and more organisms are sequences and more and more individual humans are sequenced to answer the questions like genetic disorders etc. then we should even try to optimize the algorithm here to reduce the runtime as well.

The same HumanNEB was compared against Arabidopsis thaliana (a model plant), *c. elegans* (a model worm), drosophila melanogaster (fruit fly), hookworm, lungworm, etc. For those species DNA, hardly few 10s of base pairs really matched locally out of 10,000s of base pairs. In that case, our SFA comparison did not really tell us much. Hence, one thing is for sure that the method, by which we generate the SFA here, has to change a little to accommodate the insertions and deletions of base pairs.

Furthermore, many of the protein behavioral patterns depend on the secondary and 3D structure of the DNA sequence [28], which the proposed approach cannot handle. Also, they can be characterized by their other properties like their shape and size as well. So, this can be taken care of in the future implications.

This approach started from a trivial observation that a high frequent sequence is compressible [5]; subsequently the idea was extended to numerical sequences [6]. Baliga and Lin have extended this approach to computer security [7]; the idea is similar, but distinct, to [31]. Lin and Zhang have extended the idea to text analysis, and built an authorship identification system [8]. Here, an SFA is based on the sequence of function words of writing sample of an author and then other writing samples are tested against this SFA. In the sentences, the content words such as nouns, pronouns, objects, etc. were thrown away and only the functional words such as articles and prepositions, etc. were kept (observe the patterns are content independent). Moreover, even complex sentences written by human are usually not longer than 20 to 25 words. Thus, the training set sentences consist of only few states. And thus, in thus study, the confidence probability α had a higher value where their approach was working, meaning, less mergers of the states where needed to actually find the pattern difference in writing done by different authors. But, in our case, the complex DNA samples coding for protein are much longer as compared to the human written sentences.

Thus, the simplicity Baliga, Zhang, and Lin saw in their applications of automata is not seen in addressing the DNA analysis question. The problem is far more complex. Nevertheless, the results we have got out of our study have been really convincing, though with a selection of lower value to α to compensate for the longer strings making the automata more rigid.

Acknowledgment

The authors would like to thank Nikhil Kalantri, Parnika Achrekar, Qun Yu, Yue Lu, and Ahmad Marzdasht Yazdankhah for their contributions towards improving the software in general. The results presented here are based on these accumulated improvements.

References

[1] Crick, F. (1970). Central dogma of molecular biology. Nature, 227(5258), 561-563.

[2] Dawkins, R. (2006). The selfish gene (No. 199). Oxford university press.

[3] Dynan, W. S. (1986). Promoters for housekeeping genes. Trends in Genetics,2, 196-197.

[4] Anderson, M. A., Tanzi, R. E., Watkins, P. C., Ottina, K., Wallace, M. R., Sakaguchi, A. Y., ... & Bonillah, E. (2004). A polymorphic DNA marker genetically linked to Huntington's disease. Landmarks in Medical Genetics: Classic Papers with Commentaries, 51, 153.

[5] Lin, T. Y. (1993). Rough Patterns in Data - Rough Sets and Foundation of Intrusion Detection Systems, Journal of Foundation of Computing and Decision Sciences, Vol.18, No. 3-4, 1993. 225-241.

[6] Lin, T. Y. (1999). Patterns in numerical data: practical approximations to Kolmogorov complexity. In New Directions in Rough Sets, Data Mining, and Granular-Soft Computing (pp. 509-513). Springer Berlin Heidelberg.

[7] Baliga, P., & Lin, T. Y. (2005). Kolmogorov complexity based automata modeling for intrusion detection. In Granular Computing, 2005 IEEE International Conference on (Vol. 2, pp. 387-392). IEEE.

[8] Lin, T. Y., & Zhang, S. (2009). An Automata Based Authorship Identification System. In New Frontiers in Applied Data Mining (pp. 134-142). Springer Berlin Heidelberg.

[9] De Branges, L. (1959). The Stone-Weierstrass theorem. Proceedings of the American Mathematical Society, 10(5), 822-824.

[10] Streisinger, G., Okada, Y., Emrich, J., Newton, J., Tsugita, A., Terzaghi, E., & Inouye, M. (1966). Frameshift mutations and the genetic code. In Cold Spring Harbor Symposia on Quantitative Biology (Vol. 31, pp. 77-84). Cold Spring Harbor Laboratory Press.

[11] Jones S. (2009). The Britannica guide to Genetics, Running Press.

[12] Agutter P., Wheatley D. (2007). About Life – Concepts in Modern Biology, Springer.

[13] Sekar, R., Bendre, M., Dhurjati, D., & Bollineni, P. (2001). A fast automaton-based method for detecting anomalous program behaviors. In Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on (pp. 144-155). IEEE.

[14] Carrasco, R. C., & Oncina, J. (1994). Learning stochastic regular grammars by means of a state merging method. In Grammatical Inference and Applications (pp. 139-152). Springer Berlin Heidelberg.

[15] Hérisson, J., Payen, G., & Gherbi, R. (2007). A 3D pattern matching algorithm for DNA sequences. Bioinformatics, 23(6), 680-686.

[16] Lipson, A., & Hazelhurst, S. (2001). DNA pattern matching using FPGAs. In Proceedings of the twelth annual symposium of the Pattern Recognition Association of South Africa (pp. 180-185).

[17] Burks, C., & Farmer, D. (1984). Towards modeling DNA sequences as automata. Physica D: Nonlinear Phenomena, 10(1), 157-167.

[18] Yeol, J. W., Barjis, I., & Ryu, Y. S. (2005). Modeling of system biology: from DNA to protein by automata networks. In Intelligent Sensing and Information Processing, 2005. Proceedings of 2005 International Conference on (pp. 523-528). IEEE.

[19] Coste, F., & Kerbellec, G. (2006). Learning automata on protein sequences. In JOBIM (pp. 199-210).

[20] HKashiwabara, A. Y., Vieira, D. C. G., Machado-Lima, A., & Durham, A. M. (2007). Splice site prediction using stochastic regular grammars. Genetics and Molecular Research, 6, 105-115.

[21] Sakakibara, Y., Brown, M., Underwood, R. C., Mian, I. S., & Haussler, D. (1994). Stochastic context-free grammars for modeling RNA. In System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on (Vol. 5, pp. 284-293). IEEE.

[22] Baldi, P., Brunak, S., Chauvin, Y., Engelbrecht, J., & Krogh, A. (1995). Periodic sequence patterns in human exons. In Ismb (pp. 30-38).

[23] Murdock, M., Cotter, N., & Gesteland, R. (1991, July). Neural network based pattern recognition for sequenced DNA autoradiograms. In Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on (Vol. 2, pp. 909-vol). IEEE.

[24] Gildner, M. D., Christens-Barry, W. A., Martin, J. C., & Hawk, J. F. (1988). DNA sequence analysis by optical pattern recognition. In 1988 Orlando Technical Symposium (pp. 238-245). International Society for Optics and Photonics.

[25] Xiao, X., Shao, S., Ding, Y., Huang, Z., Chen, X., & Chou, K. C. (2005). Using cellular automata to generate image representation for biological sequences. Amino Acids, 28(1), 29-35.

[26] Labeit, S., Ottenheijm, C. A., & Granzier, H. (2011). Nebulin, a major player in muscle health and disease. The FASEB Journal, 25(3), 822-829.

[27] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. Journal of molecular biology, 215(3), 403-410.

[28] Bošnački, D., ten Eikelder, H. M., Steijaert, M. N., & de Vink, E. P. (2008). Stochastic analysis of amino acid substitution in protein synthesis. In Computational Methods in Systems Biology (pp. 367-386). Springer Berlin Heidelberg.

[29] Li, M, Vitanyi. (1997). An introduction to Kolmogorv Complexity and its Applications, 2nd ed, Springer.

[30] Hopecroft, J. (1969). Formal Language and their relation to Automata, Addison-Wesley.

[31] Steven A. Hofmeyr, Stephanie Forrest, Anil Somayaji. ((1998). Intrusion Detection Using Sequences of System Calls. Journal of Computer Security 6(3): 151-180.

[32] Lin, T. Y. (2009). Granular Computing, Introduction to. Encyclopedia of Complexity and Systems Science: 4313-4317

[33] Lin, T. Y. (2009). Granular Computing: Practices, Theories, and Future Directions. Encyclopedia of Complexity and Systems Science: 4339-4355

[34] Kelley, J. (1955). General Topology, van Nostrand.