Malaysian Journal of Computing, 8 (2): 1652-1663, 2023
Copyright © UiTM Press
eISSN: 2600-8238

GRAPHICAL USER INTERFACE FOR BOUNDED-ADDITION FUZZY SPLICING SYSTEMS AND THEIR VARIANTS

Mathuri Selvarajoo[1*], Mohd Pawiro Santono[2], Fong Wan Heng[3], and Nor Haniza Sarmin[4]

[1*,2]*College of Computing, Informatics and Mathematics, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia*
[3,4]*Department of Mathematical Sciences, Faculty of Science, Universiti Teknologi Malaysia, UTM, 81310 Johor Bahru, Johor, Malaysia*
[1*]mathuri@tmsk.uitm.edu.my, [2]pawirosantono98@gmail.com, [3]fwh@utm.my and [4]nhs@utm.my

ABSTRACT

A splicing system is one of the early theoretical proposals of the DNA-based computation device. The splicing operation starts when two DNA molecules are cut at specific subsequences with the presence of restriction enzymes: the first part is then connected to the second part of the other molecule, or vice versa, to produce splicing languages. Fuzzy with bounded-addition operation has been introduced as a restriction in splicing systems to increase the generative power of the languages generated. In this research, a graphical user interface is developed to generate all the splicing languages generated by bounded-addition fuzzy splicing systems and their variants. An algorithm is developed using JAVA and Visual Studio Code software in order to replace the time-consuming manual computation of the languages generated by bounded-addition fuzzy DNA splicing systems and their variants.

Keywords: *Graphical User Interface, Fuzzy Bounded-Addition, Splicing Systems, Formal Language Theory.*

Received for review: 16-01-2023; Accepted: 02-10-2023; Published: 10-10-2023
DOI: 10.24191/mjoc.v8i2.20270

1. Introduction

Each living entity has its own deoxyribonucleic acid (DNA). Watson and Crick (Yusof et al., 2011) proposed the double-helical structure of DNA for the first time in 1953. Nucleotides, which are monomers, are used to make DNA molecules. The structure of nucleotides is fairly simple, consisting of only three components: sugar, phosphate, and base (Amos et al., 2002). The sequence of their bases, Adenine, Guanine, Cytosine, and Thymine, abbreviated as *A*, *G*, *C*, and *T*, respectively, distinguished these DNA structures. Hydrogen bonds connect these bases utilizing base-complementary rules, in which *A* pairs with *T*, *G* pairs with *C*, and vice versa. These pairing rules can be expressed as *a*, *g*, *c*, and *t* (Yusof et al., 2011).

Head (Păun, 1996) in 1987 introduced splicing systems as a mathematical model of the recombinant behaviour of double-stranded DNA (dsDNA) and the enzymes that cut and paste dsDNA. Restriction enzymes, which can be found naturally in bacteria, can cut DNA fragments at certain sequences known as restriction sites, whereas ligases can re-join DNA fragments with complementary ends (Amos et al., 2002). The mathematical model introduced by Head is made up of a finite alphabet V, a finite set of initial strings over the alphabet A, and a finite number of rules R that act on the strings through iterative cutting and pasting that generate new strings (Yusof et al., 2011). This splicing mechanism generates a language called a splicing language. All splicing languages with finite sets of axioms and rules have been proven to be regular which has the lowest generative power in Chomsky's Hierarchy (Kari & Kopecki, 2017). Several restrictions are imposed on splicing systems in order to increase the generative power of the languages generated by splicing systems (Hamzah et al., 2014). Restrictions on the use of rules, such as probability, group, weights and fuzzy have been introduced (Hamzah et al., 2014; Karimi et al., 2014; Nguyen et al., 2013; Santono et al. 2021; Turaev et al., 2012). Fuzzy is important in solving decision decision-making problems (Ahmad et al., 2020). These restrictions have one thing in common: they increase the language's generative power up to context-sensitive languages.

Restricted splicing systems can be considered as theoretical models for universally programmable DNA-based computers, which is significant in terms of DNA computing. However, obtaining languages with generative power equivalent to the Turing machine is still unsuccessful. In (Santono et al. 2022), he introduced the concept of bounded-addition fuzzy splicing systems; for each axiom, the truth values are associated with the string in a closed interval $[0, 1]$, and the truth value of a string z is obtained by applying a bounded-addition fuzzy operation to the truth values of strings x and y (Santono et al. 2023). A threshold language is defined as a subset of the language generated based on some cut-points in [0, 1].

The purpose of this study is to develop a graphical user interface to generate all the splicing languages generated by bounded-addition fuzzy splicing systems and their variants. An algorithm is developed using JAVA language and an integrated development environment for JAVA using Visual Studio Code software to replace the time-consuming manual computation of the languages generated by bounded-addition fuzzy splicing systems and their variants.

The following shows the breakdown of the paper's structure. Section 2 provides various key definitions and notations from formal language, splicing systems, and bounded-addition fuzzy splicing systems. Section 3 presents the algorithm for the graphical user interface. Section 4 provides the result and Section 5 concludes the research with a discussion on the overall findings.

2. Preliminaries

In this section, some prerequisites were covered by outlining the basic concepts and notations of the formal language and the splicing system theories that will be used later. More details can be referred to (Hopcroft et al., 2000; Rozenberg & Salomaa, 1997; Sarmin et al., 2010).

The following general notations are used throughout the paper. The term $\in$ denotes an element's membership in a set, whereas $\notin$ denotes the absence of set membership. The strictness of the inclusion is specified by $\subset$, while $\subseteq$ stands for inclusion. The symbol $\varnothing$ represents an empty set, while $|X|$ represents the cardinality of a set X.

The families of recursively enumerable, context-sensitive, context-free, linear, regular and finite languages were denoted by **RE, CS, CF, LIN, REG** and **FIN** respectively. For these language families, the next strict inclusions, named Chomsky hierarchy (see (Rozenberg & Salomaa, 1997)), holds:

$$\textbf{FIN} \subset \textbf{REG} \subset \textbf{LIN} \subset \textbf{CF} \subset \textbf{CS} \subset \textbf{RE.}$$

Next, a basic definition of a splicing system and a theorem on the family of languages generated by a splicing language are recalled.

Definition 1 (Head, 1987): A splicing system (EH) is a 4-tuple $\gamma = (V, T, A, R)$ where V is an alphabet, $T \subseteq V$ is a terminal alphabet, A is a finite subset of V^{+} and $R \subseteq V{*}\#V{*}\$V{*}\#V{*}$ is the set of splicing rules where * denotes the nonempty sequences and # denotes the pasting operation.

Theorem 1 (Păun et al., 1998): The relations in the following Table 1 hold, where at the intersection of the row marked with F_1 with the column marked with F_2, there appear either the family EH(F_1, F_2) or two families F_3, F_4 such that $F_3 \subset$ EH(F_1, F_2) $\subseteq F_4$.

Table 1. The family of languages generated by splicing systems.

$F_1 \backslash F_2$	**FIN**	**REG**	**LIN**	**CF**	**CS**	**RE**
FIN	**REG**	**RE**	**RE**	**RE**	**RE**	**RE**
REG	**REG**	**RE**	**RE**	**RE**	**RE**	**RE**
LIN	**LIN, CF**	**RE**	**RE**	**RE**	**RE**	**RE**
CF	**CF**	**RE**	**RE**	**RE**	**RE**	**RE**
CS	**RE**	**RE**	**RE**	**RE**	**RE**	**RE**
RE	**RE**	**RE**	**RE**	**RE**	**RE**	**RE**

Next, the definition of a bounded-addition fuzzy splicing system is presented (Eqs. (1) and (2)).

Definition 2 (Santono et al. 2021): A bounded-addition fuzzy splicing system is a 6-tuple $\gamma^{\oplus} = (V, T, A^{\oplus}, R, \mu, \oplus)$ where *V, T, R* are defined as for a usual extended splicing system, $\mu : V^{*} \times [0,1]$ is a fuzzy membership function, $A^{\oplus}$ is a subset of $V^{*} \times [0,1]$ such that

$$\sum_{i=1}^{n} \mu(x_i) \leq 1 \tag{1}$$

and $\oplus$ is a bounded-addition fuzzy operation on [0, 1] defined by:

$$\mu_{A+B} = \mu_A + \mu_B - \mu_A \mu_B. \tag{2}$$

A fuzzy bounded-addition operation is defined next (Eq. (3)).

Definition 3 (Santono et al. 2021): For strings with fuzzy $(x,\mu(x)),\ (y,\mu(y)),\ (z,\mu(z)) \in V^{*} \times [0,1]$ and $r \in R$ the fuzzy bounded-addition operation is defined as

$$[(x,\mu(x)),(y,\mu(y))] \text{ a }_{r}\ z(z,\mu(z))$$

if and only if $(x, y) \text{ a }_{r}\ z$ and $\mu(z) = \mu(x) \oplus \mu(y)$ is defined by:

$$\mu_{x+y} = \mu_x + \mu_y - \mu_x \mu_y \text{ where } \mu_x \mu_y \in \mu(x_i) \text{ and } x_i \in A^{\oplus}. \tag{3}$$

Thus, the fuzzy of the string $z \in V^*$ obtained by splicing operation on two strings $x, y \in V^*$ is computed by undergoing bounded-addition operation on their fuzzy membership values. The language generated by the iterative bounded-addition fuzzy splicing system is defined below (Eq. (4)).

Definition 4 (Santono et al. 2021): The language generated by an iterative bounded-addition fuzzy splicing system $\gamma^{\oplus} = (V, T, A^{\oplus}, R, \mu, \oplus)$ is defined as:

$$L_f(\gamma^{\oplus}) = \{x \in T \mid (x, \mu(x)) \in \sigma^*(A^{\oplus})\}. \tag{4}$$

3. The Algorithm in The Graphical User Interface

In this section, the design of the algorithm is explained. The set of string (DNA strand), rule (restriction enzyme) and threshold value (bounded-addition fuzzy restriction) are decided by the user. In this example of designing the algorithm, the rules used are in the form of (*a#d$c#a*) and (*a#l$a#l*). The first step begins with setting the rules (*a#d$c#a*) and the algorithm design as stated:

```
public class FuzzySplicingGUI {
   int total = 0;
   String input = "";
   JPanel stringsPanel;
   JTextField stringInput;
   JTextField threshold;
   JTextArea textarea;
   JComboBox<Integer> jComboBox;
   JComboBox<String> dropDownRule;
   List<FuzzySpliceString> spliceObjects = new ArrayList<FuzzySpliceString>();
   private final static String newline = "\n";
   private float thresholdValue;
   private int stepInput;
   HashMap<String, String[]> patternRegex = new HashMap<String, String[]>();
   private String rule1;
   private String rule2;
   private void initPatternRegex() {
      patternRegex.put("a#d$c#a", new String[] { "ad{1}", "ca{1}" });
      patternRegex.put("a#l$a#l", new String[] { "a.\\b", "a.\\b" });
      patternRegex.put("a#l$b#l", new String[] { "a.\\b", "b.\\b" });
      patternRegex.put("l#a$l#b", new String[] { "\\b.a", "\\b.b" });
   }
```

The same algorithm is used to construct the rule (*a#d$b#ad*). The next step is to generate the algorithm for the splicing operation which consists of the cutting and pasting operation. The algorithm is divided into three parts that is the initial splicing operation, followed by the iterative splicing operation and lastly the final splicing operation. The algorithm is defined below:

```
//applied rule 1
private boolean isRule1Applied(String input) {
        final String regexRule1 = rule1;
        final Pattern pattern = Pattern.compile(regexRule1);
        final Matcher matcher = pattern.matcher(input);
        while (matcher.find()) {
            if (matcher.group(0) != null) {
                return true;
            }


              }
        return false;
    }
return false;
}

//applied rule 2
private boolean isRule2Applied(String input) {
        final String regexRule1 = rule2;
        final Pattern pattern = Pattern.compile(regexRule1);
        final Matcher matcher = pattern.matcher(input);
        while (matcher.find()) {
            if (matcher.group(0) != null) {
                return true;
            }
        }
        return false;
    }
```

Next, the fuzzy membership values of the strings involved in the splicing operation are calculated and selected according to the threshold value decided by the user. The algorithm considering the fuzzy membership values is shown next.

```
private void algoCalculate(String input1, String noInput1, String input2, String noInput2) {
        if (isRule1Applied(input1) && isRule2Applied(input2)) {
            String output1 = splitStringInput1(input1);
            System.out.println("Slice String 1 from " + input1 + " to " + output1);
            String output2 = splitStringInput2(input2);
            System.out.println("Slice String 2 from " + input2 + " to " + output2);
            String result = output1 + output2;
            System.out.println(input1 + " + " + input2 + " = " + result);
            float resultCalculation = resultCalculator(noInput1, noInput2);
            System.out.println("a+b-ab=" + noInput1 + "+" + noInput2 + "-" + "(" + noInput1 +
      "*" + noInput2 + ")="
                    + resultCalculation);
            if (thresholdValue < resultCalculation) {
                FuzzySpliceString o = new FuzzySpliceString();
                o.setInput(result);
                o.setValue(resultCalculation);
                spliceObjects.add(o);
            }
        }
    }


            }
        }
        return false;
    }
```

Next, all the languages generated are stored and only the selected languages according to threshold values are displayed as output. The output will be generated by the algorithm as stated:

```
private JPanel createLegendButton(String input) {
        JButton button = new JButton("Run Submissions");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                String key = String.valueOf(dropDownRule.getSelectedItem());
                setRule(key);
                spliceObjects = new ArrayList<FuzzySpliceString>();
                String x = String.valueOf(jComboBox.getSelectedItem());
                stepInput = Integer.parseInt(x);
                thresholdValue = Float.parseFloat(threshold.getText());
                System.out.println(stepInput);
                String output = stringInput.getText();
                textarea.setText("");
                displayOutput(output, 1);
                String[] outputString = output.split(" ");
                if (outputString.length == 4) {
                    if (stepInput >= 2) {
                     calculation(outputString[0], outputString[1], outputString[2], outputString[3], 2);
                    }
                } else {
```

```
          for (String v : outputString) {
            System.out.println(v);
          }
        }
      }
    });
```

Finally, a graphical user interface is developed. In the interface, the string and threshold values are decided by the user. The user can also choose the rule and the number of iterations of the splicing operation. The algorithm for developing the interface is shown next.

```
 public FuzzySplicingGUI() {
    initPatternRegex();
    JFrame frame = new JFrame();
    frame.setLayout(new GridLayout(0, 1));
    stringInput = setTextField();
    stringsPanel = createLegend("Strings", stringInput);
    frame.add(stringsPanel);
    threshold = setTextField();
    frame.add(createLegend("Threshold", threshold));
    frame.add(createLegendDropDown("Number of Steps"));
    frame.add(legendDropDownRule("Rules Applied"));
    // frame.add(createLegendStaticText("Rules Applied"));
    frame.add(createLegendButton("Execution"));
    JScrollPane scrollPane = createTextArea();
    GridBagConstraints c = new GridBagConstraints();
    c.fill = GridBagConstraints.BOTH;
    c.weightx = 1.0;
    c.weighty = 1.0;
    frame.add(scrollPane, c);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setTitle("Splicing GUI");
    frame.setSize(500, 800);
    frame.setVisible(true);
```

4. Results

A graphical user interface is constructed for the algorithm developed using JAVA language and an integrated development environment for JAVA using Visual Studio Code software in order to replace the time-consuming manual calculation and also it is user-friendly. In the interface, the string (DNA strand), rule (restriction enzyme) and threshold value (bounded-addition fuzzy restriction) are decided by the user. The procedure of using the graphical user interface is straightforward as it is user-friendly. The steps and figures for generating bounded-addition fuzzy DNA splicing languages using the interface are shown next.

Step 1: The initial graphical user interface of the bounded-addition fuzzy DNA splicing system is illustrated in Figure 1.

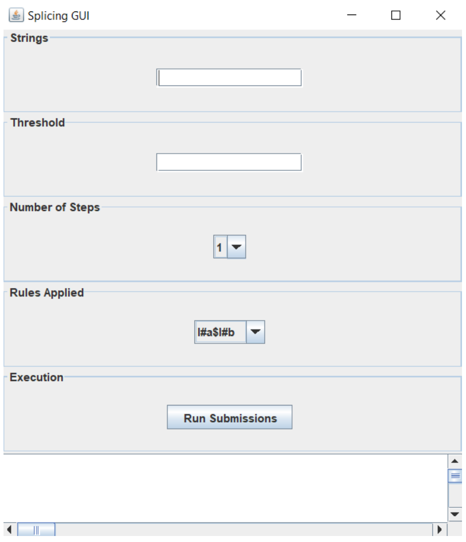


Figure 1. Initial graphical user interface.

Step 2: Fill in the strings with fuzzy membership value, threshold value and select the rule. The image is illustrated in Figure 2.

For this example, the strings inserted are *dad* and *cad* with fuzzy values of 0.3 and 0.7 respectively. The threshold value is decided to be 0.01 which means that all the strings generated should have fuzzy membership value greater than 0.01. The rule chosen for this example is (*a#d$c#a*) as this rule can be applied to the strings inserted earlier.

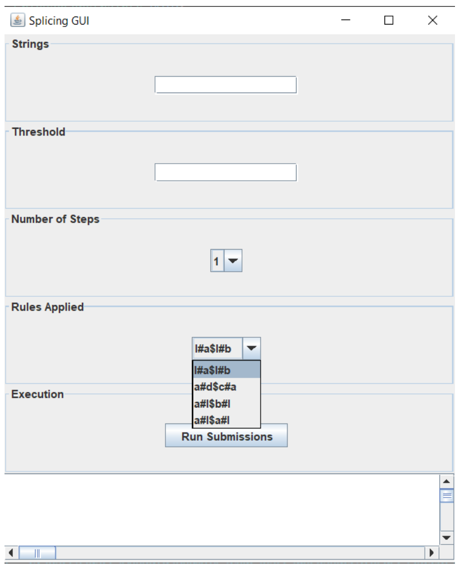
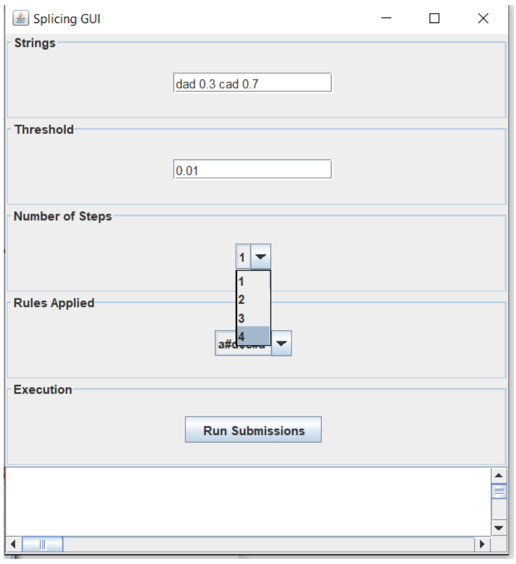


Figure 2. The image after inputs are filled/ selected.

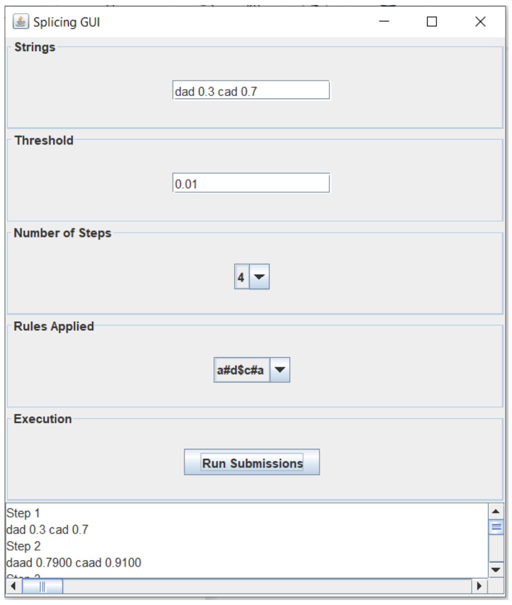
Step 3: Next, the number of iterations of the bounded-addition fuzzy DNA splicing operation is selected as illustrated in Figure 3. In this example, the number of iterations chosen is four.



Strings: dad 0.3 cad 0.7
Threshold: 0.01
Number of Steps: 1
1
2
3
4
Rules Applied: a#d$c#a
Execution

Figure 3. The image of selecting the number of iterations for the bounded-addition fuzzy DNA splicing operation.

Step 4: The splicing languages are generated by pressing the *Run Submissions* button as illustrated in Figure 4. The output will be generated in the interface for the user's view. In this example, the output in Step 1 is the original strings with their fuzzy membership values. The following steps list down all the languages generated with fuzzy membership values greater than 0.01 which satisfy the threshold value (fuzzy restriction) as required by the user.



Strings: dad 0.3 cad 0.7
Threshold: 0.01
Number of Steps: 4
Rules Applied: a#d$c#a
Execution

```
Step 1
dad 0.3 cad 0.7
Step 2
daad 0.7900 caad 0.9100
```

Figure 4. The image of the generated splicing languages.

This graphical user interface also enables the user to view all the languages generated by the respective bounded-addition fuzzy DNA splicing system. To do so, the user can choose to view all the splicing languages generated by setting the threshold value to 0.

5. Conclusion

In conclusion, a graphical user interface for bounded-addition fuzzy DNA splicing systems and their variants is generated using the JAVA programming language and the Visual Studio Code software's integrated development environment for JAVA. This graphical user interface is user-friendly as the user can select the string, rule, and threshold value that correspond to the DNA strand and restriction enzyme. This interface also allows the user to select the number of iterations for the bounded-addition fuzzy DNA splicing systems, as well as to view all the languages produced by the bounded-addition fuzzy DNA splicing systems and their variants. In the future, designing a graphical user interface with a different operation such as multiplication can be explored.

Acknowledgement

The first and second authors would like to thank the Ministry of Higher Education and Universiti Teknologi MARA Malaysia for the financial funding through the research grant FRGS-RACER (600-IRMI/FRGS-RACER 5/3 (050/2019)). The third and fourth authors would also like to thank the Ministry of Higher Education and Research Management Center (RMC) UTM for the UTMShine Grant Vote No. Q.J130000.2454.09G89.

Funding

The authors received funding from Universiti Teknologi MARA and Universiti Teknologi Malaysia through the FRGS-RACER grant and UTMShine Grant respectively.

Author Contribution

Author 1 prepared the introduction and literature review. Author 2 wrote the research methodology and designed the algorithm of the interface and interpreted the results. Author 3 and author 4 conclude the result and oversaw the article writing.

Conflict of Interest

The authors have no conflicts of interest to declare.

References

Ahmad, S. A. S., Mohamad, D., Azman, N. I. (2020). Similarity based fuzzy inferior ratio for solving multicriteria decision making problems. *Malaysian Journal of Computing, 5* (*2*), 597-608.

Amos, M., Pun, G., Rozenberg, G., & Salomaa, A. (2002). Topics in the Theory of DNA Computing. *Theoretical Computer Science*, *287*(1), 3–38. https://doi.org/10.1016/S0304-3975(02)00134-2

Hamzah, N. Z. A., Mohd Sebry, N. A., Fong, W. H., Sarmin, N. H., & Turaev, S. (2014). Splicing Systems over Permutation Groups of Length Two. *Malaysian Journal of Fundamental and Applied Sciences*, *8*(2), 83–88. https://doi.org/10.11113/mjfas.v8n2.127

Head, T. (1987). Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, *49*(6), 737–759. https://doi.org/10.1007/BF02481771

Hopcroft, J. E., Motwani, R., Rotwani, & Ullman, J. D. (2000). *Introduction to Automata Theory, Languages and Computability*.

Kari, L., & Kopecki, S. (2017). Deciding whether a regular language is generated by a splicing system. *Journal of Computer and System Sciences*, *84*, 263–287. https://doi.org/10.1016/j.jcss.2016.10.001

Karimi, F., Turaev, S., Sarmin, N. H., & Fong, W. H. (2014). Fuzzy splicing systems. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8733*, 20–29. https://doi.org/10.1007/978-3-319-11289-3_3

Nguyen, N. T., Trawiński, B., Katarzyniak, R., & Jo, G. S. (2013). Probabilistic Splicing Systems. *Studies in Computational Intelligence*, *457*(January). https://doi.org/10.1007/978-3-642-34300-1

Păun, G. (1996). On the Splicing Operation. *Discrete Applied Mathematics*, *70*(1), 57–79. https://doi.org/10.1016/0166-218X(96)00101-1

Păun, G., Rozenberg, G., & Salomaa, A. (1998). DNA computing: New computing paradigms. *Computers & Mathematics with Applications*, *37*(3), 134. https://doi.org/10.1016/s0898-1221(99)90411-x

Rozenberg, G., & Salomaa, A. (1997). Handbook of Formal Languages. In *Handbook of Formal Languages* (Issue January). https://doi.org/10.1007/978-3-642-59126-6

Santono, Mohd Pawiro; Selvarajoo, Mathuri; Heng, Fong Wan; Sarmin, N. H. (2021). Some Properties of Bounded-Addition Fuzzy Splicing Systems. *Kalahari Journals*, *6*(3), 2698–2705.

Santono, Mohd Pawiro; Selvarajoo, Mathuri; Heng, Fong Wan; Sarmin, N. H. (2022). Bounded-Addition Fuzzy Simple Splicing Systems. *Journal of Algebraic Statistics*, *13*(2), 2079–2089.

Santono, Mohd Pawiro; Selvarajoo, Mathuri; Heng, Fong Wan; Sarmin, N. H. (2023). The Properties of Bounded-Addition Fuzzy Semi-Simple Splicing Systems. *ICMSS 2022*, *6*(3), 354–363.

Sarmin, N. H., Yusof, Y., & Wan Heng, F. (2010). Some characterizations in splicing systems. *AIP Conference Proceedings, 1309*(December), 411–418. https://doi.org/10.1063/1.3525142

Turaev, S., Gan, Y. S., Othman, M., Sarmin, N. H., & Fong, W. H. (2012). Weighted splicing systems. *Communications in Computer and Information Science, 316 CCIS*, 416–424. https://doi.org/10.1007/978-3-642-34289-9_46

Yusof, Y., Sarmin, N. H., Goode, T. E., Mahmud, M., & Heng, F. W. (2011). An Extension of DNA Splicing System. *Proceedings - 2011 6th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2011*, 246–248. https://doi.org/10.1109/BIC-TA.2011.67