Available online at
http://www.ibnusina.utm.my/jfs

Journal of
Fundamental
Sciences

Article

Recognition of Simple Splicing Systems using *SH*-Automaton

Fong Wan Heng[1,*], Nor Haniza Sarmin[2], Zuwairie Ibrahim[3]

[1]*Ibnu Sina Institute for Fundamental Science Studies, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor.*
[2]*Department of Mathematics, Faculty of Science, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor.*
[3]*Center for Artificial Intelligence and Robotics (CAIRO), Department of Mechatronics and Robotics, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia.*

** Author to whom correspondence should be addressed; E-mail: fwh@ibnusina.utm.my*

Received 24 October 2008
http://dx.doi.org/10.11113/mjfas.v4n2.41

ABSTRACT

Splicing language is the language which results from a splicing system. Splicing system was first introduced by Tom Head in 1987 as the mathematical model of systems of restriction enzymes acting on initial DNA molecules. Splicing languages are closely related to automata theory. Simple splicing systems can be recognized by *SH*-automata diagrams due to the regularity of splicing languages. *SH*-automaton defines exactly one language which is the language generated by the simple splicing system. In this paper, the concept of firm and maximal firm subwords are introduced. Some examples are then given to illustrate the maximal firm subwords of a word in a simple splicing system. Taking the *SH*-automata concept, which is a short compact way of encoding normal non-deterministic automata in the special case of *SH* systems, the maximal firm subwords of the initial words of an *SH* systems serve as the labels for the associated *SH*-automaton. Some examples which will show the maximal firm subwords of the words in the initial set *I*, the regular expression for the language generated by the given splicing system and the simplest non-deterministic automaton that recognizes the corresponding splicing system are also given

| Splicing system | Splicing language | *SH*-automaton | Maximal firm subwords | Regular expression |

1. Introduction

Splicing system was developed as a mathematical model of systems of restriction enzymes acting on initial DNA molecules [1]. DNA molecules are made up of nucleotides, and these nucleotides differ from each other by their bases, namely Adenine (A), Cytosine (C), Guanine (G) and Thymine (T) respectively. DNA molecules can be cut by restriction enzymes at particular sequence of the nucleotides. After cutting, the resulting fragments can be pasted together by a ligase, thus forming a molecule of recombinant DNA. The set of double-stranded DNA molecules that may arise from a splicing system is called the splicing language. The definitions of a splicing system and a splicing language are stated in the following.

Definition 1. [1] (Splicing System and Splicing Language)
A *splicing system* $S = (A, I, B, C)$ consists of a finite set of alphabet A, a finite set of initial strings I 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. $L(S)$ is the language generated by a splicing system S which consists of the strings in I and all strings that can be obtained by adjoining the words *ucxfq* and *pexdv* 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 $L = L(S)$.

A particular type of splicing system discussed in this paper is the simple splicing (*SH*) system. The formal definitions of a simple splicing system and a simple splicing language are given in the following.

Definition 2. [2] (Simple Splicing System and Simple Splicing Language)

Let $S = (A, I, R)$ be a splicing system in which all rules in R have the form (a, 1; a, 1), where a is in A. Then S is called a *simple splicing system*. A splicing language L is said to be a *simple splicing language* if L can be generated by a simple splicing system.

Splicing languages are regular, but not all regular languages are splicing languages [3]. The definition of a regular language is given in the following.

Definition 3. [4] (Regular)

A language L is called *regular* if and only if there exist a deterministic finite accepter M such that $L = L(M)$.

For the definition of regular language, deterministic finite accepter is used instead of deterministic finite automaton since deterministic finite automaton [5] is also known as deterministic finite accepter in [4]. Therefore, since simple splicing languages are regular, they can be recognized by automata diagrams. Labels for an *SH*-automaton are actually the maximal firm subwords of a word. In the next section, the concepts of firm and maximal firm subwords of a word are discussed.

2. Maximal Firm Subwords

For the simple splicing language L we say that a word w in L is *firm* if it contains no occurrence of a letter in the rule set R. Recall that by a factor of a word w in A^* meant any word y, there are words x and z for which $w = xyz$, where x, z in A^*. So a *maximal firm subword* of a word w should be a factor y of w in which no letter in R occurs in y but if $w = uayz$ with a in A, then a is in R and if $w = xybv$ with b in A, then b is in R.

If $w = uayz$ and a is not in R, then ay would be a firm factor of w that is one letter longer than y and so y would not be a maximal firm subword of w. If $w = xybv$ and b is not in R, then yb would be a firm factor of w that is one letter longer than y and so y would not be a maximal firm subword of w. A word y is a maximal firm subword of w as long as y can get, and stay, inside w. In other words, a maximal firm subword of w is the longest possible factor of w that contains no letter in R.

A word w in A^* that contains no letter in R cannot be cut at all, thus it is considered to be firm. If $w = xry$ with neither x nor y null then w can be cut into two pieces and should therefore be not firm. For simple splicing languages, a word w is firm if and only if it contains no element of R.

Some examples to illustrate the maximal firm subwords of a word in a simple splicing system are given next. They differ in terms of the number of rules and initial strings involved. Example 1 involves a splicing system with only one rule with the initial set I left unspecified.

Example 1.
Let $A = \{a, b\}$, $B = \{b\}$, I will be left unspecified. There is only one rule, namely $r = (b, \lambda, b, \lambda)$. A word w in A^* is firm with respect to r if (and only if) w is in a^*. The maximal firm subwords of the word *aabaaabaaba* are indicated via underscores: *aabaaabaaba*.

In the next example, maximal firm subwords of a word in a splicing system having two rules with initial set I unspecified are given.

Example 2.
Let $A = \{a, b, c, d, e\}$, $B = \{b, d\}$, I will be unspecified. There are only two rules, namely $r = (b, \lambda, b, \lambda)$ and $r' = (d, \lambda, d, \lambda)$. These two rules can be abbreviated as b and d respectively. A word w in A^* is firm with respect to r if w in $\{a, c, d, e\}^*$. A word w in A^* is firm with respect to r' if w in $\{a, b, c, e\}^*$. A word w in A^* is firm with respect to $R = \{r, r'\}$ if and only if w is in $\{a, c, e\}^*$. The maximal firm subwords of the word *aabacadcccbceeab* are indicated via underscores: *aabacadcccbceeab*.

Example 3 and Example 4 show the maximal firm subwords of words in a splicing language L generated by a splicing system involving a specified set I, one and two rules respectively. The language generated is listed in recursive form and the maximal firm subwords of words in that language are indicated via underscores.

Example 3.
Let $A = \{a, b\}$, $B = \{b\}$ and $I = \{aaabaa, aba\}$. The language generated by (A, B, I) is $L = \{aaabaa, aba, aaaba, abaa\}$. The maximal firm subwords of words in L are indicated via underscores: *aaabaa*, *aba*, *aaaba*, *abaa*.

Example 4.
Let $A = \{a, b, c, d, e\}$, $B = \{b, d\}$ and $I = \{aabcedccbee\}$. The language generated by (A, B, I) is $L' = \{aab(cedccb)^*ee\}$. The maximal firm subwords of words in L' are indicated via underscores: *aab*(*ced ccb*)**ee*.

Example 5 involves a more complicated splicing system involving two rules and two initial strings.

Example 5.
Let $A = \{a, b, c, d, e\}$, $B = \{b, d\}$ and $I = \{abcde, edcba\}$. The language generated by (A, B, I) is $L'' = \{a(bcdc)^*ba, e(dcbc)^*de, a(bcdc)^*bcde, e(dcbc)^*dcba\}$. The maximal firm subwords of words in L'' are indicated via underscores: *a*(*bcdc*)**ba*, *e*(*dcbc*)**de*, *a*(*bcdc*)**bcde*, *e*(*dcbc*)**dcba*.

In order to visualize and represent finite automata, *transition diagrams* which involve vertices and edges are used [4]. An automaton recognizes a language L if each string accepted by the automaton is in L and each string in L is accepted by the automaton [6]. The simple splicing automata (*SH*-automata) concept is explained in the following.

3. *SH*-automata Concept

Taking the *SH*-automata concept, which is a short compact way of encoding normal non-deterministic automata in the special case of *SH* systems, the maximal firm subwords of the initial words of an *SH* systems serve as the labels for the associated *SH*-automaton.

One trip through the *SH*-automaton is an arbitrary walk through the graph entering via an entrance following any path until exit. The language is an infinite set. We can generate a lot of different words from an *SH*-

automaton, but *SH*-automaton defines exactly one language which is exactly the language generated by the simple splicing system.

Some examples are given in the following which will show the maximal firm subwords of the words in the initial set *I*, the regular expression for the language generated by the given splicing system and the simplest non-deterministic automaton that recognizes the corresponding splicing system. The examples differ in the number of initial strings and rules involved.

Example 6 involves a splicing system with only one string in *I* and one rule in *R*.

Example 6.
Suppose $A = \{a, b\}$, $I = \{aabaaaaabaaabaa\}$ and $R = \{b\}$. The maximal firm subwords of the only word in *I* are *aa*, *aaaaa*, *aaa*. The regular expression for this language is $aab(aaab+aaaaab)^*aa$. The simplest non-deterministic automaton that recognizes $L(A, I, R)$ is given in Figure 1.

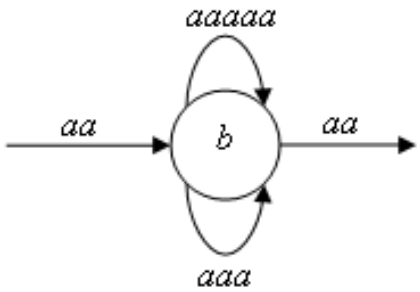

Figure 1: Automaton for Example 6.

Example 7 involves a splicing system with two strings in *I* and one rule in *R*.

Example 7.
Suppose $A = \{a, b\}$, $I = \{abaaabaaabaa, aabaaaaabaaaaabaaaaa\}$ and $R = \{b\}$. The maximal firm subwords of *I* are *a*, *aaa*, *aa*, *aaaaa*. The regular expression for this language is $(a+aa)b(aaab+aaaaab)^*(aa+aaaaa)$. Notice that we will get the same language generated if *I* were $\{abaaabaa, aabaaaaabaaaaa\}$ or if *I* were $\{abaaaaa, aabaaaaabaaabaa\}$. The simplest non-deterministic automaton that recognizes $L(A, I, R)$ is given in Figure 2.

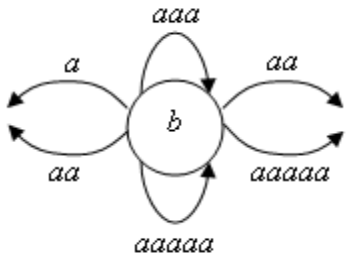

Figure 2: Automaton for Example 7.

The following example involves a splicing system with two strings in I and also two rules in R.

Example 8.
Suppose $A = \{a, b, c\}$, $I = \{abaacaaaaaba, abaaabaaacaaba\}$ and $R = \{b, c\}$. The maximal firm subwords of I are a, aa, $aaaaa$, aaa. The regular expression for this language is $ab[aaab+(aac+aaac)(aab+aaaaab)]^*a$. The simplest non-deterministic automaton that recognizes $L(A, I, R)$ is shown in Figure 3.

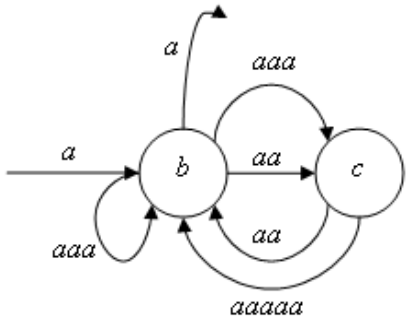

Figure 3: Automaton for Example 8.

Example 9 involves a splicing system with one string in I and one rule in R, where the symbols in R may appear adjacent to each other in I.

Example 9.
Suppose $A = \{a, b\}$, $I = \{abaaabbaabbba\}$ and $R = \{b\}$. The maximal firm subwords of I are a, aaa, aa. The regular expression for this language is $ab(aaab+b+aab)^*a$. Note that $(a+b)^* = A^*$; a^*b^* are those words in which all occurrences of an 'a' come before all occurrences of 'b'. The simplest non-deterministic automaton that recognizes $L(A, I, R)$ is given in Figure 4.

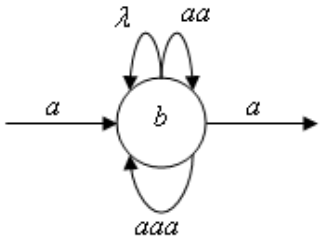

Figure 4: Automaton for Example 9.

Therefore, simple splicing systems can be recognized by *SH*-automaton using transition diagrams as illustrated in this section.

4. Conclusion

In this paper, the concept of maximal firm subwords is introduced. The maximal firm subwords of the initial words of a simple splicing system are shown to serve as the labels for the associated *SH*-automaton. Also,

SH-automaton defines exactly one language which is exactly the language generated by the simple splicing system.

5. Acknowledgement

We would like to express our gratitude to Prof. Tom Head from State University of New York at Binghamton, Binghamton, New York, USA, for his collaboration in this research. We would also like to thank the Ministry of Science, Technology and Innovation (MOSTI) Malaysia and the Research Management Centre (RMC), UTM for the financial funding through e-Science Fund Vote No 79081.

6. References

[1] T. Head, Formal Language Theory and DNA: An Analysis of the Generative Capacity of Specific Recombinant Behaviors, Bulletin of Mathematical Biology, 49 (1987) 737-759.
[2] E. G. Laun, Constants and Splicing Systems, Ph.D. Thesis, State University of New York at Binghamton, 1999.
[3] R. W. Gatterdam, Splicing Systems and Regularity, International Journal of Computer Math, 31 (1989) 63-67.
[4] P. Linz, An Introduction to Formal Languages and Automata, third ed., Jones and Bartlett Publishers Inc, USA, 2001.
[5] D. Kelley, Automata and Formal Languages, Prentice-Hall Inc, USA, 1995.
[6] M. V. Lawson, Finite Automata, Chapman & Hall/CRC, USA, 2004