J. Integr. Sci. Technol.2023, 11(2), 474

Article

Journal of Integrated
SCIENCE& TECHNOLOGY

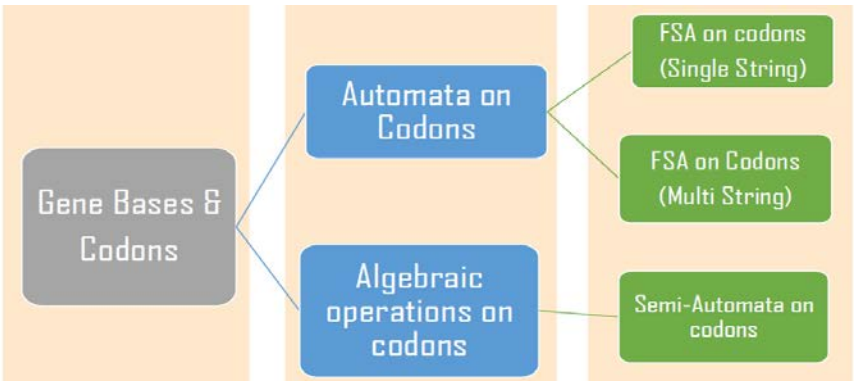
Automata on Genetic Code

Mridul Dutta,[1*] Sanjoy Kalita,[2] Padma Bhushan Borah[3]

[1]Department of Mathematics, Dudhnoi College, Dudhnoi-783124, Goalpara, Assam, India. [2]Department of Mathematics, Assam Don Bosco University, Tepesia-782402, Kamrup, Assam, India. [3]Department of Mathematics, Behali Degree College, Biswanath-784167, Assam, India.

Received on: 22-Oct-2022, Accepted and Published on: 22-Dec-2022

ABSTRACT



Automata are system modeling and analysis tools. By using an automaton, researchers model an abstraction of the behavior of a system so that they can derive the answer with desired analysis from it. The emphasis of this article is the use of automata as a modeling tool to model, simulate, and analyze nucleotide codon processing. There is enormous potential for mathematical and computational approaches that lead to fundamental insights and important practical validation of genetic biology research. Mathematical and computational approaches have been valued in physics, and have played an increasingly crucial role in chemistry over the past twenty years. Now, more and more genetic biologists and biochemists are interested in using automata theory applications and mathematical methods in their work. This work is meant to study automata theory on genetic code. The automaton for the functioning of nucleotide codons of RNA is defined here along with building of different RNA-based automata that give the same state. The results have been illustrated with examples and non-examples besides presentation of example of a module semi-automatic machine.

Keywords: Automata, Codons, Genetic Codes, Module Semi-automaton

INTRODUCTION

The fascinating area of computer science is called automata theory. Mathematicians started creating machines that theoretically and practically mimicked human characteristics to perform calculations more rapidly and accurately during the 20th century, and this is when it first took root. The term "automaton" itself refers to automatic procedures that carry out the creation of particular processes. In a nutshell, automata theory is concerned with the computational logic of so-called automata, which are small, simple machines. Automata theory is a growing area of theoretical study. It has practical uses in a wide range of fields, such as the study of computing behavior. The theory of automata can be used practically. Numerous real-world finite-state systems are described, such as games, puzzles, logical assertions in mathematics, real-time systems, distributed systems, error-correcting codes, etc.[1-4]

Automata is used in Biology to examine a variety of characteristics of life, including DNA, pigments, and associated proteins. Ribo nucleic Acid (RNA) or Deoxy ribonucleic Acid (DNA) make up the genetic material of living things. Chromosomes, mitochondria, and chloroplasts all contain DNA. It functions as genetic material in many organisms that pass genetic information from one generation to the next.[5-9] There are numerous studies reported giving a glimpse towards the recapitulation of well-known history and literature in this field.[1-15]

DNA/RNA computation is a growing area that links the gap in the middle of automata theory and genetic code. Mathematics has been applied in biology for a long time. Several instructions and mathematical techniques are applied in the biological sciences. There are many techniques used in different scientific disciplines to analyze the sequences of DNA and its related RNA. One of these is

*Corresponding Author: Mridul Dutta, Department of Mathematics, Dudhnoi College, Dudhnoi-783124, Goalpara, Assam, India.
Tel: +919957545489
Email:mridulduttamc@gmail.com

Cite as: *J. Integr. Sci. Technol., 2023, 11(2), 474.*
URN:NBN:sciencein.jist.2023.v11.474

 ISSN: 2321-4635
http://pubs.thesciencein.org/jist

finite automata. Finite Automata can be recycled to analyze the sequences of DNA/RNA. In this article, we report the investigation of selected concepts of genetic algebra, in particular for application in genetic codes automation.

PRELIMINARIES

In this article, we attempt to explore DNA, RNA, and protein analysis using the approaches of mathematics and automata theory.

A Semiautomata is a triple $M = (Q, \Sigma, \delta)$ where Q and Σ are state and input sets and $\delta: Q \times \Sigma \to Q$ is called the state transition function. If Q and Σ are R-modules (with the same) and δ is an R-homomorphism, then we called M as a Module - Semiautomaton and abbreviate this by MSA and is denoted by $M = (Q, \Sigma, \delta)_R$.

A Finite Automata[10] can be formally defined as a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where $Q(\neq \phi)$ is a finite set of states, Σ is a finite non-empty set of inputs, $\delta: Q \times \Sigma \to Q$ is defined by $\delta(q_0, a) = q_1$ where $q_0, q_1 \in Q, a \in \Sigma$. $q_0 \in Q$ is the initial state, F is the set of final states, and $F \subseteq Q$. Let Σ^* be the collection of all finite strings over the alphabet Σ, including the empty string. The function δ extends to a function $Q \times \Sigma^* \to Q$ (still denoted by δ) in the following natural way: for every $q \in Q$ and $s \in \Sigma^*$, we set $\delta(q, s) := q$ if s is a empty string and $\delta(q, s) := \delta(\delta(q, z), a)$ if $s = za$ for some string $z \in \Sigma^*$ and some letter $a \in \Sigma$. A string x is accepted by a finite state automata $M = (Q, \Sigma, \delta, q_0, F)$ if $\delta(q_0, x) = p$ for some $p \in F$. A final state is also called an accepting state. The input is accepted when all input is read and matched by transitions and the automaton is in a final state. Also, the table which represents the list of transition functions (rules) of a finite automaton is called the transition table.[10]

An Automata $M = (Q, \Sigma, \Delta, \delta, \lambda)$ is a system where Q is the set of states, Σ is the set of inputs, Δ is the set of outputs, $\delta: Q \times \Sigma \to Q$ is defined as state transition function and $\lambda: Q \times \Sigma \to \Delta$ is defined as an output function respectively.[14]

To find any alterations, additions, or deletions in genes or genetic material, finite automata are used for DNA/RNA pattern analysis.Finite automata have an advantage over random mutations, incorrect sequencing, incomplete data, or DNA specification.[11]

RNA can be conceptualised mathematically as a string of the letters A, G, C, and U.The four bases give us 64 codons as distinct strings of length 3. The three strings UAA, UAG, and UGA, which have the function of stopping the biosynthesis process, are known as stop codons.Starting codon refers to the codon AUG that initiates translation.There are only 20 amino acids in DNA (or RNA), but 64 codons make up the genetic code.[9,15] The codons that code for hydrophobic amino acids can be followed by codons that code for hydrophilic amino acids in this manner.

This ordering of the 64 codons implicitly gives an ordering of the four RNA(or DNA) bases. From which 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 $\mathbb{Z}_4$. By considering the same order of bases, T. Ali and C. K. Phukan (2013)[1] defined a product operation on the base set $P = \{A, C, G, U\}$. With these two binary operations, the set P fulfills the postulates of a commutative ring structure with an identity element.[8,9,15,16]

Table 1: Operations on P

$+$	A	C	G	U		$\cdot$	A	C	G	U
A	A	C	G	U		A	A	A	A	A
C	C	G	U	A		C	A	C	G	U
G	G	U	A	C		G	A	G	A	G
U	U	A	C	G		U	A	U	G	C

Ali and Phukan (2013)[1] arranged all the codons in the genetic code table by using the Cartesian product of the ring Pi.e.$P \times P \times P$ and denote it as C_G i.e. $C_G = P \times P \times P = \{xyz \mid x, y, z \in \{A, C, G, U\}\}$. A sum and product operations is defined between the codons in the following way

$$xyz + x'y'z' = (x + x')(y + y')(z + z')$$
$$xyz \cdot x'y'z' = (x \cdot x')(y \cdot y')(z \cdot z')$$

with these two operations $(C_G, +, \cdot) \cong (\mathbb{Z}_4 \times \mathbb{Z}_4 \times \mathbb{Z}_4, +, \cdot)$

The investigation of automaton structures, including transition tables, output tables, and state diagrams, in addition to some significant algebraic structures that naturally occur in the genetic code, is the primary driving force behind this work.

In order to make the ring on the set of 64 codons isomorphic to the ring of integers modulo 64, a sum operation and a product operation were added to the set of codons. i.e., $(Z_{64}, +, \cdot)$.

AUTOMATON DEFINED ON OPERATION OF CODONS

Sanchez et. al.[12] defined algebraic operations on genetic codes based on the DNA/RNA bases. They have defined a sum operation " $+$" on the sets of bases taking them in two different orders. With the ordered set of bases $\{A, C, G, U\}$ they have defined Primary algebra, and with the ordered set of base$\{U, G, C, A\}$ they defined the Dual algebra with the following operations[12]:

Table 2: Operations on P

Primal						**Dual**				
$+$	A	C	G	U		$+$	U	G	C	A
A	A	C	G	U		U	U	G	C	A
C	C	G	U	A		G	G	C	A	U
G	G	U	A	C		C	C	A	U	G
U	U	A	C	G		A	A	U	G	C

This is also verified that these set of bases with the sum operation form additive abelian groups isomorphic to the group $(\mathbb{Z}_4, +)$. In the genetic code tables above, they found that transitions and transversions are associated with changes in parity. It is assumed that the two arrays of the four base sets $\{A, C, G, U\}$ and $\{U, G, C, A\}$, which reflect the biological relevance of its base codons, form two orders in the codon set.

With the help of these four bases and the operations they have defined the 64 genetic codes and a sum operation on these genetic codes, and also verified that this set of 64 codons with that operation forms a group and is isomorphic to the additive abelian group $(\mathbb{Z}_{64}, +)$.[12]

Sanchez et. al.[12] define the sum operation on the codon sets as follows: Let $X_1X_2X_3$ and $Y_1Y_2Y_3$ are two codons to be added, then

the addition of the bases will be 1. In the order 3,1,2 and the bases will be added according to the sum table.

If $X_i + Y_i = Z_i$ and Z_i are previous in order to X_i and Y_i in the ordered set of bases, then Z_i is written and the base C (or G for the dual group of bases) is added to the next position.

1. For the last base pair only the sum Z_i is written.

Let us take the sum of CGU and CUG, then we will add the bases in the order $U+G, C+C, G+U$. Now $U+G=C$ and a C will be added to $C+C$. So we have $(C+C)+C=G+C=U$ and lastly $G+U=C$.[12] Thus we have the sum as

$$CGU + CUG = (C+C)(G+U)(U+G) = UCC$$

The cyclic character of the sum of codons is hereditarily derived from the base sum, while the order of significance of the bases in the codons is emphasized in the mandate to establish the sum algorithm.

In our work, we tried to represent this concept with the help of automata. We represented the sum operation of the codons with the help of a two-state automata $M=(Q,\Sigma,\Delta,\delta,\lambda)$ where $Q=\{q_0,q_1\}\Sigma=\{AA,AC,AG,AU,CA,CC,CG,CU,GA,GC,GG,$ $GU,UA,\ UC,UG,\ UU\}$, $\Delta=\{A,C,G,U\}q_0$ is the initial state, q_1 is the final state. the state transition function $\delta: Q\times\Sigma\to Q$ is defined in the Table 3.

Table 3: Transition Table

δ	AA	AC	AG	AU	CA	CC	CG	CU	GA	GC	GG	GU	UA	UC	UG	UU
q_0	q_0	q_0	q_0	q_0	q_0	q_0	q_0	q_1	q_0	q_0	q_1	q_1	q_0	q_1	q_1	q_1
q_1	q_0	q_0	q_0	q_1	q_0	q_0	q_1	q_1	q_0	q_1	q_1	q_1	q_1	q_1	q_1	q_1

the output function $\lambda: Q\times\Sigma\to\Delta$ is defined in the Table 4 .

Table 4: Output Table

λ	AA	AC	AG	AU	CA	CC	CG	CU	GA	GC	GG	GU	UA	UC	UG	UU
q_0	A	C	G	U	C	G	U	A	G	U	A	C	U	A	C	G
q_1	C	G	U	A	G	U	A	C	U	A	C	G	A	C	G	U

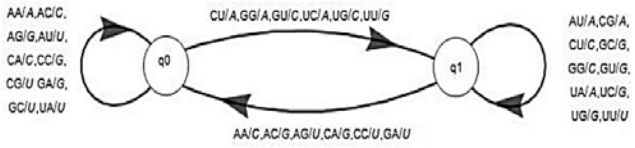


Figure 1: State diagram

In this automaton, if we give input XY of length 2 in the state q_i then first we add $X+Y$ by using the base table. Here we have two cases:

Case 1: $X+Y$ results in an element previous in order in the tables then q_0 will transit to q_1 or there will be a loop at q_1 with the corresponding output.

Case 2: $X+Y$ results in an element later in order in the tables then we will get a loop at q_0 or q_1 will transit to q_0 with the corresponding output.

With these operations defined above, we now define the sum of the codons as follows

$$XYZ + X'Y'Z' = \lambda(\delta(q_0,ZZ'),XX')\lambda(\delta(\delta(q_0,ZZ'),XX'),YY')\lambda(q_0,ZZ')$$

Example 3.1. Let us consider the sum of the codons AGC and UGU. Then the sum operation we have defined, will give us

$$AGC + UGU = \lambda(\delta(q_0,CU),AU)\lambda(\delta(\delta(q_0,CU),AU),GG)\lambda(q_0,CU)$$

From the transition and output tables we have, For$\lambda(q_0,CU),\lambda(q_0,CU)=A$ For $\lambda(\delta(q_0,CU),AU)$, we have $\delta(q_0,CU)=q_1$, and $\lambda(q_1,AU)=A$, so $\lambda(\delta(q_0,CU),AU)=$ A. For $\lambda(\delta(\delta(q_0,CU),AU),GG),\delta(q_1,AU)=q_1$, and $\lambda(q_1,GG)=C$, so $\lambda(\delta(\delta(q_0,CU),AU),GG)=C$. Finally, we obtained $AGC+UGU=ACA$.

Example 3.2. We consider another example, $CGU+CUG$, and evaluate the sum as follows

$$\begin{aligned} CGU + CUG &= \lambda(\delta(q_0,UG),CC)\lambda(\delta(\delta(q_0,UG),CC),GU)\lambda(q_0,UG) \\ &= \lambda(q_1,CC)\lambda(\delta(q_1,CC),GU)C = U\lambda(q_0,GU)C \\ &= UCC \end{aligned}$$

$$\therefore CGU + CUG = UCC$$

In our study, we have considered the primal base set for all these operations with q_0 as the initial state and q_1 as the final state. While studying these we observed that the outputs at q_1 are actually following the sum operation of the dual base sum table. We also observed that when we consider the automata with respect to the dual base set, the same automata can represent the operations on this set with q_1 as the initial stage and q_0 as the final stage.

MODULE SEMIAUTOMATA DEFINED ON GENETIC CODES

A well-designed automata skeleton can perform mundane and unnecessary manual tasks with speed and efficiency, reducing the test cost of maintenance with lower risks. In this portion, we give certain automata/discrete dynamical systems a new idea about genetic codon. We are striving to adapt automata that must go through the properties of the DNA/RNA sequence in genetic biology.

DIFFERENT AUTOMATA ON BASE SET OF RNA GIVES SAME STATES

We consider a finite state automata$M_1=(Q,\Sigma,\delta,q_0,F)$ where, $Q=\Sigma=$ Base set of RNA i.e. $\{A,C,G,U\}$, $\delta: Q\times\Sigma\to Q$ is defined in the Table 5, $q_0=A$ is the initial state of the automata, $F=G$ is the final state of the automata.

Table 5: Transition Table

Single Base(Input) / Base(State)	A	C	G	U
A	A	C	G	U
C	C	G	U	A
G	G	U	A	C
U	U	A	C	G

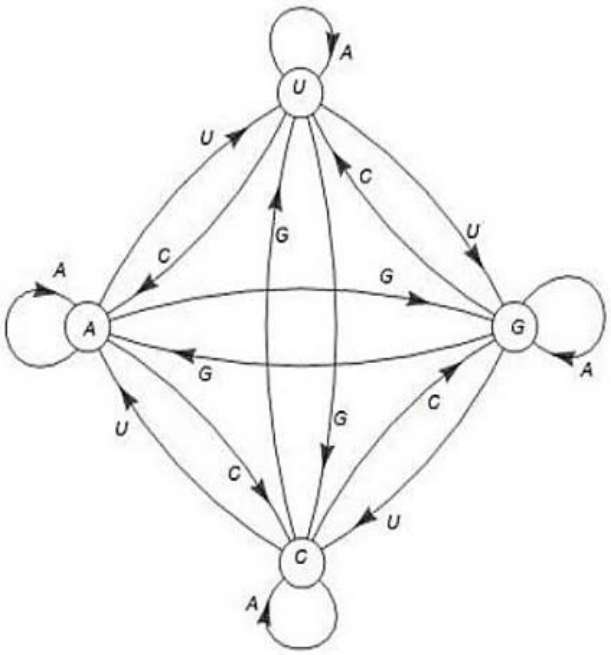


Figure 2: State diagram

We consider a finite state automata $M_2 = (Q, \Sigma, \delta, q_0, F)$ where, $Q =$ Base set of RNA i.e. $\{A, C, G, U\}, \Sigma = \{AA, AC, AG, AU, CA, CC, CG, CU, GA, GC, GG, GU, UA, UC, UG, UU\}, \delta: Q \times \Sigma \to Q$ is defined in the Table 6, $q_0 = A$ is the initial state of the automata, $F = G$ is the final state of the automata.

Table 6: Transition Table

Input / State	AA	AC	AG	AU	CA	CC	CG	CU	GA	GC	GG	GU	UA	UC	UG	UU
A	A	C	G	U	C	G	U	A	G	U	A	C	U	A	C	G
C	C	G	U	A	G	U	A	C	U	A	C	G	A	C	G	U
G	G	U	A	C	U	A	C	G	A	C	G	U	C	G	U	A
U	U	A	C	G	A	C	G	U	C	G	U	A	G	U	A	C

State diagram is shown in Figure 3.

We consider a finite-state-automata $M_3 = (Q, \Sigma, \delta, q_0, F)$ where, $Q =$ Base set of RNA i.e.$\{A, C, G, U\}, \Sigma =$ Base triplet set of RNA of length 3, $\delta: Q \times \Sigma \to Q$ is defined in the Table 6, $q_0 = A$ is the initial state of the automata, $F = G$ is the final state of the automata.

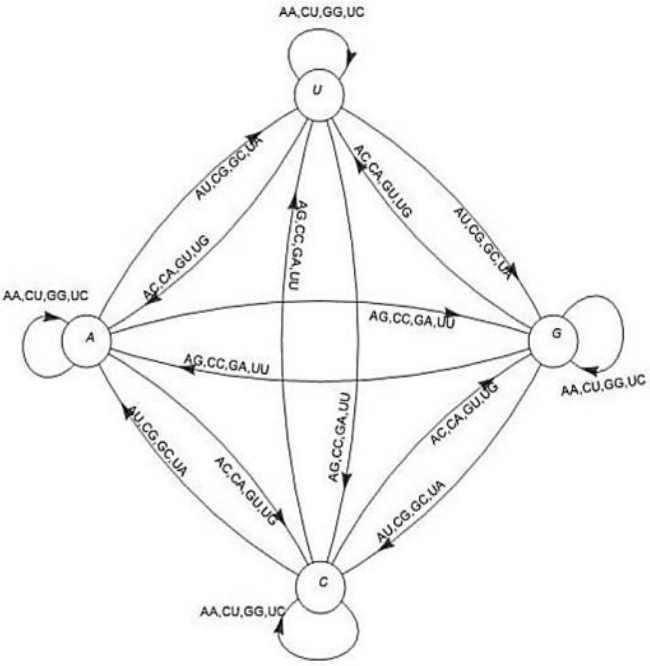


Figure 3: State diagram of M_2

Table 7: Transition Table

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

We considered four disjoint subsets of the set of codons after studying the transition table as follows

$$\alpha = \{AAA, ACU, AGG, AUC, CAU, CCG, CGC, CUA, GAG, GCC, GGA, GUU, UAC, UCA, UGU, UUG\}$$
$$\beta = \{AAC, ACA, AGU, AUG, CAA, CCU, CGG, CUC, GAU, GCG, GGC, GUA, UAG, UCC, UGA, UUU\}$$
$$\gamma = \{AAG, ACC, AGA, AUC, CAC, CCA, CGU, CUG, GAA, GCU, GGG, GUC, UAU, UCG, UGC, UUA\}\backslash$$
$$\eta = \{AAU, ACG, AGC, AUA, CAG, CCC, CGA, CUU, GAC, GCA, GGU, GUG, UAA, UCU, UGG, UUC\}$$

The state diagram of M_3 is shown in Figure 4.

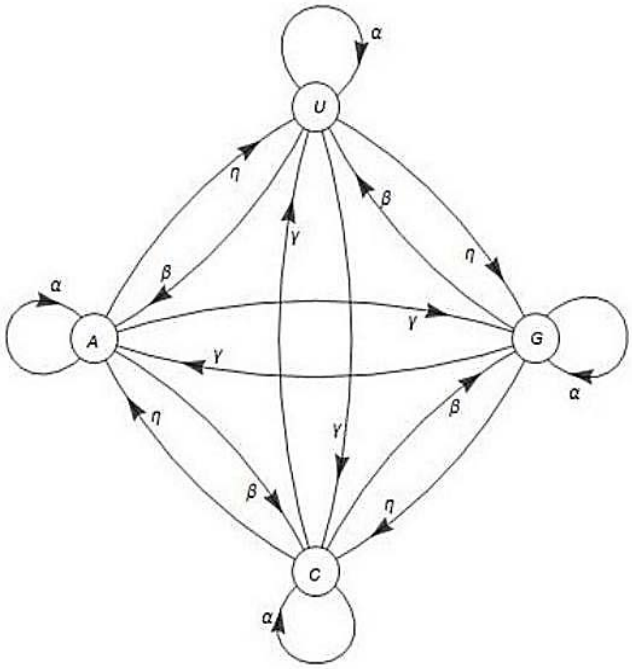


Figure 4: State diagramof M_3

We construct three deterministic finite automata taking inputs as a single length, double-length, and triple length from RNA base set. Here we observe that if we input any string of length multiples of 6 in the above three different automata we get the same state.

Example 4.1. We consider two RNA sequences i.e. CCAAAGUUAAUU, $ACGGGCUGUGC$

We have $\delta(A, CCAAAGUUAAUU)$ according to M_1 as follows

$$\begin{aligned}\delta(A, CCAAAGUUAAUU) &= \delta(\delta(A, C), CAAAGUUAAUU)\\ &= \delta(C, CAAAGUUAAUU)\\ &= \delta(\delta(C, C), AAAGUUAAUU)\\ &= \delta(\delta(G, A), AAGUUAAUU)\\ &= \delta(\delta(G, A), AGUUAAUU)\\ &= \delta(\delta(G, A), GUUAAUU)\\ &= \delta(\delta(G, G), UUAAUU) = \delta(\delta(A, U), UAAUU)\\ &= \delta(\delta(U, U), AAUU) = \delta(\delta(G, A), AUU)\\ &= \delta(\delta(G, A), UU) = \delta(\delta(G, U), U) = \delta(C, U)\\ &= A\end{aligned}$$

According to M_2 we have for $\delta(A, CCAAAGUUAAUU)$ as follows

$$\begin{aligned}\delta(A, CCAAAGUUAAUU) &= \delta(\delta(A, CC), AAAGUUAAUU)\\ &= \delta(\delta(G, AA), AGUUAAUU)\\ &= \delta(\delta(G, AG), UUAAUU) = \delta(\delta(A, UU), AAUU)\\ &= \delta(\delta(G, AA), UU) = \delta(G, UU) = A)\end{aligned}$$

And $\delta(A, CCAAAGUUAAUU)$ with respect to M_3 is

$$\begin{aligned}\delta(A, CCAAAGUUAAUU) &= \delta(\delta(A, CCA), AAGUUAAUU)\\ &= \delta(\delta(G, AAG), UUAAUU)\\ &= \delta(\delta(A, UUA), AUU) = \delta(G, AUU) = A\end{aligned}$$

Also, we have $\delta(A, ACGGGCUGUGC)$ relative to M_1, M_2 and M_3 respectively

$$\begin{aligned}(\delta(A\ , ACGGGCUGUGC) &= \delta(\delta(A, A), CGGGCUGUGC)\\ &= \delta(A, CGGGCUGUGC)\\ &= \delta(\delta(A, C), GGGCUGUGC)\\ &= \delta(\delta(C, G), GGCUGUGC)\\ &= \delta(\delta(U, G), GCUGUGC)\\ &= \delta(\delta(C, G), CUGUGC) = \delta(\delta(U, C), UGUGC)\\ &= \delta(\delta(A, U), GUGC) = \delta(\delta(U, G), UGC)\\ &= \delta(\delta(C, U), GC) = \delta(\delta(A, G), C) = \delta(G, C)\\ &= U\end{aligned}$$

$$\begin{aligned}\delta(A, \&ACGGGCUGUGC) &= \delta(\delta(A, AC), GGGCUGUGC)\\ &= \delta(\delta(C, GG), GCUGUGC)\\ &= \delta(\delta(C, GC), UGUGC) = \delta(\delta(A, UG), UGC)\\ &= \delta(\delta(C, UG), C) = \delta(G, C) = Stuck\end{aligned}$$

$$\begin{aligned}\delta(A\&ACGGGCUGUGC) &= \delta(\delta(A, ACG), GGCUGUGC)\\ &= \delta(\delta(U, GGC), UGUGC) = \delta(\delta(A, UGU), GC)\\ &= \delta(A, GC) = Stuck\end{aligned}$$

Example 4.2.(*Example of Module-Semiautomata)* We consider Semiautomata$M = (C_G, P, \delta)_P$ which is a Module-Semiautomata because

(a) C_G is a module over P, Since $C_G = \mathbb{Z}_4 \times \mathbb{Z}_4 \times \mathbb{Z}_4$ is the set of all 64 codons, $P = \mathbb{Z}_4$ is the set of all four bases A, C, G, U. Considering $(C_G, +) = (\mathbb{Z}_4 \times \mathbb{Z}_4 \times \mathbb{Z}_4, +)$ is an additive abelian group and $(P, +, \cdot) = (\mathbb{Z}_4, +, \cdot)$ is a commutative ring with $f: \mathbb{Z}_4 \times (\mathbb{Z}_4 \times \mathbb{Z}_4 \times \mathbb{Z}_4) \to \mathbb{Z}_4 \times \mathbb{Z}_4 \times \mathbb{Z}_4$ i.e. $f: P \times C_G \to C_G$ is such that $f(a, u) = au = ua, \forall a, b \in P = \mathbb{Z}_4, u, v \in C_G$ with the following properties (i) $a(u + v) = au + av$, (ii) $(a + b)u = au + bu$, (iii) $(ab)u = a(bu)$, (iv) $1u = u$ Together satisfies all the conditions of a module structure on C_G over P.

(b) P is a module over P, Since $(P, +) = (\mathbb{Z}_4, +)$ is an additive abelian group and $(P, +, \cdot) = (\mathbb{Z}_4, +, \cdot)$ is a commutative ring with $g: \mathbb{Z}_4 \times \mathbb{Z}_4 \to \mathbb{Z}_4$i.e.$g: P \times P \to P$ is such that $g(b, t) = bt, \forall b, c, t, s \in P = \mathbb{Z}_4$ with the following properties

(i) $b(t + s) = bt + bs$, (ii) $(b + c)t = bt + ct$, (iii) $(b.c).t = b.(c.t)$, (iv)$1t = t$ Together satisfies all the conditions of a module structure on P over P

(c) $\delta: (\mathbb{Z}_4 \times \mathbb{Z}_4 \times \mathbb{Z}_4) \times \mathbb{Z}_4 \to \mathbb{Z}_4 \times \mathbb{Z}_4 \times \mathbb{Z}_4$i.e.$\delta C_G \times P \to C_G$ defined by $\delta(q, a) = q, \forall q \in C_G, a \in P$ is a ring homomorphism.

Hence, $M = (C_G, P, \delta)_P$ is a Module-Semiautomata.

CONCLUSION

In this article, the authors have presented the genetic biological coding in the form of automata i.e. the authors have studied automata for codons in genetic biology, because the relevant codon properties have enabled us to build a genetic codon structure that is isomorphic to the algebraic structures. In this article, the authors defined automata on the operation of codons of DNA or RNA nucleotides. We attempted to construct different automata on a base

set of RNA which gives the same state. Also, we introduce an example of Module-Semiautomata.

CONFLICT OF INTEREST

Authors declare no conflict of interest.

REFERENCES

1. T. Ali, C. K. Phukan, Topology in Genetic Code Algebra, *Math. Sci. – Int. Res. J.*, **2013**, 2, 179.
2. F. Antoneli, L. Braggion, M. Forger, JEM Hornos, Extending the search for Symmetries in the Genetic Code, *Int. J. Mod Phys. B*, **2003**, 17, 3135-3204.
3. G. Hofer, Ideals and reachability in machines. In North-Holland Mathematics Studies Vol.137(**1987**), pp.123-131. North-Holland.
4. J. Balakrishnan, Symmetry Scheme for Amino Acid Codons, *Phys. Rev. E*, **2002**, 65, 021912-(1-5).
5. I. Barjis, J. W. Yeol, 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).
6. J. D. Bashford, P. D. Jarvis, The Genetic Code as a Periodic Table: Algebraic Aspects, *Biosystems*, **2000**, 57, 147-161.
7. A.N. Dalini, E. Elahi, H. M. Ahrabian, Ronaghi, A New DNA Implementation of Finite State Machines, *Int. J. Computer Sci. Appl.* **2006**, 3(1), 51-60.
8. N. Gohain. T. Ali, A. Akhatar, Reducing Redundancy of Codon through Total Graphs, *Am. J. Bioinformatics*, **2015**, 4 (1),
9. N. Gohain, T. Ali, A. Akhatar, Lattice structure and distance Matrix of Genetic Code, *J. Biological Systems*, **2015**, 23 (3), 1-20.
10. J. E. Hopcroft, R. Motwani, V. Ullman, Introduction to Automata Theory, Language and Computations, (**2001**) Addison Wesley.
11. Y. Saeed, T. Alyas, S. Naseem, DNA pattern Analysis using Finite Automata. *Int. Res. J. Comp. Sci.*, **2014**, 1(2), 1-4.
12. R. Sanchez, E. Morgado, R. Grau, Gene Algebra from a Genetic Code Algebraic Structure, *J. Math. Biol.*, **2005**, 51, 431-457.
13. R. Selvakumar, M. R. Muhammad,G. P. Devi, Computational Model for the Extraction of Human Erythropoietin, *Int. J. Chem Tech Res.*, **2013**, 5(5), 2623-2629.
14. G. F. Pilz, Near-rings and non-linear dynamical systems. In North-Holland Mathematics Studies, Vol. 137, (**1987**) pp. 211-232, North-Holland.
15. M. Dutta, S. Kalita, and H. K. Saikia, Automata on Genetic Structure, *Algebraic Structures and their applications*, **2022**, 9(1), 109-119.
16. J. Demongeot, A. Elena, G. Weil. Potential automata. Application to the genetic code III. *Comptes Rendus Biologies*, **2006**, 329(12), 953-962.