Algebraic Structures and Their Applications

Algebraic Structures and Their Applications Vol. 9 No. 1 (2022) pp 109-119.

Research Paper

AUTOMATA ON GENETIC STRUCTURE

MRIDUL DUTTA*, SANJOY KALITA, HELEN K. SAIKIA

Abstract. In this paper, the authors have represented the genetic structures in terms of automata. With the algebraic structure defined on the genetic code authors defined an automaton on those codons as $\Sigma = (C_G, P, A_M, F, G)$ where P is the set of the four bases A, C, G, U as a set of alphabets or inputs, C_G is the set of all 64 codons, obtained from the ordering of the elements of P, as the set of states, A_M is the set of the 20 amino acids as the set of outputs that produce during the process. F and G are transition function and output function respectively. Authors observed that $M(\Sigma) = (\{f_a : a \in P\}, \circ)$ defined on the automata Σ where $f_a(q) = F(q, a) = qa, \ \ q \in C_G, a \in P$ is a monoid called the syntactic monoid of Σ, with $f_a \circ f_b = f_{ba} \ \forall a, b \in P$. Studying the structure defined in terms of automata it is also observed that the algebraic structure $(M(C_G), \ +, \ \cdot)$ forms a Near-Ring with respect to the two operations $'+'$ and $'\cdot'$ where $M(C_G) = \{f | f : C_G \to C_G\}$.

DOI: 10.29252/as.2021.2468

MSC(2010): 92B05, 17D92, 68Q70, 20M32.

Keywords: Automata, Transition function, Monoid, Near-ring, Genetic code, Codons.

Received: 16 October 2020, Accepted: 08 September 2021.

*Corresponding author

1. Introduction

An automaton (in plural Automata) is an abstract self-operating machine that follows a predetermined sequence of operations automatically gives an output from an input. The input may be energy, information, materials etc. The system works without the intervention of man. Automata theory plays a major role in huge applied areas. The most significant areas include communication, transportation, health care, electronic banking, etc. Mainly finite automata are significant in many different areas, including Electrical Engineering, Linguistics, Computer Science, Philosophy, Biology, Mathematics, etc. In Computer science, automata are widely used in text processing, Compilers, Software and hardware design, network protocol, etc.

In finite automata; the term *finite* comes from the number of states and the number of inputs being finite; automation is from the deterministic structure or machine, i.e the change of state is completely governed by the input. Generally, it is called Deterministic Finite Automata (DFA) [10]. The Deterministic Finite Automata consists of (i) A finite set of states, (ii) A finite set of inputs (iii) A finite set of transitions. The states are completely determined by the prior state and the input and state of the machine changes after every executed instruction. The machine simply starts at an initial state and reaches the final state by changing from state to state based on the instruction and the prior state [10].

In Biology, it is used in mollusk and pine cones growth and pigmentation pattern analysis, DNA, RNA, Protein pattern examining and analysis, etc. The genetic material of living organisms are either DNA or RNA. DNA means Deoxyribonucleic acid and RNA means Ribonucleic acid. DNA is found in chromosomes, mitochondria, and chloroplasts. It behaves as the genetic material in most of the organisms which transfer genetic information from one generation to the next. DNA as an acidic material present in the nucleus was first recognized by Friedrich Meischer in 1868. In 1953, James Watson and Francis Crick described a world-famous Double Helix structure for DNA. Rosalind Franklin used the x-ray diffraction technique to capture the first high-quality images of the DNA molecule. Maurice Wilkins showed the pictures to J. Watson, who had been working with F. Crick on the structure of the DNA molecule. These pictures gave Watson and Crick huge information to propose a double-stranded, helical, complementary, anti-parallel model for DNA that gives the 1962 Nobel Prize in physiology or medicine [1]. Researcher George Gamov works on how DNA functions to make protein. Many Biologists like R. W. Holley, H. G. Khorana, M. W. Nirenberg worked on how genetic information is translated into proteins. They won the Nobel Prize(1968) in Physiology or Medicine jointly for their work. Authors J. D. Bashford, I. Tsohantjis, P. D. Jarvis (1998) worked on a super-symmetric model for the evolutions on genetic code, like a periodic table [5]. J. Balakrishnan (2002) studied on a symmetric scheme for amino acid codons [3]. F. Antoneli, L. Braggion, M. Forger, J. E. M. Hornos (2003) had studied the symmetries in the

genetic code [2]. Researchers like R. sanchez, E. Morgado, R. Grau, I. Barjis, J. W. Yeol, Y. S. Ryu etc. (2002-2007) studied Gene algebra from a genetic code algebra structure [4, 14]. Authors A. N. Dalini, E. Ekhi, Y. Sakakibara worked on Bio-molecular computing of finite-state automata, a new DNA implementation of finite state machines in 2006 [6]. R. Selvakumar, M. R. Muhammad, G. P. Devi (2013) had done their research on a computational model for the extraction of human erythropoietin [14]. Also, authors Y. Saeed, S. Nassem, etc. (2014) worked on DNA pattern analysis using Finite automata [10]. Most recently authors like T. Ali, C. K. Phukan, N. Gohain, A. Akhtar worked on Topology in genetic code algebra, Lattice structure, Distance Matrix of Genetic Code, Reducing Redundancy of codons through a total graph, etc. (2012-2020) [1, 7, 8]. DNA and RNA are polynucleotides. They are made up of smaller molecules called nucleotides. DNA is made of two polynucleotides and RNA is made of a single polynucleotide strand. DNA is a nucleic acid made of long chains of nucleotides. In DNA the four nitrogenous bases are Adenine(A), Cytosine(C), Guanine(G), and Thymine(T). In RNA, the four nitrogen bases are Adenine(A), Cytosine(C), Guanine(G), and Uracil(U). Again Nitrogenous bases are two types Pyramidines(T, C, U) and Purines (A, G). Adenine always base pairs with Thymine (or Uracil if RNA) and Cytosine always base pair with Guanine[7, 8]. The Gene theory is one of the basic principles of Biology. The main concept of this theory is that traits are passed from parents to offspring through gene transmission. Genes are located on chromosomes and consist of DNA. They are passed from parent to offspring through reproduction. Messenger RNA (mRNA) is a form of RNA that carries the instructions for making the protein from a gene and delivers it to the site of translation. A nucleic acid word (three-nucleotide sequence) is known as Codon. Transfer RNA(tRNA) single strands of RNA that temporally carries a specific amino acid on one end and has an anticodon i.e. complementary to an mRNA codon. Protein synthesis has two major tracks. The process of making RNA from DNA is known as transcription and the RNA directions are used to make a protein from amino acid is known as translation. The genetic code is a set of regulations defining how the four-letter code of DNA is translated into the 20 letter codes of amino acids which are the ingredient of proteins. The genetic code finds out the connections between the three-nucleotide codon on the mRNA and the inclusion of the correct amino acid into a protein.

2. Preliminaries

Mathematics is progressively accepted as a significant apparatus to study multiplex systems. It helps us to discuss, understand, and develop many fields in real life. Mainly all branches of mathematics are very useful in biology containing algebra, calculus, number theory, automata

theory, topology, etc. In this article, we try to attempt to study DNA, RNA, and Protein analysis by using the tools of mathematics and automata theory.

A deterministic finite automaton is a five tuple, $\Sigma = (Q, A, q_0, Q_n, F)$ where Q is the set of states, A is the input alphabet, $q_0 \in Q$ the start state, $Q_n \subset Q$ is the set of final states and $F : Q \times A \to Q$ transition function.

Two major elements in finite automata are states and inputs. Here the change of state is fully governed by the input and the current state. The state moves from one place to another place due to the input and reaches the final state. The term deterministic refers to the fact that on each input there is one and only one state to which the automaton can transit from its current state.

The abstract definition of automata is a quintuple as described below, An automata is a system of the type $\Sigma = (Q, A, B, F, G)$ where Q is a set of states, A is a set of inputs, B is a set of outputs, $F : Q \times A \to Q$ and $G : Q \times A \to B$ are functions usually known as state transition function and output function respectively[10].

An algebraic system $(N, +, \cdot)$ consisting of a set N and two binary operations "+" and "$\cdot$" is called a Near-rings if $(N, +)$ is a group (not necessarily abelian), $(N, \cdot)$ is a semi group and $(n + n')n'' = nn'' + n'n''$ holds for all $n, n', n'' \in N$[11].

We consider the set of finite sequences of elements of the set including the empty $\wedge$. In other words in our study of automata, we extend the input set A to the free monoid $\overline{A} = F_A$ with $\wedge$ as identity. We also extend F from $Q \times A$ to $Q \times \overline{A}$ by defining $q \in Q$, $a_1, a_2, a_3 \ldots a_n \in A$ such that $\overline{F}(q, \wedge) = q$, $\overline{F}(q, a_1) = F(q, a_1)$, $\overline{F}(q, a_1a_2) = F(F(q, a_1), a_2)$ and so on. In this way , we obtain a function $\overline{F} : Q \times \overline{A} \to Q$. Similarly, we can obtain $\overline{G} : Q \times \overline{A} \to B$. Let us consider the automata $\Sigma = (Q, A, B, F, G)$. For $\alpha \in \overline{A}$, let $f_\alpha : Q \to Q$ defined as $f_\alpha(q) = \overline{F}(q, \alpha)$ where $(\{f_\alpha : \alpha \in \overline{A}\}, \circ) = M(\Sigma)$ is a monoid called the syntactic monoid of Σ, $\forall \alpha, \beta \in \overline{A}$, we have $f_\alpha \circ f_\beta = f_{\beta\alpha}$[9].

Mathematically, RNA can be considered as a sequence of four letters A, G, C, U that represents the four bases. These four bases give us 64 codons, out of which, the codon AUG known as starting codon start the translation process and the three codons UAA, UAG and UGA are known as stop codons or nonsense codons and their role is to stop the biosynthesis process. In DNA(or RNA), 64 codons make up the genetic code, though there are only 20 amino acids. The genetic codes are the protein sequences which are the groups of these 20 amino acids namely Phenylalanine(F), Leucine(L), Isoleucine(I), Methionine(M), Valine(V), Serine(S), Proline(P), Threonine(T), Alanine(A), Tyrosine(Y), Histidine(H), Glutamine(Q), Asparagine(N), Lysine(K), Aspartic acid(D), Glutamic acid(E), Cysteine(C), Tryptophan(W), Arginine(R), Glycine(G) [8].

The ordering of the 64 codons implicitly gives an ordering of the four RNA(or DNA) bases. From these two orders of the base set $\{A, C, G, U\}$ and $\{U, G, C, A\}$ are obtained and further a sum operation is defined on these two ordered sets. The two sets are isomorphic to the cyclic group $\mathbb{Z}_4$[13]. By considering the same order of bases, T. Ali and C. K. Phukan (2013) defined a product operation on the base set $P = \{A, C, G, U\}$. With these two binary operations the set P fulfils the postulates of a commutative ring structure with identity element[7, 8].

TABLE 1: Operations on P

+	A	C	G	U
A	A	C	G	U
C	C	G	U	A
G	G	U	A	C
U	U	A	C	G

·	A	C	G	U
A	A	A	A	A
C	A	C	G	U
G	A	G	A	G
U	A	U	G	C

Ali and Phukan(2013) arranged all the codons in the genetic code table by using the Cartesian product of the ring P i.e. $P \times P \times P$ and it is denoted as C_G where $P \times P \times P = \{(x, y, z) : x, y, z \in \{A, C, G, U\}\}$ i.e. $C_G = \{xyz : x, y, z \in \{A, C, G, U\}\}$. The sum and product operations are defined between the codons in the following way $xyz + x'y'z' = (x + x')(y + y')(z + z')$ and $xyz \cdot x'y'z' = (x \cdot x')(y \cdot y')(z \cdot z')$ with these two operations $(C_G, +, \cdot) = (\mathbb{Z}_4 \times \mathbb{Z}_4 \times \mathbb{Z}_4, +, \cdot)$

Authors T. Ali and C.K. Phukan[1] defined an operation on C_G as a module over P i.e. considering $\mathbb{Z}_4$ as a ring and $\mathbb{Z}_4 \times \mathbb{Z}_4 \times \mathbb{Z}_4$ as an additive abelian group. The function $f : \mathbb{Z}_4 \times (\mathbb{Z}_4 \times \mathbb{Z}_4 \times \mathbb{Z}_4) \to \mathbb{Z}_4 \times \mathbb{Z}_4 \times \mathbb{Z}_4$ is such that $f(a, u) = au = ua$, and $\forall a, b \in \mathbb{Z}_4, u, v \in \mathbb{Z}_4 \times \mathbb{Z}_4 \times \mathbb{Z}_4$ with the following properties

(i) $a(u + v) = au + av$
(ii) $(a + b)u = au + bu$
(iii) $(a \cdot b)u = a(bu)$
(iv) $1u = u$

N. Gohain and T. Ali [7, 8] defined a function $f : C_G \to C_G$ s.t. $\alpha \in C_G$, $f(\alpha) = f(\alpha_1\alpha_2\alpha_3) = (UUU - \alpha_1\alpha_2\alpha_3)$. They defined and studied a graph structure named total graph of the set of codons.

The main motivation of this work is for exploring automata structure i.e. transition table, output table and state diagram with some important algebraic structures which naturally occur in genetic code. A sum operation and a product operation had been introduced in the set of codons in a manner that the ring obtained on the set of 64 codons is isomorphic to the ring of integers modulo 64 i.e. $(\mathbb{Z}_{64}, +, \cdot)$.

3. Main work

A well-designed automata framework can take care of mundane and unnecessary manual tasks with speedy, efficiency reduce test maintenance cost as well as lower risks. In this section, we give certain automata/discrete dynamical systems with new concepts on genetic code. We try to fit a automata which is really helpful to explore the characteristic on DNA, RNA sequence of Genetic Biology. In DNA/RNA pattern analysis[12], automata can be widely used to know the character, alteration, or duplicate in gene theory. The main benefit is of utilizing automata is that incomplete information, errors in sequence arrangements, irrelevant mutation etc. in DNA/RNA sequence can be identified more easily.

We define an automata, $\Sigma = (C_G, P, A_M, F, G)$ where $F : C_G \times P \to C_G$ defined by $F(q, a) = qa$,$\forall q \in C_G$, $a \in P$ given in Table 2. The output function $G : C_G \times P \to A_M$ is defined by the Table 3. It is already defined[13] that $C_G \cong \mathbb{Z}_4 \times \mathbb{Z}_4 \times \mathbb{Z}_4$, $P \cong \mathbb{Z}_4$ where $(C_G, +, \cdot)$ and $(P, +, \cdot)$ gives ring structures.

The input function is defined as in Table 2

TABLE 2: Transition Table

F	A	C	G	U
AAA	AAA	AAA	AAA	AAA
AAC	AAA	AAC	AAG	AAU
AAG	AAA	AAG	AAA	AAG
AAU	AAA	AAU	AAG	AAC
ACA	AAA	ACA	AGA	AUA
ACC	AAA	ACC	AGG	AUU
ACG	AAA	ACG	AGA	AUG
ACU	AAA	ACU	AGG	AUC
AGA	AAA	AGA	AAA	AGA
AGC	AAA	AGC	AAG	AGU
AGG	AAA	AGG	AAA	AGG
AGU	AAA	AGU	AAG	AGC
AUA	AAA	AUA	AGA	ACA
AUC	AAA	AUC	AGG	ACU
AUG	AAA	AUG	AGA	ACG
AUU	AAA	AUU	AGG	ACC
CAA	AAA	CAA	GAA	UAA
CAC	AAA	CAC	GAG	UAU
CAG	AAA	CAG	GAA	UAG
CAU	AAA	CAU	GAG	UAG
CCA	AAA	CCA	GGA	UUA
CCC	AAA	CCC	GGG	UUU
CCG	AAA	CCG	GGA	UUG
CCU	AAA	CCU	GGG	UUC
CGA	AAA	CGA	GAA	UGA
CGC	AAA	CGC	GAG	UGU
CGG	AAA	CGG	GAA	UGG
CGU	AAA	CGU	GAG	UGC

F	A	C	G	U
GAA	AAA	GAA	AAA	GAA
GAC	AAA	GAC	AAG	GAU
GAG	AAA	GAG	AAA	GAG
GAU	AAA	GAU	AAG	GAC
GCA	AAA	GCA	AGA	GUA
GCC	AAA	GCC	AGG	GUU
GCG	AAA	GCG	AGA	GUG
GCU	AAA	GCU	AGG	GUC
GGA	AAA	GGA	AAA	GGA
GGC	AAA	GGC	AAG	GGU
GGG	AAA	GGG	AAA	GGG
GGU	AAA	GGU	AAG	GGC
GUA	AAA	GUA	AGA	GGA
GUC	AAA	GUC	AGG	GCU
GUG	AAA	GUG	AGA	GCG
GUU	AAA	GUU	AGG	GCC
UAA	AAA	UAA	GAA	CAA
UAC	AAA	UAC	GAG	CAU
UAG	AAA	UAG	GAA	CAG
UAU	AAA	UAU	GAG	CAC
UCA	AAA	UCA	GGA	CUA
UCC	AAA	UCC	GGG	CUU
UCG	AAA	UCG	GGA	CUG
UCU	AAA	UCU	GGG	CUC
UGA	AAA	UGA	GAA	CGA
UGC	AAA	UGC	GAG	CGU
UGG	AAA	UGG	GAA	CGG
UGU	AAA	UGU	GAG	CGC

CUA	AAA	CUA	GGA	UCA		UUA	AAA	UUA	GGA	CCA
CUC	AAA	CUC	GGG	UCU		UUC	AAA	UUC	GGG	CCU
CUG	AAA	CUG	GGA	UCG		UUG	AAA	UUG	GGA	CCG
CUU	AAA	CUU	GGG	UGC		UUU	AAA	UUU	GGG	CCC

The output function is defined as in the Table 3

TABLE 3: Output Table

G	A	C	G	U		G	A	C	G	U		G	A	C	G	U		G	A	C	G	U
AAA	K	K	K	K		CAA	K	Q	E	-		GAA	K	E	K	E		UAA	K	-	E	Q
AAC	K	N	K	N		CAC	K	H	E	Y		GAC	K	D	K	D		UAC	K	Y	E	H
AAG	K	K	K	K		CAG	K	Q	E	-		GAG	K	E	K	E		UAG	K	-	E	Q
AAU	K	N	K	N		CAU	K	H	E	Y		GAU	K	D	K	D		UAU	K	Y	E	H
ACA	K	T	R	I		CCA	K	P	G	L		GCA	K	A	R	V		UCA	K	S	G	L
ACC	K	T	R	I		CCC	K	P	G	F		GCC	K	A	R	V		UCC	K	S	G	L
ACG	K	T	R	M		CCG	K	P	G	L		GCG	K	A	R	V		UCG	K	S	G	L
ACU	K	T	R	I		CCU	K	P	G	F		GCU	K	A	R	V		UCU	K	S	G	L
AGA	K	R	K	R		CGA	K	R	E	-		GGA	K	D	K	G		UGA	K	-	E	R
AGC	K	S	K	S		CGC	K	R	E	C		GGC	K	D	K	G		UGC	K	C	E	R
AGG	K	R	K	R		CGG	K	R	E	W		GGG	K	D	K	G		UGG	K	W	E	R
AGU	K	S	K	S		CGU	K	R	E	C		GGU	K	D	K	G		UGU	K	C	E	R
AUA	K	I	R	T		CUA	K	L	G	S		GUA	K	V	R	A		UUA	K	L	G	P
AUC	K	I	R	T		CUC	K	L	G	S		GUC	K	V	R	A		UUC	K	F	G	P
AUG	K	M	R	T		CUG	K	L	G	S		GUG	K	V	R	A		UUG	K	L	G	P
AUU	K	I	R	T		CUU	K	L	G	S		GUU	K	V	R	A		UUU	K	F	G	P

The output table shows that the inputs on any codon encode either one of the 20 amino acids or any one of the stop codons UAA, UAG or UGA as an output. This output table will help to identify the base strings(inputs) or the set of codons in the DFA Σ that encode a particular amino acid or protein.

The state diagram of the automata $\Sigma = (C_G, P, A_M, F, G)$ defined on the set of codons is given in figure 1

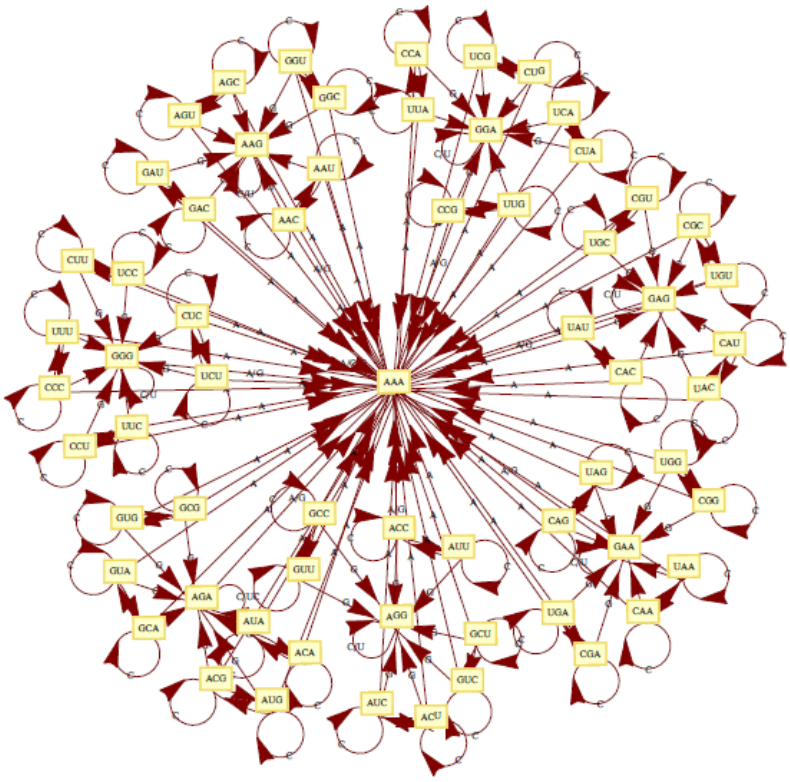

Fig 1: State Diagram of Σ

In automata theory for the string representation of polynucleotide sequences of DNA/RNA where the information about the exact sequences is more or less certain. The Syntactic monoid helped to the researcher to inveastigate the nature of strings over the alphabet A, G, C, T and are used in DNA computing to encode information/data are found very successful. Algebraic characterization of many classes of codes through their Syntactic monoid has been extensively studied. We also get a monoid structure for further study about parallel decomposition of DNA/RNA strings, find distances between the sequences. A Monoid automaton is very adequate tool for the analysis of Non-determinism and parallelism of DNA/RNA computing.

For $a \in P$, we define $f_a : C_G \to C_G$ by $f_a(q) = F(q, a) = qa, q \in C_G$ The semiautomaton defined on C_G is given in the Table 4

TABLE 4: Semi Automata Table

f_a	f_A	f_C	f_G	f_U	f_a	f_A	f_C	f_G	f_U
AAA	AAA	AAA	AAA	AAA	GAA	AAA	GAA	AAA	GAA
AAC	AAA	AAC	AAG	AAU	GAC	AAA	GAC	AAG	GAU
AAG	AAA	AAG	AAA	AAG	GAG	AAA	GAG	AAA	GAG
AAU	AAA	AAU	AAG	AAC	GAU	AAA	GAU	AAG	GAC
ACA	AAA	ACA	AGA	AUA	GCA	AAA	GCA	AGA	GUA
ACC	AAA	ACC	AGG	AUU	GCC	AAA	GCC	AGG	GUU
ACG	AAA	ACG	AGA	AUG	GCG	AAA	GCG	AGA	GUG
ACU	AAA	ACU	AGG	AUC	GCU	AAA	GCU	AGG	GUC
AGA	AAA	AGA	AAA	AGA	GGA	AAA	GGA	AAA	GGA
AGC	AAA	AGC	AAG	AGU	GGC	AAA	GGC	AAG	GGU
AGG	AAA	AGG	AAA	AGG	GGG	AAA	GGG	AAA	GGG
AGU	AAA	AGU	AAG	AGC	GGU	AAA	GGU	AAG	GGC
AUA	AAA	AUA	AGA	ACA	GUA	AAA	GUA	AGA	GGA
AUC	AAA	AUC	AGG	ACU	GUC	AAA	GUC	AGG	GCU
AUG	AAA	AUG	AGA	ACG	GUG	AAA	GUG	AGA	GCG
AUU	AAA	AUU	AGG	ACC	GUU	AAA	GUU	AGG	GCC
CAA	AAA	CAA	GAA	UAA	UAA	AAA	UAA	GAA	CAA
CAC	AAA	CAC	GAG	UAU	UAC	AAA	UAC	GAG	CAU
CAG	AAA	CAG	GAA	UAG	UAG	AAA	UAG	GAA	CAG
CAU	AAA	CAU	GAG	UAG	UAU	AAA	UAU	GAG	CAC
CCA	AAA	CCA	GGA	UUA	UCA	AAA	UCA	GGA	CUA
CCC	AAA	CCC	GGG	UUU	UCC	AAA	UCC	GGG	CUU
CCG	AAA	CCG	GGA	UUG	UCG	AAA	UCG	GGA	CUG
CCU	AAA	CCU	GGG	UUC	UCU	AAA	UCU	GGG	CUC
CGA	AAA	CGA	GAA	UGA	UGA	AAA	UGA	GAA	CGA
CGC	AAA	CGC	GAG	UGU	UGC	AAA	UGC	GAG	CGU
CGG	AAA	CGG	GAA	UGG	UGG	AAA	UGG	GAA	CGG
CGU	AAA	CGU	GAG	UGC	UGU	AAA	UGU	GAG	CGC
CUA	AAA	CUA	GGA	UCA	UUA	AAA	UUA	GGA	CCA
CUC	AAA	CUC	GGG	UCU	UUC	AAA	UUC	GGG	CCU
CUG	AAA	CUG	GGA	UCG	UUG	AAA	UUG	GGA	CCG
CUU	AAA	CUU	GGG	UGC	UUU	AAA	UUU	GGG	CCC

The set $\{fa : a \in P\}$ along with the operation $\circ$ forms a monoid. This monoid $(\{fa : a \in P\}, \circ) = M(\Sigma)$ is called the syntactic monoid of Σ and $\forall a, b \in P$, we have $f_a \circ f_b = f_{ba}$.

The operation on $M(\Sigma)$ is given in the Table 5

TABLE 5: Operations on $M(\Sigma)$

$\circ$	f_A	f_C	f_G	f_U
f_A	f_A	f_A	f_A	f_A
f_C	f_A	f_C	f_G	f_U
f_G	f_A	f_G	f_A	f_G
f_U	f_A	f_U	f_G	f_C

We consider a function $f : C_G \to C_G$, where C_G is an additive group and $C_G = \{xyz : x, y, z \in \{A, C, G, U\}\}$ defined by $f(\alpha) = f(\alpha_1\alpha_2\alpha_3) = (UUU - \alpha_1\alpha_2\alpha_3)$. We consider the

system $M(C_G) = \{f|f : C_G \to C_G\}$. Addition and multiplication in $M(C_G)$ are defined as follows; If $f, g \in M(C_G)$ then for each $\alpha \in C_G$

(i) $(f + g)(\alpha) = f(\alpha) + g(\alpha) = 2(UUU - \alpha_1\alpha_2\alpha_3)$

(ii) $(f \cdot g)(\alpha) = f(g(\alpha)) = UUU - g(\alpha) = \alpha_1\alpha_2\alpha_3$

Then the algebraic structure $(M(C_G), +, \cdot)$ forms a Near-Ring.

4. Conclusion

From the above study, we can say that automata and their various algebraic properties can be used to study genetic structures. As finite automata can be used to study the changes caused by mutation or to find missing gene, we hope the definition of automata and algebraic exposer on the genetic structure system will be very useful in future. Also the automata defined on the codon set is already a DFA so it will help to analyse the DNA pattern. Automata is a non-linear discrete dynamical system. With the study on reachability, stabilization of the chaotic behaviors of the DNA/RNA sequences can be studied. Knowing automata we can discuss different types of algebraic structures on the sets of codons, bases, and amino acids of DNA/RNA. It can help to know the relationship between DNA, RNA, Amino acids more clearly. We are hopeful that with this introduction of automata on codons we can expand our study to a wide range and obtain many fruitful results relating to genetics and Mathematics.

References

[1] T. Ali and C. K. Phukan, *Topology in genetic code algebra*, Math. sci. int. res. j., **2** (2013) 179

[2] F. Antoneli, L. Braggion, M. Forger and JEM Hornos, *Extending the search for symmetries in the genetic code*, Int. J. Mod. Phys. B, **17** No. 17 (2003) 3135-3204.

[3] J. A. N. A. K. I. Balakrishnan, *Symmetry scheme for amino acid codons*, Phys. Rev. E, **65** No. 2 (2002) 021912.

[4] I. S. S. A. C. Barjis, J. W. Yeol and Y. S. Ryu, *Modeling of protein production process by Finite Automata (FA)*, In Proceedings of 2005 WSEAS International Conference on: Mathematical Biology and Ecology (MABE'05) (2005).

[5] J. D. Bashford and P. D. Jarvis, *The genetic code as a periodic table: algebraic aspects*, Biosystems, **57** No. 3 (2000) 147-161.

[6] A. Nowzari-Dalini, E. Elahi, H. Ahrabian and M. Ronaghi, *A New DNA Implementation of Finite State Machines*, Int. J. Comput. Sci. Appl., **3** No. 1 (2006) 51-60.

[7] N. Gohain, T. Ali and A. Akhatar, *Reducing Redundancy of Codons through Total Graph*, Am. j. bioinform., **4** No. 1 (2015) 1.

[8] N. Gohain, T. Ali and A. Akhatar, *Lattice structure and distance matrix of genetic code*, J. Biol. Syst., **23** No. 03 (2015) 485-504.

[9] G. Hofer, *Left ideals and reachability in machines*, Theor. Comput. Sci., **68** No. 1 (1989) 49-56.

[10] J. E. Hopcroft, R. Motwani and J. D. Ullman, *Introduction to automata theory, languages, and computation*, Language and Computations, Acm Sigact News, **32** No. 1 (2001) 60-65.

[11] C. Musili, *Introduction to rings and modules*, 2nd Rev. Edition, Narosa Publishing House, 1994.

[12] Qura-Tul-Ein, Y. Saeed, S. Naseem, F. Ahmad, T. Alyas, N. Tabassum, *DNA pattern analysis using finite automata*, International Research Journal of Computer Science (IRJCS), **1** No. 2 (2014) 1-4.

[13] R. Sánchez, E. Morgado and R. Grau, *Gene algebra from a genetic code algebraic structure*, J. Math. Biol., **51** No. 4 (2005) 431-457

[14] R. Selvakumar, M. Rashith Muhammad and G. Poornima Devi, *Computational model for the extraction of human erythropoietin*, Int. J. ChemTech. Res., **5** (2013) 2623-2629.

Mridul Dutta

Department of Mathematics, Dudhnoi College

Dudhnoi-783124, Goalpara, Assam, India.

mridulduttamc@gmail.com

Sanjoy Kalita

Department of Mathematics,

School of Fundamental and Applied Sciences

Assam Don Bosco University

Tepesia-782402, Kamrup, Assam India.

sanjoykalita1@gmail.com

Helen K. Saikia

Department of Mathematics, University of Gauhati

Guwahati-781014, Assam India.

hsaikia@yahoo.com