Jurnal Teknologi

Full paper

Modelling the Behaviour of Single Stage Splicing Language: A Yusof Goode Computational Approach

Wen Li Lim,* Yuhani Yusof, Norhayati Rosli, Mohammad Hassan Mudaber

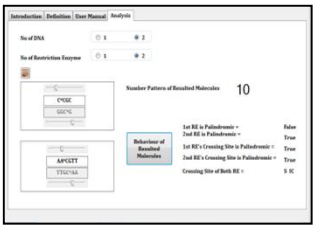
Faculty of Industrial Sciences & Technology, Universiti Malaysia Pahang, 26300 UMP Gambang, Kuantan, Pahang

*Corresponding author: wlilim@yahoo.com

Article history

Received: 05 October 2014
Received in revised form:
21 November 2014
Accepted: 1 February 2015

Graphical abstract



Abstract

Yusof-Goode (Y-G) splicing system is a formal characterization of the generative capacity of specified enzymatic activities acting on DNA molecules with new extension symbolization of representing rule. The output of Y-G splicing system can be categorized into three types of single stage splicing language namely active persistent, transient and inert persistent language. It is both money and time consuming to conduct laboratory experiments to determine the behaviour of splicing language. Hence, research has been conducted to predict the characteristic of single stage splicing language based on limit adjacency matrix computational modelling in order to optimize time and money. The utilization of software programming has been developed through Visual Basic Software for scientists to determine the behaviour of single stage splicing language as well as the number types of resulted DNA molecules restricted to at most two strings and two rules with one cutting site. The output from the program was found to match the outcomes of wet lab experiments.

Keywords: Yusof-Goode splicing system; splicing language; single stage limit language.

Abstrak

Sistem hiris-cantum Yusof-Goode (Y-G) adalah satu pencirian rasmi kepada kapasiti penjana bagi aktiviti-aktiviti enzim pembatas tertentu ke atas molekul-molekul DNA dengan penyimbolan lanjutan baru dalam mempersembahkan peraturan. Hasil daripada sistem hiris-cantum Y-G boleh dikategorikan kepada tiga jenis bahasa hiris-cantum peringkat tunggal iaitu bahasa sementara, bahasa lengai dan bahasa berterusan aktif. Duit dan kekangan masa untuk menjalankan ujikaji makmal dalam menentukan ciri-ciri bahasa hiris-cantum peringkat tunggal. Oleh itu, penyelidikan telah dijalankan bagi meramal ciri-ciri bahasa hiris-cantum peringkat tunggal berdasarkan model berkomputer matrik had bersebelahan bagi mengoptimumkan masa dan duit. Penggunaan pengaturcaraan perisian telah dibangunkan melalui Perisian Visual Basic supaya ahli-ahli sains dapat menentukan tingkah laku bahasa hiris-cantum peringkat tunggal dan juga bilangan jenis molekul-molekul DNA yang dihasilkan terhad kepada paling banyak dua jujukan dan dua peraturan dengan satu belah pemotongan. Keputusan daripada program didapati sepadan dengan hasil ujikaji di makmal.

Kata kunci: Sistem Hiris-Cantum yusof-goode; bahasa hiris-cantum; bahasa had peringkat tunggal

1.0 INTRODUCTION

Splicing system was primarily introduced by Head [1] as a formal characterization of the generative capacity of specified enzymatic activities acting on initial deoxyribonucleic acid (DNA) molecules via the framework of Formal Language theory. There have been many developments of splicing system such as Head [1], Goode-Pixton [2] and Yusof-Goode (Y-G) [3] models. All these models can perform universal computation, which means it can generate the entire family of recursively enumerable languages in Chomsky hierarchy of formal languages [1, 2, 3]. Y-G model is applied in the computational model for this paper as this new splicing system was based on the characteristics of the restriction enzyme itself, which presents the translucent behaviour of DNA biological process.

Head [1] has defined splicing language as the language generated by a splicing system. In order to analyze the outcome of a splicing system, which is closer to the observation in the laboratory, Goode and Pixton [4] has formally defined limit languages, the molecules that are left after the biochemical reaction has run to completion. The molecules that do not participate in further splicing at this final stage are called inert languages. In the same paper, the authors defined transient languages in the sense that the molecules are used up in the system at equilibrium. Then, Yusof [3] introduced the molecules that can participate in further splicing at terminal stage, and the corresponding formal language defined by the splicing system is named as active persistent languages.

In this paper, the behaviour of single stage splicing language of Y-G splicing system involving at most two strings and two rules with one cutting site is modeled using limit adjacency matrix

similar to that presented in [6]. The main difference is in this paper, the behaviour of Y-G splicing system is approached by programming the linear adjacency matrix in Visual Basic software. Results are then compared with the lab experiments [3, 5, 7].

■2.0 COMPUTATIONAL MODEL GENERATION

2.1 Assumptions

In this paper, the following assumptions are followed:

1. A finite initial number of different double-stranded DNA (dsDNA) sequences and a finite number of enzymes are generated.
2. The probability of digestion and ligation efficiency is assumed to be equal.
3. Restriction enzymes can cut DNA molecules at specific recognition sites, producing molecules with only sticky ends.
4. Each DNA strands has exactly one recognition site for one restriction enzyme and they do not contain the sites for the other enzymes.
5. The DNA strands are linear.
6. The DNA strands are assumed to be dephosphorylated on its 5 ' ends to avoid blunt end ligation

The behaviour of single stage splicing languages is analyzed by using limit adjacency matrix method [6] while Y-G model [3] is used to compute the number types of single stage splicing language.

2.2 Preprocessing And Importing Into Visual Basic

2.2.1 Input Parameters

A Visual Basic software [8] programming script has been developed to predict the number of resulted molecules. The software requires the user to enter the number of DNA strings and restriction enzymes as inputs. Then, the user will need to enter the sequences of the DNA strings, that is alphabet *A* representing Adenine, alphabet *T* representing Thymine, alphabet *C* representing Cytosine and alphabet *G* representing Guanine. The base pair of the entered DNA string will be generated correspondingly. The user can then choose the cutting site of the DNA string by sliding the bar beneath the entered and computer generated DNA strings. After all required inputs have been keyed into the fields, the software will generate the number patterns of strings produced as the description in the next section.

2.2.2 Computing Splicing Language

Given a DNA string, the software will generate a finite set of strings based on the entered number and configuration of DNA strings with cutting site. Also, the software will create another dummy DNA string, which is 180 degrees rotated to the initial DNA string. These DNA strings will then be cut at the crossing site to produce several pieces of DNA fragments with sticky ends. These DNA fragments will be ligated to search for all possible combinations based on their base pairs, which in Formal Language Theory terms are called splicing languages. The computed splicing languages are compared with the wet lab experiment.

2.2.3 Analyzing the Behaviour of Splicing Language

Each splicing language is analyzed according to a limit adjacency matrix [6]. $n \times n$ matrix is formed if there exists n number of splicing languages. All splicing languages will then be cut and ligated again at the crossing site, if they present, to produce other DNA strings, which are represented by 1 in a limit adjacency matrix. These cutting and recombination behaviour will continue until the resulted strings do not contain any crossing site. These strings, w_i, when no cutting is possible are called inert limit language, which are represented by a row of zeros in a limit adjacency matrix as follows:

$$w_i \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \text{---- (1)}$$

Based on the above mentioned script which simulates the recombinant behaviour of DNA strings, the result is shown in a matrix form and being analyzed by using limit adjacency matrix method [6]. Each DNA strings and restriction enzyme with different characteristic are simulated to obtain the results. Flowchart in **FIGURE 1** shows the operations of cutting and pasting as well as the generating limit language operations.

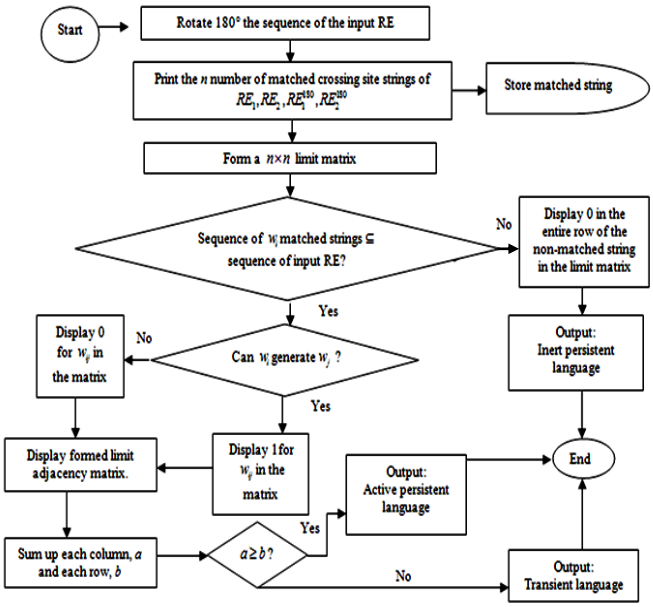
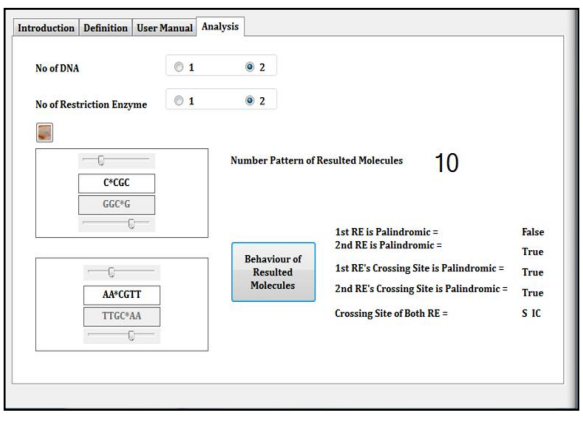


Figure 1 Flowchart on determining the behaviour of splicing language

■3.0 RESULTS AND DISCUSSION

Two runs on the script have been performed based on the enzymes used in wet lab experiments [3, 5, 7]. **FIGURE 2** and **FIGURE 3** illustrate the results of the runs. These results are then summarized in **TABLE 1** along with the outcome from the wet lab experiments.



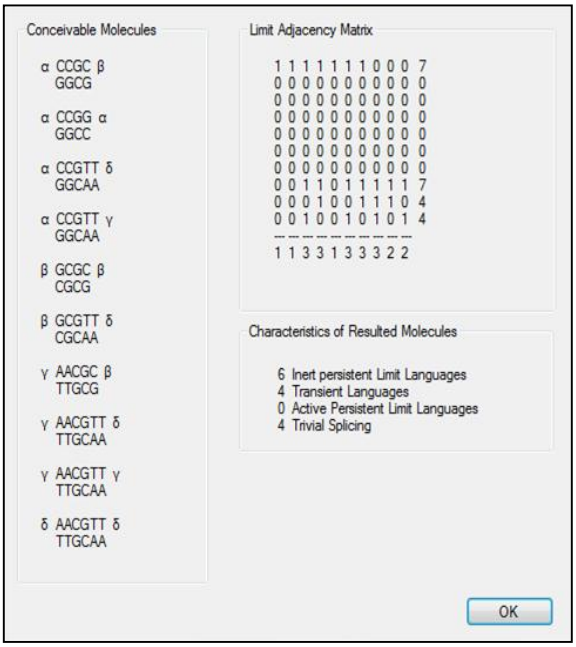


Figure 2 Results obtained from visual basic: modeling the behaviour of splicing languages with two dna and two restriction enzymes, *bgl*i and *dra*iii

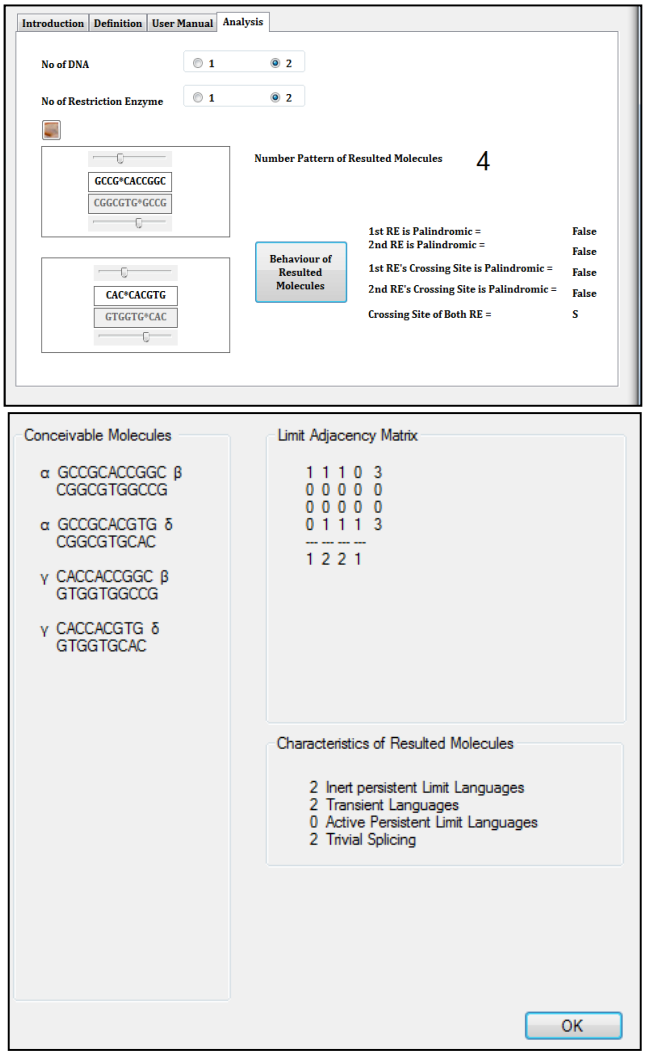


Figure 3 Results obtained from visual basic: modeling the behaviour of splicing languages with two dna and two restriction enzymes, *aci*i and *Acl*I

Table 1 Comparison of the results from wet lab experiments and software

Item	Number of Strings	Number of Rules	Enzymes Used	Lab Results	Software Results
1	2	2	*Bgl*I and *Dra*III [5, 7]	The two initial strings disappeared. Both adult strands increased throughout the duration of the reaction	2 inert persistent languages, 2 transient languages
2	2	2	*Aci*I and *Acl*I [3]	Existence of six inert persistent languages, three active persistent languages and one transient languages	6 inert persistent languages, 4 transient languages

As shown in item 1 of **TABLE 1**, the results from the software are consistent with the outcome of the wet lab experiment. Adult strings are renamed to inert persistent language in [3] and the strings that disappeared have been formally defined by Goode and Pixton in [4] as transient language. This is to be expected since the operations of both software and experiment are similar. The initial set of DNA strings was cleavage at the crossing site and relegated into the predicted set of adult strings. The initial strings gradually disappear since they are being consumed to produce adult strings and no further recombination back to the initial strings.

For item 2 of **TABLE 1**, there is no active persistent language according to the results from the software. To compare the results with the wet lab experiment done which shows the existence of six inert persistent languages, the software successfully predicted the result. Besides, one molecule is shown in wet lab experiment as transient, which is similar to the results provided by the software. However, there are three molecules presented as active persistent language in [3], which is different from the predicted result in the software above. From the conclusion in [3], it is due to the quantity of strings during the experiment yet the prediction in the software above is ignoring the possibility of unbalanced numbers of molecules available for various reactions by the definition of limit language. Nevertheless, limit adjacency matrix in **FIGURE 2** shows that $\sum_{r}^{10} a_{9,r} = \sum_{r}^{10} a_{10,r} = 4 > \sum_{r}^{10} a_{r,9} = \sum_{r}^{10} a_{r,10} = 2,$ $\sum_{r}^{10} a_{8,r} = 7 > \sum_{r}^{10} a_{r,8} = 3$, but $\sum_{r}^{10} a_{1,r} = 7$ $\sum_{r}^{10} a_{r,1} = 1$. Hence, the later string is certainly transient but the other three strings can be an ambiguous case to stay in between active persistent and transient language depending on the quantity of initial strings. Therefore, the software simulates the DNA splicing process in [3, 5, 7] in determining the behaviour of splicing languages.

4.0 CONCLUSION

The computational results from the software prove that the behaviour of splicing languages is predictable after comparing with the wet splicing systems and theorems. In conclusion, the conceivable molecules that the program provides do behave as theorems predict.

Acknowledgement

The authors gratefully acknowledge Ministry of Education (MOE) and Research and Innovation Department, Universiti Malaysia Pahang (UMP) for the financial funding through UMP Research Grant Vote No: RDU 130354 and RAGS Grant Vote No: RDU131404.

References

[1] 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.
[2] Laun, T.E.G. 1999. *Constants and Splicing Systems. State University of New York at Binghamton*. Ph.D. Thesis.
[3] Yusof, Y. 2012. *DNA Splicing System Inspired by Bio Molecular Operations*. Ph.D. Thesis, Universiti Teknologi Malaysia.
[4] Goode, E. and Pixton, D. 2004. Splicing to The Limit. *Lecture Notes in Computer Science*. 2950: 189–201.
[5] Lim, D.S.F. 2006. *Splicing Systems and Language*. MSc. Dissertation. Universiti Teknologi Malaysia.
[6] Lim, W.L. and Yusof, Y. Modeling Limit Languages via Limit Adjacency Matrix and Yusof-Goode Approaches. 2014. *Proceeding of International Conference of Mathematics, Engineering and Industrial Applications 2014 (ICoMEIA).* in press.
[7] Laun, E. and Reddy, K.J. 1999. Wet Splicing Systems. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. 48: 73–83.
[8] Visual Basic, Microsoft.NET, 2010