Prosiding Simposium Kebangsaan Sains Matematik ke -16
3-5 Jun 2008

REDUCTION OF SPLICING SYSTEMS USING SOLID CODES

Fong Wan Heng
Ibnu Sina Institute for Fundamental Science Studies,
Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor.
fwh@ibnusina.utm.my

Nor Haniza Sarmin
Department of Mathematics, Faculty of Science,
Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor.
nhs@mel.fs.utm.my

Zuwairie Ibrahim
Center for Artificial Intelligence and Robotics (CAIRO),
Department of Mechatronics and Robotics, Faculty of Electrical Engineering,
Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor.
zuwairiee@fke.utm.my

Abstract:

Splicing system was originally developed by Tom Head in 1987 as the generative capacity of systems of restriction enzymes acting on DNA molecules. It can be modelled mathematically using Formal Language Theory. In this paper, some new concepts of solid codes are presented with some examples. For an S_kH system, where k is the length of the longest word in a rule R, it may be reduced to a simple splicing system if R is a solid code. Two examples are given to show that splicing systems having solid rules can be reduced to simple splicing systems by replacement of the occurrences of words in the rule R by letters of an alphabet of new letters.

Introduction

Splicing system was first introduced by Head in 1987 [Head, 1987]. These systems are used to model the recombinant action of restriction enzymes and a ligase on deoxyribonucleic acid (DNA) molecules. DNA is the genetic material in an organism which is made by joining nucleotides in a repetitive way into long, chain-like polymers. Nucleotides consist of three components, namely phosphate, sugar and a nitrogeneous base. The nucleotides differ from each other by their bases, namely adenine (A), cytosine (C), guanine (G) and thymine (T). Thermodynamically stable hydrogen bonding occurs between thymine and adenine, and between cytosine and guanine. There are two hydrogen bonds between adenine and thymine and three hydrogen bonds between cytosine and guanine. Two single-stranded DNAs (ssDNAs) can anneal together to form a double-stranded DNA (dsDNA) by hydrogen bonds between the bases. These bolds hold between the complimentary [Tamarin, 2002]. For example, a sequence of dsDNA

$$5' – \text{CCGGTACG} – 3'$$
$$3' – \text{GGCCATGC} – 5'$$

can be represented as string ccggtacg, or [C/G][C/G][G/C][G/C][T/A][A/T][C/G][G/C], where 5′ – CCGGTACG – 3′ and 3′ – GGCCATGC – 5′ represent two complimentary ssDNAs.

Enzymes are molecules that control chemical reaction in cells by acting as catalysts, that is, to speed up the rate of chemical reactions without being consumed themselves in the process. Two types of enzymes relevant for the modelling of splicing system are restriction enzymes and ligase. Restriction enzymes are endodeoxyribonucleases that recognize specific nucleotide sequences in double stranded DNA and cleave both strands of the double helix [Kim, 1997]. The place where restriction enzyme can cut a molecule is called a cutting site. After a DNA molecule is cut by a restriction enzyme, DNA fragments with sticky ends or blunt ends are produced. For DNA fragments with sticky ends, if these DNA fragments are mixed together, the fragments could join with each other subjected to the base pairing between their sticky ends. The union can be made permanent by another ligase enzyme, which forms covalent bonds along the backbone of each strand. This process, which is called the ligating operation, can results in a molecule of recombinant DNA.

Researches in splicing system established the relationship between formal language theory and the study of informational macromolecules. A model can be expressed by or identified with a language. Specific tasks of modelling have given rise to specific types of languages [Rozenberg and Salomaa, 1997]. The set of dsDNAs that may arise from an initial set of DNA molecules in the presence of specified enzyme activities is represented as a language over the four-symbol alphabet of deoxyribonucleotide pairs.

This paper shows the reduction of S_k splicing systems (with solid rules) into simple splicing systems. Several definitions used in this research are listed below.

Preliminaries

Let A be a fixed finite set used as an alphabet for the free monoid A^* that consists of all strings of symbols in A, including the null string [Head, 1998]. The formal definitions of a splicing system and a splicing language are stated in the following.

Definition 1. [Head, 1987] (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 $L = L(S)$.

The definition of a null context splicing system, which is used in the definition of a S_k splicing system, is given in the following

Definition 2. [Head, 1987] (Null Context Splicing System)
A *null context splicing system* is a splicing system $S = (A, I, B, C)$ for which each cleavage pattern in B and each in C has the form $(1, x, 1)$.

A particular type of splicing system and splicing language is the S_k splicing system and S_k splicing language, which are defined in the following.

Definition 3. [Head, 1998] (S_k Splicing System, S_k Splicing Language)
Let k be an integer ≥ -1. An *S_k splicing system* (S_kH system) is an null context splicing system $G = (A, I, R)$ for which, for each string r in R, length $r \leq k$. A language L is called an *S_k splicing language* (S_kH language), if there exists an S_kH system $G = (A, I, R)$ for which $L = L(G)$. The family of S_k splicing languages is denoted by S_kH.

An S_kH system may be reduced to a simple splicing system. The definitions of a simple splicing system and a simple splicing language are stated in the following.

Definition 4. [Laun, 1999] (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.

We introduce the concepts of solid and non-solid codes in the next section. These codes can be used to reduce S_k splicing systems into simple splicing systems (also called SH systems)

Solid and Non-Solid Codes

We first define some terms that will be useful in the later session.

Definition 3. (Solid Code, Solid Relative to a Language L)
A set S of words in A^* is a *solid code* if
(1) $w = xyz$ can hold with both w and y in S only when x and z are null.
(2) xy in S and yz in S can hold only if y is null.

In other words, we can say that a set S of words in A^* is a *solid code* if
(1) no word in S is a subword of any other word in S;
(2) no two distinct words in S overlap non-trivially.

A subset S in A^* is *solid relative to* L if u, v in S, then there is no factor w of L for which:
(i) $w = u = svt$ unless s and t are null;
(ii) $w = v = sut$ unless s and t are null;
(iii) $w = uv''$ where $u = u'v'$, $v = v'v''$, unless v' is null; and
(iv) $w = u'v$ where $v = u''v''$, $u = u'u''$, unless u'' is null.

We illustrate the definitions of solid and non-solid codes through three examples as given in the following.

Examples of Solid and Non-Solid Codes

Example 1.
Let $A = \{a, c, g, t\}$ be a set of alphabets. Examples of solid codes are $S_1 = \{a, c, g, t\}$, $S_2 = \{acg, accg, acccg, accccg, acccccg\}$ and $S_3 = ac^*g$, which is an infinite solid code. Examples of non-solid codes are:
(i) $T_1 = \{gac, ag\}$, since gac and ag are overlapping non-trivially,
(ii) $T_2 = \{ac, ag, acg, tttt\}$, since ac is a subword of acg,
(iii) $T_3 = \{acg, gca\}$, since acg and gca are overlapping non-trivially,
(iv) $T_4 = \{acggt, cg\}$, since cg is a subword of $acggt$.

In the next two examples, splicing systems with the rules not being solid are presented.

Example 2.
Suppose $A = \{a, c, g, t\}$, $I = \{tagcca, cagccagt\}$ and $R = \{ag, agc\}$. R is not solid since ag is a subword of agc.

Example 3.
Suppose $A = \{a, c, g, t\}$, $I = \{acgtttagca, ccgtaatagcc\}$ and $R = \{cg, tagc\}$. R is not solid since cg and $tagc$ overlap non-trivially.

When R is a solid code, the S_kH system (A, I, R) where k is the length of the longest word in R, may be identified with an appropriate $SH = S_1H$ system.

In the next section, reduction of S_k splicing systems to simple splicing systems using the concept of solid codes will be shown.

Reduction of Splicing Systems

In this section, two examples are presented to illustrate the reduction of S_k splicing systems having rule R as a solid code. Example 4 shows how an S_3H system can be reduced to an SH system using the concept of solid codes.

Example 4.
Suppose $S = (A, I, R)$ is a splicing system with $A = \{a, c, g, t\}$, $I = \{ccatgttagcgactgt, tacgattgttcgac\}$ and $R = \{gtt, cga\}$ a solid code. The set I can be partitioned into $I = \{ccat/gtt/ag/cga/ctgt, ta/cga/tt/gtt/cga/c\}$, since R is solid. Therefore words come apart naturally and uniquely into code words and `words-in-between'. Since the maximum length of a word in R is three, we have specified an S_3H system. However, due to the nice clean parsing into words in R and words-between, such an S_3H is really pretty much the same as an easily specified SH system over a different alphabet.

Suppose that we choose a new letter not in A for each word in R, say p and q. Then SH system is formed by using the parsed version of I given above: $A' = \{a, c, g, t\}$, $I' = \{ccatpagqctgt, taqttpqc\}$, $R' = \{p, q\}$. This is an SH system and obviously, it has 'maximal firm words': $ccat$, ag, $ctgt$, ta, tt, c. The language $L(A, I, R)$ is the image of the language $L(A', I', R')$ under the homomorphism of $h : (A') \rightarrow A^*$ generated by the one-to-one function $h: A' \rightarrow \{a, c, g, t, gtt, cga\}$ defined by $h(a) = a$, $h(c) = c$, $h(g) = g$, $h(t) = t$, $h(p) = gtt$, and $h(q) = cga$.

The following example shows how an S_4H system having R as a solid code can be reduced to an SH system.

Example 5.
Suppose $S = (A, I, R)$ is a splicing system with $A = \{a, c, g, t\}$, $I = \{tacggagtcacgagtc, agagtgacc\}$ and $R = \{ac, gagt\}$ a solid code. The set I can be partitioned into $I = \{t/ac/g/gagt/c/ac/gagt/c, a/gagt/g/ac/c\}$ since R is solid. Therefore, words come apart naturally and uniquely into code words and 'words-in-between'. Since the maximum length of a word in R is 4, we have specified an S_4H system. However, due to the nice clean parsing into words in R and the words-between, such an S_4H is really pretty much the same as an easily specified SH system over a different alphabet.

Suppose new letters not in A for each word in R, say p and q, are chosen.. Then, an SH system is formed by using the parsed version of I given above: $A' = \{a, c, g, t, p, q\}$, $I' = \{tpgqcpqc, aqgpc\}$, $R' = \{p, q\}$. This is an SH system and it has 'maximal firm words': t, g, c, and a. The language $L(A, I, R)$ is the image of the language $L(A', I', R')$ under the homomorphism of $h : (A') \rightarrow A^*$ generated by the one-to-one function $h: A' \rightarrow \{a, c, g, t, ac, gagt\}$ defined by $h(a) = a$, $h(c) = c$, $h(g) = g$, $h(t) = t$, $h(p) = ac$, and $h(q) = gagt$.

Example 4 and Example 5 have shown how splicing systems $L(A, I, R)$ having solid rules R can be reduced to SH systems by replacement of the occurrences of words in R by letters of an alphabet of new letters.

Conclusion

This study discusses the reduction of splicing systems to simple splicing systems using the new concept of solid codes. Some examples of solid and non-solid codes are given, including two examples that illustrate the reduction of S_kH systems to simple splicing systems.

Acknowledgements

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.

References

Head, T. (1987). Formal Language Theory and DNA: An Analysis of the Generative Capacity of Specific Recombinant Behaviors, *Bulletin of Mathematical Biology*, 49: 737-759.

Head, T. (1998). Splicing Representations of Strictly Locally Testable Languages, *Discrete Applied Mathematics*, 87: 139-147.

Kim, S. M. (1997). Computational Modeling for Genetic Splicing Systems, *SIAM J. Comput,* 26(5): 1284-1309.

Laun, E. G. (1999). Constants and Splicing Systems, Ph.D. Thesis, State University of New York at Binghamton.

Rozenberg, G. and Salomaa, A. (1997). Handbook of Formal Languages: Vol.1. Word, language, grammar, Springer-Verlag Berlin Heidelberg, New York.

Tamarin, R. H. (2002). Principles of Genetics, 7th. Ed, McGraw-Hill, New York.