Splicing System in Automata Theory : A Review

Splicing System in Automata Theory : A Review

S H Khairuddin, M A Ahmad and N Adzhar

Pusat Sains Matematik, Universiti Malaysia Pahang, 26300 Lebuhraya Tun Razak, UMP Gambang, Pahang, Malaysia

sitihajarmohdkhairuddin96@gmail.com

Abstract. The study of formal language theory rapidly evolves after Tom Head introduce his research on formal language theory in 1987. Splicing system involves the process of cutting and pasting on DNA molecules with the presence of restriction enzymes and ligase, respectively. A mathematical model of the splicing system has been developed by using the concept of formal language theory, which is a branch of theoretical computer science and applied discrete mathematics, and also informational macromolecules. Over the year, theoretical results in splicing systems have contributed to new research in formal language theory focused on modelling of biochemical processes. In this paper, the relation between formal language theory and some related molecular biological terms are explored. In addition, new ideas in the framework of biomolecular science, for example, the design of automated enzymatic processes are then discussed. Then, a mutual relation that exist in these field is then explained. The regular language can be implemented in the splicing system to show the DFA structure in the splicing system.

1. Introduction

A polymer that strung together from monomers and also an important cell located inside every living organism is called deoxyribonucleic acid (DNA). DNA molecules are made up from thousands of pairs of nucleotides which consist of a base, phosphate, and sugar [1]. The base is where the information in DNA stored and it built up from four types of chemical bases which are adenine (A), guanine (G), cytosine (C) and thymine (T). Furthermore, by following the Watson-Crick complementarity the only possible pairing out of those bases are adenine with thymine (A - T), guanine with cytosine (C - G) and vice versa [2]. In 1987, Tom Head introduced the formal presentation in presenting the recombination of DNA molecules [3]. As time evolved, many researchers made research regarding splicing system which then contribute to the formation of enhanced or extended version of splicing system such as Paun, Pixton, Goode-Pixton (G-P) and Yusof-Goode (Y-G) splicing systems.

Besides, formal language theory is a branch of theoretical computer science. It is devoted to the study of sets of finite strings (called languages) of symbols chosen from a finite set which called an alphabet [4]. Besides, formal language theory is always resembled to the study of informational macromolecules. The language is associated with each pair of sets where the first set is consisting of double stranded DNA (dsDNA) molecules and another set consists of the recombination behaviors allowed by specified classes of enzymatic activities [5]. The associated language consists of strings of symbols that represent the primary structures of the DNA molecules that may arise from the original set of DNA molecules under the given enzymatic activities. In addition, the associated languages that analysed by means of a new generative formalism called splicing system [6].

Basically, a splicing system that relates with formal language theory and the study of informational macromolecules is originally introduced by Head in 1987 [6]. Splicing system is a formal model that uses contextual cross-over operation over words to generate languages called splicing languages [7]. This cross-over splicing process is validated through the behavior of basic biomolecular processes. The cross-over process involving cut and paste of dsDNA which is performed by restriction enzymes and a ligase respectively. In addition, restriction enzymes act on dsDNA molecules by cleaving certain recognised segments and leaving short single stranded overhangs [6]. Molecules with the same overhangs can join with each other in the presence of a ligase enzyme. Tom Head proved that if the splicing process is performed by a finite set of certain simple rules, then splicing of finite set of words can generate the class of strictly locally testable languages [8].

On the other side, the latest suggested field in the framework of biomolecular science is the design of automated enzymatic processes [9]. In addition, automated enzymatic process is an enzymatic process that is illustrated by using the automata diagram [10]. Moreover, splicing system is meant to have a finite set of rules (modelling enzymes) applied on a finite set of initial strings (modelling DNA sequences). A splicing system (or H-system) is a triple, where A is a finite alphabet, I is the initial language and B and C are the set of rules which is for Head splicing system and otherwise for Y-G splicing system such that A is a finite alphabet, I is the initial language and R is the set of rules. In this paper, the focus is on finite splicing systems instead of infinite splicing system.

Finally, after the language had been generalised, the automata diagram is then illustrated to show the flow of the limit language and how the presence of the limit language in splicing system works. Automata theory is a study of abstract machine and automata which also included the computational problems that can be solve using it [11]. Somehow, a regular language and finite automata are the importance role in pattern matching for developing the automata diagram that generated by the regular language [12]. In the literature, various methods are available for constructing deterministic finite automata (DFA) like subset construction method which finds DFA from non-deterministic finite automata (NFA) [13].

The formal presentation in presenting the recombination of DNA molecules are discussed and presented. The fundamental knowledge of splicing system is then explained. Moreover, the relationship between splicing system and automata theory are explained in a simplest way by using transition diagram.

2. Splicing Systems

Splicing system from biological perspective is modelled based on the formulation of the new hybrid DNAs after undergoes process such as cutting and pasting dsDNA with the presence of restriction enzyme and ligase in a test tube [14]. Mathematical modelling of the splicing system inspired from the presentation of DNA molecules as a series of alphabets, namely A, C, G and T. The following example is first illustrated to give the basic idea on the DNA splicing process which later leads to the mathematical modelling of the splicing system.

The definition of splicing system which has been introduced by Head in [6] is given as follows.

Definition 1 [3]: Head Splicing System

A splicing system $S = (A, I, B, C)$ consists of a finite alphabet A, a finite set I of initial strings in A^*, and finite sets B and C of triples (c, x, d) with c, x and d in A^*. Each such triple in B or C is called a pattern. For each such triple the string cxd is called a site and the string x is called a crossing. Patterns in B are called left patterns and patterns in C are called right patterns. The language $L = L(S)$ generated by S consists of the strings in I and all strings that can be obtained by adjoining the words $ucxfq$ and $pexdf$ to L whenever $ucxdv$ and $pexfq$ are in L and (c, x, d) and (e, x, f) are patterns of the same hand. A language L is a splicing language if there exists a splicing system S for which $L = L(S)$.

Additionally, from the string cxd this can conclude that the alphabet c is called left context and the alphabet d is called the right context. Moreover, the splicing system that initiated by Head be made up of four different set of elements which are *A, I, B* and *C*. *A* is a set of alphabet , *I* is a set of initial string and both *B* and *C* is a set of rule which represent $5'$-overhang or blunt end and $3'$-overhang respectively. In addition, after the cutting process by the restriction enzyme, the process of pasting will occur with the presence of ligase and the process will strictly obey Watson-Crick complementary. Restriction enzyme will cut the DNA molecules and three types of the cut will be observed either $5'$-overhang, $3'$-overhang and blunt end. The illustration of cutting process of the DNA is explained below. The illustration below based on the dsDNA initial strands and restriction enzymes that have been chosen in [15].

The following string contains restriction sites of *Abs*I that is $(cc, tcga, gg)$.

C	C	T	C	G	A	G	G
G	G	A	G	C	T	C	C

Figure 1. Initial string of dsDNA.

When the restriction enzyme is added, it will produce $5'$-overhang cut. The illustration is given as follows:

C	C	T	C	G	A	G	G
G	G	A	G	C	T	C	C

Figure 2. The cutting site of dsDNA that produces $5'$-overhang.

C	C				
G	G	A	G	C	T

T	C	G	A	G	G
				C	C

Figure 3. Initial string after the cutting process.

The following string contains restriction sites of *Aat*II that is $(g, acgt, c)$.

G	A	C	G	T	C
C	T	G	C	A	G

Figure 4. The initial string of dsDNA.

When the restriction enzyme is added, it will produce $3'$-overhang cut. The illustration is given as follows:

G	A	C	G	T	C
C	T	G	C	A	G

Figure 5. The cutting site of dsDNA that produces $3'$-overhang.

G	A	C	G	T
C				

				C
T	G	C	A	G

Figure 6. Initial string after the cutting process.

Lastly, the following string contains restriction sites of *Aan*I that is (tta, taa).

T	T	A	T	A	A
A	A	T	A	T	T

Figure 7. Initial string of dsDNA.

When the restriction enzyme is added, it will produce blunt end cut. The illustration is given as follows:

T	T	A |	T	A	A
A	A	T |	A	T	T

Figure 8. The cutting site of dsDNA that produces blunt-end.

T	T	A
A	A	T

T	A	A
A	T	T

Figure 9. Initial string after the cutting process.

The example below illustrates the ideas of splicing system introduced by Head.

Example 1 Let $S = \{A, I, B, C\}$ be Head splicing system, where $A = \{a, c, g, t\}$, I (unspecified), $B = \{cc, tcga, gg\}$ and $C = \varnothing$. The restriction enzyme used is *Abs*I. Where α with α' and β with β' are complement to each other and $\alpha, \beta, \alpha', \beta' \in A^*$. The initial string of dsDNA I_0 is presented.

$$I_0 = \begin{matrix} 5'-\alpha CCTCGAGG\beta-3' \\ 3'-\alpha' GGAGCTCC\beta'-5' \end{matrix}$$

Then, in a solution there are multiple copies of dsDNA and also to be considered the 180 degrees rotation of the initial string [16] . I_{180} is present as follows:

$$I_{180} = \begin{matrix} 5'-\beta' CCTCGAGG\alpha'-3' \\ 3'-\beta GGAGCTCC\beta'-5' \end{matrix}$$

After the first splicing occurs, the language is generated as follows:

$$\{\alpha cctcgagg\beta\} \mapsto^R \{\alpha cctcgagg\beta, \alpha cctcgagg\alpha', \beta' cctcgagg\beta\}$$

The splicing language is resulted by the splicing system. The types of splicing language are then discussed in the next section. There are several types of splicing system and they are defined as follow.

3. Types of Splicing Language

Definition 2 [17] : Two Stages Splicing Language

Let $S = (A, I, B, C)$ be a splicing system. Let $L = L(S)$ be the first stage of one splicing language produce by splicing system and $L' = L'(S)$ is the second stage of two splicing language produce by S that consist of $L = L(S)$ and all splicing languages that can be resulted by splicing L. Then, the union stage one and stage two splicing language are called two stages splicing language.

Definition 3 [18] : First Order Limit Language

The first order limit language or better known as limit language is the set of words that are predicted to appear if some amount of each initial molecule is present, and sufficient time has passed for the reaction to reach equilibrium state, regardless of the balance of the reactants in a particular experimental run of the reaction. To illustrate the language in splicing system, the following example is given

Example 2 Let $S=(A,I,B,C)$ where A is a set of alphabets, $A=\{a,c,g,t\}$ then, $I=(aaaa,bbbb)$ is a set of initial strings then $B=\{r\}$ and $C=\varnothing$. $B=\{r\}$ such that $r_1=(a,aa,a)$ and $r_2=(b,bb,b)$. So that the splicing language is $L(S)=\{aaaa,bbbb,abbb,baaa\}$ and the first order limit language is $L_1(S)=\{abbb,baaa\}$. The string $aaaa$ and $bbbb$ are used to form the string $abbb$ and $baaa$. Then the process of cutting cannot be further. Hence, it is the first order limit language.

The extension of first order limit language which is second order limit language is the defined below.

Definition 4 [19] : Second order Limit Language

Let $L(S)$ be the splicing language of a splicing system S and $L_1(S)$ is the first order limit language. A splicing language is called a second order limit language, denoted by $L_2(S)$, if the set of string produce in $L_2(S)$ is distinct from the set of strings of $L(S)$, that is $L_2(S)\cap L(S)=\emptyset$ and $L_2(S)\cap L_1(S)=\emptyset$, $L_2(S)\not\subset L(S)$.

The next definition is the transient language and active persistence language that have been used in splicing system.

Definition 5 [9] : Transient Language

If a set of string eventually used up and disappears in a given system, the splicing language is called transient language.

Definition 6 [13] : Active persistence Language

An active persistence language is containing in a limit language. The active persistence language is a set of string that participate in a splicing system.

Then, the languages below is generated by the fuzzy splicing system and defined as follows.

Definition 7 [22] : Fuzzy Language

The fuzzy language is generated by the fuzzy splicing system, $\gamma=(V,T,A,R,\mu,\odot)$ is defined as $L_f(\gamma)=\{(z,\mu(z))\in\sigma^*{}_f(A):z\in T^*\}$.

Definition 8 [22] : Crisp Language

The crisp language is generated by a fuzzy splicing system $\gamma=(V,T,A,R,\mu,\odot)$ is defined as $L_c(\gamma)=\{z:(z,\mu(z))\in L_f(\gamma)\}$.

A splicing language can be rewritten in a form of a language generated by grammar. Automata diagram is the illustration used to illustrate the splicing language mechanism. In addition, generators and acceptors are the mechanism for defining the language. Furthermore, the acceptor in this study is deterministic finite automaton meanwhile the generator is grammar. The grammar is therefore defined below.

Definition 9 [5]: Grammar

A grammar can be formally written as a 4-tuple (N,T,S,P) where:

N or V_n is a set of variables or non-terminal symbols

T or Σ is a set of terminal symbols

S is special variable call start symbols, $S \in N$

Then, the production, $P: S \to AB, A \to a, B \to b$

The following example discussed the steps involve in obtaining the grammar from the language.

Example 3 Based on Example 1, the splicing system S where is splicing language, $L(S)$, can be written as a language generated by a grammar G.

$$L(S) = \{\alpha cctcgagg\beta, \alpha cctcgagg\alpha', \beta' cctcgagg\beta\}$$

Therefore, the grammar G_1 is $G_1 = (\{S, A, B, C\}, \{a, c, g, t\}, S, P_1)$ where P_1 consist of:

$$S \to A$$
$$A \to \alpha cctcgB$$
$$B \to agg\beta$$
$$C \to \lambda$$

The sequence of language generated by grammar G_1, $L(G_1)$ is $S \Rightarrow A \Rightarrow \alpha cctcB \Rightarrow \alpha cctcgagg\beta$

Next, the grammar G_2 is $G_2 = (\{S, A, B, C\}, \{a, c, g, t\}, S, P_2)$ where P_2 consist of:

$$S \to A$$
$$A \to \alpha cctcgB$$
$$B \to agg\alpha'$$
$$C \to \lambda$$

The sequence of language generated by grammar G_2, $L(G_2)$ is

$S \Rightarrow A \Rightarrow \alpha cctcgB \Rightarrow \alpha cctcgagg\alpha'$

Lastly, grammar G_3 is $G_3 = (\{S, A, B, C\}, \{a, c, g, t\}, S, P_3)$ where P_3 consist of:

$$S \to A$$
$$A \to \beta' cctcgB$$
$$B \to agg\beta'$$
$$C \to \lambda$$

Based on the above grammar G_1, G_2, G_3, then the automaton diagram is then illustrated in the next subtopic.

Grammar helps in structuring the automaton diagram and determine the number of states to be included in the diagram [23]. All language that have been introduce above are the language that will be recognised by the deterministic finite automata [24]. Deterministic finite automaton is a tool that helps to recognised language in a set of DNA strings [25]. In the next section, the further information of deterministic finite automata is then presented.

Chomsky Classification of Grammars

According to Chomosky in 1956 [27], there are four types grammar, Type 0, Type 1, Type 2 and Type 3. All grammar types are different from each other. Type 0 is unrestricted grammar which accepts only recursively numerable language and Turing machine is the type of automaton. Second, Type 1 is context-

sensitive grammar that accepts context-sensitive language and linear-bounded automaton is the type of automata works. Next, Type 2 that can accept contact-free grammar and context-free language and push-down automaton works. Finally, the highlight grammar type used in this paper is regular grammar and only accepts a regular language and works with automatic finite state. The illustration below shows each scope of grammar.

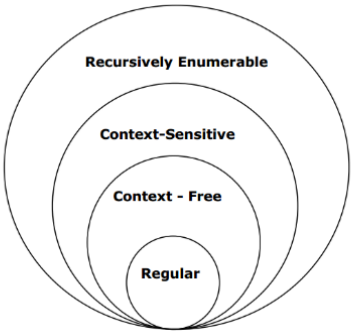

Figure 10. Containment of Type 3 $\subseteq$ Type 2 $\subseteq$ Type 1 $\subseteq$ Type 0.

4. The Deterministic Finite Automaton and Its Application for The Splicing System

A deterministic automaton is a tool that recognise a splicing language. The splicing process are conducted and then splicing language are then generated. Then, formal definition and the mechanism are then defined and explained, respectively.

A deterministic finite automata (DFA) accepts a string if it started from the start state and moving from state to state, each time following the arrow that corresponds the current input character, it reaches a final state when the entire input string is consumed [26]. Otherwise, it rejects the string. Formally, a deterministic finite automaton is a 5-tuple [27] :

Q is a finite set called the states
Σ is a finite set called the alphabet
$\delta : Q \times \Sigma \to Q$ is the transition function
$q_0 \in Q$ is the starting state
$F \subseteq Q$ is the set of accepting state

Under the following conditions, a deterministic finite acceptor (DFA) can be operated. It is assumed to be in the initial state q_0, at the initial time, with its input mechanism or the arrow is on the input string's left symbol. The input mechanism moves one step to the right at each movement of the automaton, so that each movement consists of one input string. The input mechanism moves one step to the right at each movement of the automaton, so that each movement consists of one input symbol. If the automaton is in one of its final states after reaching the end symbol. Then it accepts or rejects the string. The mechanism of input reads only one symbol per step and moves from left to right. The transition function δ commands the movement from one internal state to another. For example, if

$$\delta(q_0, a) = q_1$$

then if the DFA is in state q_0 and input symbol is a, the DFA will go to state q_1.

In addition, illustration of finite automata visualisation is done by using transition graphs. The transition graph consists of vertices represent states and transitions are represented by edges. The labels on the vertices are the state names, whereas the edge labels are the input symbol's current values.

For example, if q_0 and q_1 are internal states of some DFA, then the graph associated with *M* will have one vertex labelled q_0 and another labelled q_1. An edge (q_0, q_1) labelled a represents the transition $\delta(q_0) = q_1$. The initial states are identified through incoming unlabelled arrows that do not originate from any vertex. The final states are drawn by using double circle.

In general, if $M = (Q, \Sigma, \delta, q_0, F)$ is a DFA, then it can be represented by a transition graph G_m that has exactly $|Q|$ verticals and each one labelled with a different $q_i \in Q$.

Example 3 DFA can be represented as follows.

$$M = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_1\})$$

Where δ is given by

$$\begin{aligned} &\delta(q_0, 0) = q_0 \qquad &&\delta(q_0, 1) = q_1 \\ &\delta(q_1, 0) = q_0 \qquad &&\delta(q_1, 1) = q_2 \\ &\delta(q_2, 0) = q_2 \qquad &&\delta(q_2, 1) = q_1 \end{aligned}$$

DFA agrees to accept 01 string. Starting in state q_0, first read the symbol is 0. Looking at the graph's edges, the automaton stays in state that can be seen. After that, the symbol 1 is been read and the automaton is transferred into state q_1. We are now in a final state q_1 at the end of the string and at the same time. The string 01 is therefore accepted. The DFA does not accept string 00 because it will be in state after reading two consecutives 0's. The graph of the transition is shown below.

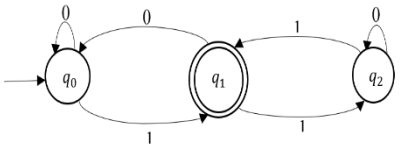

Figure 11. The automaton diagram for example 3.

Based on the basic example presented above, the grammar generated before in Example 3 is associated with the automaton diagram to show the flows of the grammar by representing deterministic finite automata.

Example 4 Based on grammar G_1, G_2, G_3 in the Example 3, represent the DFA, grammar

$$M = \left(\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, c, g, t\}, \sigma, q_0, \{q_2, q_3, q_5\}\right)$$

.

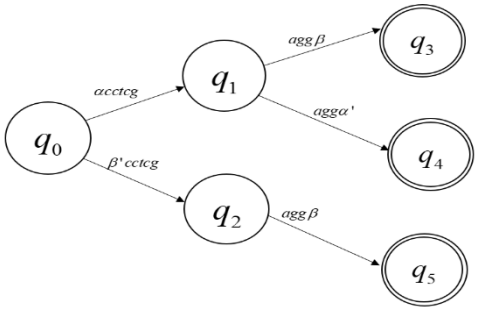

Figure 12. The automaton diagram for example 4.

Regular language in Deterministic Finite Automata

Every finite automaton accepts some language. All possible finite automata are being considered and it will get a set of language associate with it [13] . Only limited family of languages will be accepted by DFA. The structure and properties of the languages in this family will become clearer as our study proceeds.

Language L is called regular if and only if there is existing of DFA M such that

$$L = L(M)$$

The mechanism below show that if and only if the language is generated by grammar is a regular language and the process is as follows:

$$L = \{awa : w \in \{a,b\}^*\}$$

By finding the DFA of above language, it shows that the language is regular. The first basic construction of a DFA of this language is started by checking whether a string begins and ends with an a. The solution is complicated by the fact that there is no explicit way of testing the end of the string. This problem can be overcome by putting the DFA into a final state whenever the second a is encountered. If this is not the end of the string, and another b is found, it will take the DFA out of the final state. It continues by each a taking the automaton back to its final state.

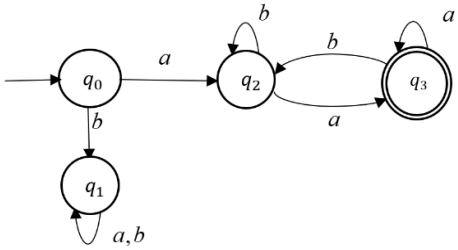

Figure 13. The automaton diagram for regular language.

DFA will only accept a string if it starts and ends with an a. Since DFA is designed for the language, it can be claimed that the language is regular by definition.

5. Conclusion

Based on the previous discussion, there are several types of splicing system and in this paper, the basic concept of splicing system, splicing language and deterministic finite automata is been discussed. In this paper, the focus is on regular language and the deterministic finite automata is used to recognised the regular language. In addition, the definition given in this paper are not fully used in the example since this paper is a review paper, the reader of this paper might use it. In other hand, another type of automata is not relevant in this study for an example non-deterministic finite automaton because the language that we discussed is a finite language and deterministic finite automata is only relevant with a regular language. We also discuss the relationship of splicing system and automata theory. This shows that automata theory is the way to improve our understanding of splicing system work under a systematic transition diagram.

Acknowledgement

All authors are indebted to Universiti Malaysia Pahang for the financial funding through RDU1703278**.**

References

[1] Paun and G. Rozenberg AS 1998 *DNA computing: New computing paradigms* (New York: Springer-Verlag Berlin Heidelberg)
[2] Tamarin RH 2001 *Principle of Genetics* 7th. ed. (USA: The McGraw-Hill Companies)
[3] Head T 1987 *Formal Language Theory and DNA: An Analysis of the Generative Capacity of Specific Recombinant Behaviors.* **49(6)** 737–759
[4] Fong WH and Ismail NI 2018 *Generalisations of DNA Splicing Systems with One Palindromic Restriction Enzyme.* **34(1)** 59–71
[5] Kari L and Kopecki S 2017 *J. Comput Syst Sci.* **84** 263–287
[6] Head T 1998 *Splicing representations of strictly locally testable languages*. **98**
[7] Freund R, Kari L and Păun G 1999 *Theory Comput Syst.* **32(1)** 69–112
[8] Head T 1998 *Splicing Languages Generated With one Sided Context, Computing With Bio-molecules-Theory and Experiments* Gh. Păun ed (Springer-Verlag, Singapore) pp 269–283
[9] Babu S 2015 *Applications of Finite Automata in Text Search – A Review.* **5(5)** 116–119
[10] Wason S, Rathi S and Kumar P 2014 *Research Paper on Automata*. **1(5)** 507–510
[11] P L 2012 *An introduction to formal languages and automata* 5th ed (USA:Jones and Bartlett, LLC)
[12] Santhanam V 2017 *Pattern Matching using Computational and Automata Theory.* pp 1295–1299
[13] Gribkoff E 2013 *Applications of Deterministic Finite Automata A Non-Exhaustive List of DFA Applications.* pp 1–9
[14] Goode TE 1991 *DIMACS Ser Discret Math Theor Comput Sci.* **48** pp 73-83
[15] New England BioLab. Enzyme Finder [Internet]. [cited 2019 Apr 22]. Available from: https://enzymefinder.neb.com
[16] Kari L 1991 *DNA Computing: Arrival of Biological Mathematics* (Springer) **19(2)** pp 2-9
[17] Mudaber MH, Yusof Y and Mohamad MS 2017 *J Phys Conf Ser.* **890(1)**
[18] Goode E and Pixton D 2004 *Splicing to the Limit*. pp 189–201
[19] Ahmad MA 2016 S*econd Order Limit Language and its Properties in Yusof-Goode Splicing System.* (Universiti Teknologi Malaysia)
[20] Goode E 2004 *Splicing to the Limit.* (Lect Notes Comput Sci.)
[21] Bonizzoni P and Mauri G 2005 *Theor Comput Sci.* **340(2)** pp 349–363
[22] Karimi F and Turaev S. Fuzzy Splicing System. 2014;8733(September). Available from: http://link.springer.com/10.1007/978-3-319-11289-3
[23] Bottoni P, Mauri G, Mussio P and Păun G 1998 *Grammars Working on Layered Strings.* (Acta Cybern.) **13(4)** pp 339–358
[24] Trahtman AN 2005 *Precise estimation of the order of local testability of a deterministic finite automaton.* (September 1997) pp 198–212
[25] Sitiarik P and Mindstorm L 2017 *Laboratory Model of Deterministic Finite Automaton.* (April)
[26] Hopcroft JE 2013 *Introduction to Automata Theory, Languages, and Computation* (Pearson New International Edition. Pearson Education)
[27] Konstantinidis S 2013 *Implementation and Application of Automata.* 18th Int Conf. LNCS 7983.