On Representing Natural Languages and Bio-molecular Structures using Matrix Insertion-deletion Systems and its Computational Completeness*

Lakshmanan Kuppusamy[1], Anand Mahendran[1] and Shankara Narayanan Krishna[2]

[1]School of Computing Science and Engineering, VIT University, Vellore 632 014, India

[2]Department of Computer Science and Engineering, IIT Bombay, Powai - 400 076, India

Abstract. Insertion and deletion are considered to be the basic operations in Biology, more specifically in DNA processing and RNA editing. Based on these evolutionary transformations, a computing model *insertion-deletion* system has been proposed in formal language theory. Recently, in [14], a new computing model named *Matrix insertion-deletion system* has been introduced to model various bio-molecular structures. In this paper, we represent some natural language constraints such as *triple agreement*, *crossed dependency*, *copy language* using Matrix insertion-deletion systems and discuss how these constraints resemble some bio-molecular structures. Next, we analyze the computational completeness result for Matrix insertion-deletion system where the importance is given for not using any contexts when deletion rule takes place. We see that when the insertion-deletion system is combined with matrix grammar, the universality result is obtained with weight just 3 $(1, 1; 1, 0)$ whereas for insertion-deletion systems, the universality result is available with weight 4 $(1, 1; 1, 1)$.

1 Introduction

Linguistics agreed in late 1960's that many natural languages including English, are not context-free [4]. This initiated the idea of thinking grammars beyond the scope of context-free. As membership problem is tough to handle for context-sensitive grammars, they are not a good model for describing natural languages. To overcome these difficulties and to give a syntactical representation for natural languages, a notion called *mildly context-sensitive* (MCS) *grammar formalisms* has been defined by (majority of) computational linguistics people and later many attempts have been made to find MCS grammar formalisms [3], [5], [15]. A grammar formalism is said to be MCS formalism if it satisfies the following properties.

1. The class of its languages contain all context-free languages
2. The class of its languages contain the following three basic non-context-free languages:
   - *triple agreements:* $L_{ta} = \{a^n b^n c^n | n \geq 1\}$,

* This work was partially supported by the project SR/S3/EECE/054/2010, Department of Science and Technology (DST), New Delhi, India.

Kuppusamy L., Mahendran A. and Krishna Narayanan S..
On Representing Natural Languages and Bio-molecular Structures using Matrix Insertion-deletion Systems and its Computational Completeness.
DOI: 10.5220/0003308900470056
In *Proceedings of the 1st International Workshop on AI Methods for Interdisciplinary Research in Language and Biology* (BILC-2011), pages 47-56
ISBN: 978-989-8425-42-3

- *crossed dependencies:* $L_{cd} = \{a^n b^m c^n d^m | n, m \geq 1\}$, and
- *copy language* $L_{cp} = \{ww | w \in \{a, b\}^*\}$.

3. All languages in the class are *parsable* in *polynomial time*.
4. All languages in the class are *semilinear* or at least satisfy the *bounded growth property*.

In the last three decades, biology played a great role in the field of formal languages by being the root for the development of various biologically inspired computing models such as *sticker systems, splicing systems, Watson-Crick automata, insertion-deletion systems, p systems* [6], [11], [12]. Since, most of the language generating devices are based on the operation of rewriting systems, the insertion-deletion systems opened a particular attention in the field of formal languages. Informally, the insertion and deletion operations of an insertion-deletion systems are defined as follows: If a string α is inserted between two parts w_1 and w_2 of a string $w_1 w_2$ to get $w_1 \alpha w_2$, we call the operation as insertion, whereas if a substring β is deleted from a string $w_1 \beta w_2$ to get $w_1 w_2$, we call the operation as deletion.

DNA molecules may be considered as strings over alphabet consisting of four symbols namely a, t, g and c (denoted as Σ_{DNA}). Similarly, RNA molecules may be considered as strings over alphabet consisting of four symbols namely a, u, g and c (denoted as Σ_{RNA}). We discuss below in brief some of the important structures seen in bio-molecules such as protein, DNA and RNA. Fig.1. shows the structures (a) *stem*, (b) *cloverleaf* and (c) *dumbbell*. Since the bio-molecular structures can be defined in terms of sequence of symbols (i.e., strings) there exists a correlation between formal grammars and bio-molecular structures. The following example witnesses this correlation. Consider a context-free language $\{ww^R \mid w \in \{a, b\}^*\}$ (where w^R is the reverse of w) and a gene sequence $ctatcgcgatag$. As the complements are $\bar{a} = t$, $\bar{t} = a$, $\bar{g} = c$ and $\bar{c} = g$, the above gene sequence resembles the palindrome language $\{w\bar{w}^R \mid w \in \Sigma^*_{DNA}\}$ where $w = ctatcg$ and $\bar{w}^R = cgatag$. Like this, the structures mentioned in Fig.1. can be represented by context-free languages.

However, there are some more structures that are predominantly available in bio-molecules which cannot be modelled by context-free grammars. Fig.2. represents such structures (a) *pseudoknot* and (b) *attenuator*. A closer look at these bio-molecular structures shows a resemblance with well known natural language constructs, such as *crossed dependencies:* $\{a^n b^m c^n d^m \mid n, m \geq 1\}$ and *copy language:* $\{ww \mid w \in \{a, b\}^*\}$, see [8, 9]. Therefore, if a formal grammar is capable of generating the context-free and non-context-free languages, then that grammar is also suitable to represent bio-molecular structures and the vice versa is also true. Next, we discuss some attempts made in formal grammars to represent the above mentioned bio-molecular structures.

In the last two decades or so, many attempts have been made to establish the linguistic behaviour of biological sequences by defining new grammar formalisms like *cut grammars* [7], *crossed-interaction grammar* [10], *simple linear tree adjoining grammars* and *extended simple linear tree adjoining grammars* [20] which are capable of generating some of the biological structures mentioned above. However, there was no unique grammar system that encapsulate all essential and important bio-molecular structures. For example double copy language cannot be modelled by a simple linear tree adjoining grammar [20]. Very recently, a new biologically inspired computing model namely *Matrix insertion-deletion* system has been introduced in [14] by com-

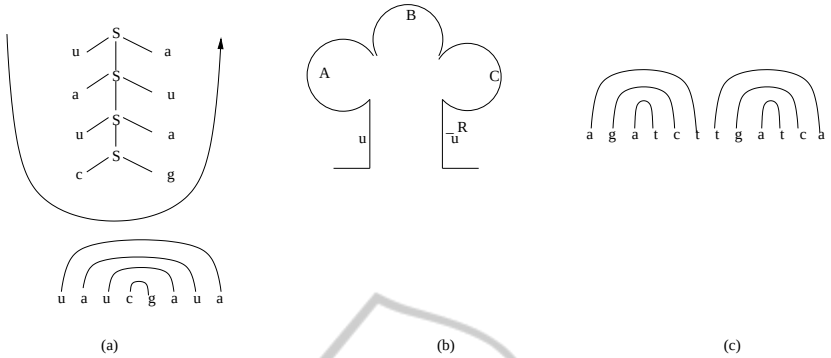

Fig. 1. Bio-molecular structures: (a) stem (b) cloverleaf (c) dumbbell.

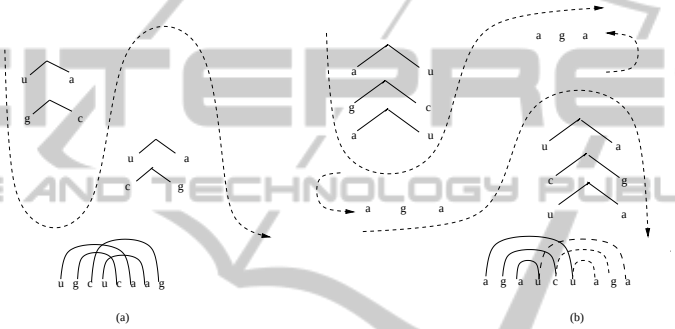

Fig. 2. Bio-molecular structures: (a) pseudoknot structure (b) attenuator.

bining insertion-deletion system and matrix grammars. This system represents all the above discussed bio-molecular structures and also the other structures such as *hairpin*, *non-ideal attenuator*, *ideal strings*, *orthodox string*. In this paper, we first briefly recall this system and using this system we generate a few natural language constraints and we discuss their coherences in bio-molecular structures. Thus we aim to show that Matrix insertion-deletion grammar system might be a unique model that is suitable for both natural language and bio-molecular representations.

Given an insertion-deletion system, the weight of the system is based on the maximal length of insertion, maximal length of the context used for insertion, maximal length of deletion, maximal length of the context used for deletion and they are (respectively) denoted as $(n, m; p, q)$. The total weight is defined as the sum of n, m, p, q. There have been many attempts to characterize the recursively enumerable languages (i.e., computational completeness) using insertion-deletion systems with less weights. In [16] the universality results were obtained with weight 5 (of the combinations $(1, 2; 1, 1)$, $(2, 1; 2, 0)$, $(1, 2; 2, 0)$). In [1] and [11], this result was improved with weight 4 (of the combinations $(1, 1; 1, 1)$ and $(1, 1; 2, 0)$ respectively). In [17], a variant of insertion-deletion systems called context-free insertion-deletion systems were introduced by considering no context in insertion and deletion rules. The universality of context-free insertion-deletion systems were proved initially with weight $(*, 0; *, 0)$ and reduced to weight 6 $(3, 0; 3, 0)$. Further, in the same paper, the result was improved to weight 5

(of the combinations $(3,0;2,0)$ and $(2,0;3,0)$). As Matrix insertion-deletion system is a variation of insertion-deletion systems and is more powerful than insertion-deletion systems, the universality result of this system must be obtained with less weight than 4. In this paper, we prove the universality result of Matrix insertion-deletion systems with just weight 3 $(1,1;1,0)$ where no contexts is considered for deletion thus the deletion operation works in a context-free manner.

2 Preliminaries

We assume that the readers are familiar with the notions of formal language theory. However, we recall the basic notions which are used in the paper. A finite non-empty set V or Σ is called an alphabet. We denote by V^* or Σ^* , the free monoid generated by V or Σ, by λ its identity or the empty string, and by V^+ or Σ^+ the set $V^* - \{\lambda\}$ or $\Sigma^* - \{\lambda\}$. The elements of V^* or Σ^* are called *words* or *strings*. For any word $w \in V^*$ or Σ^*, we denote the length of w by $|w|$. For more details on formal language theory, we refer to [13].

Next, we look into the basic definitions of insertion-deletion systems. Given an insertion-deletion system $\gamma = (V, T, A, R)$, where V is an alphabet, $T \subseteq V$, A is a finite language over V, R is a finite set triples of the form $(u, \beta/\alpha, v)$, where $(u, v) \in V^* \times V^*$, $(\alpha, \beta) \in (V^+ \times \{\lambda\}) \cup (\{\lambda\} \times V^+)$. The pair (u, v) is called as contexts. Insertion rule will be of the form $(u, \lambda/\alpha, v)$ which means that α is inserted between u and v. Deletion rule will be of the form $(u, \beta/\lambda, v)$, which means that β is deleted between u and v. In other words, $(u, \lambda/\alpha, v)$ corresponds to the rewriting rule $uv \rightarrow u\alpha v$, and $(u, \beta/\lambda, v)$ corresponds to the rewriting rule $u\beta v \rightarrow uv$.

Consequently, for $x, y \in V^*$ we can write $x \Longrightarrow^* y$, if y can be obtained from x by using either an insertion rule or a deletion rule which is given as follows: (the down arrow $\downarrow$ indicates the position where the string is inserted, the down arrow $\Downarrow$ indicates the position where the string is deleted and the underlined string indicates the string inserted/deleted)

1. $x = x_1 u^{\downarrow} v x_2,\ y = x_1 u \underline{\alpha} v x_2$, for some $x_1, x_2 \in V^*$ and $(u, \lambda/\alpha, v) \in R$.
2. $x = x_1 u \underline{\beta} v x_2,\ y = x_1 u^{\Downarrow} v x_2$, for some $x_1, x_2 \in V^*$ and $(u, \beta/\lambda, v) \in R$.

The language generated by γ is defined by

$$L(\gamma) = \{w \in T^* \mid x \Longrightarrow^* w, for\ some\ x \in A\}$$

where $\Longrightarrow^*$ is the reflexive and transitive closure of the relation $\Longrightarrow$.

Next, we discuss about the weight of the insertion-deletion system. An insertion-deletion system $\gamma = (V, T, A, R)$ is of $weight(n, m; p, q)$ if

$$\begin{aligned} n &= max\{|\alpha| \mid (u, \lambda/\alpha, v) \in R\} \\ m &= max\{|u| \mid (u, \lambda/\alpha, v) \in R\ or\ (v, \lambda/\alpha, u) \in R\} \\ p &= max\{|\beta| \mid (u, \beta/\lambda, v) \in R\} \\ q &= max\{|u| \mid (u, \beta/\lambda, v) \in R\ or\ (v, \beta/\lambda, u) \in R\} \end{aligned}$$

We denote by $INS_n^m DEL_p^q$ for $n, m, p, q \geq 0$, the family of languages $L(\gamma)$ generated by insertion-deletion systems of weight $(n', m'; p', q')$ such that $n' \leq n$, $m' \leq m$,

$p' \leq p$, $q' \leq q$. The total weight of γ is $n + m + p + q$.

Next, we will look into the definition of matrix grammars. A matrix grammar is an ordered quadruple $G = (N, T, S, M)$ where N is a set of non-terminals, T is a set of terminals, S is the start symbol and M is a finite set of nonempty sequences whose elements are ordered pairs (P, Q). The pairs are referred to as productions and written in the form $P \to Q$. The sequences are referred to as matrices and written $m = [P_1 \to Q_1, ..., P_r \to Q_r], r \geq 1$. The above matrix grammar is without appearance checking. The language generated by the matrix grammar is defined by $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$. A matrix grammar with appearance checking is defined as $G = (N, T, S, M, F)$ where F is a set of occurrences of rules in the matrices of M. While deriving, a rule may be exempted to apply if the rule is in F. The language generated by the matrix grammar with appearance checking is defined as $L_{ac}(G, F) = \{w \in T^* \mid S \Longrightarrow^* w\}$. The family of languages generated by matrix grammars without/with appearance checking is denoted by MAT^{λ}, MAT^{λ}_{ac} (the λ on the upper index indicates where the rule $P \to \lambda$ is allowed). For more details on matrix grammars, we refer to [2], [18].

3 Matrix Insertion-deletion Systems

In this section, we describe *Matrix insertion-deletion systems*. A Matrix insertion-deletion system is a construct $\Upsilon = (V, T, A, R)$ where V is an alphabet, $T \subseteq V$, A is a finite language over V, R is a finite set triples of the form in matrix format $[(u_1, \beta_1/\alpha_1, v_1), \ldots, (u_n, \beta_n/\alpha_n, v_n)]$, where $(u_k, v_k) \in V^* \times V^*$, and $(\alpha_k, \beta_k) \in (V^+ \times \{\lambda\}) \cup (\{\lambda\} \times V^+)$, with $(u_k, \beta_k/\alpha_k, v_k) \in R_{I_i} \cup R_{D_j} \cup R_{I_i/D_j}$, for $1 \leq k \leq n$. Here R_{I_i} denotes the matrix which consists of only insertion rules, R_{D_j} denotes the matrix which consists of only deletion rules and R_{I_i/D_j} denotes the matrix which consists of both insertion and deletion rules.

Consequently, for $x, y \in V^*$ we can write $x \Longrightarrow x' \Longrightarrow x'' \Longrightarrow \ldots \Longrightarrow y$, if y can be obtained from x by using a matrix consisting of insertion rules (R_{I_i}) or deletion rules (R_{D_j}) or insertion and deletion rules (R_{I_i/D_j}). In a derivation step the rules in a matrix are applied sequentially one after other in order and no rule is in appearance checking (note that the rules in a matrix are not applied in parallel). The language generated by Υ is defined by $L(\Upsilon) = \{w \in T^* \mid x \Longrightarrow^*_{R_\chi} w, \text{ for some } x \in A\}$, where $\chi \in \{I_i, D_j, I_i/D_j\}$ where $\Longrightarrow^*$ is the reflexive and transitive closure of the relation $\Longrightarrow$. Note that the string w is collected after applying all the rules in a matrix and also $w \in T^*$ only. The family of languages generated by Matrix insertion-deletion systems with weights $(n, m : p, q)$ is given as $MATINS^m_n DEL^q_p$.

Example 1. Consider the *mix* language $L_{ml} = \{n_a(w) = n_b(w) = n_c(w) \mid w \in \{a, b, c\}^*\}$. The language L_{ml} can be generated by $\Upsilon_{ml} = (\{a, b, c\}, \{a, b, c\}, \{\lambda\}, R)$ where R is given as $R_{I_1} = [(\lambda, \lambda/a, \lambda), (\lambda, \lambda/b, \lambda), (\lambda, \lambda/c, \lambda)]$. As no context is used in insertion rules a, b, c can be inserted anywhere in the derivation and the number of a, b, c are equal. It is easy to see that Υ_{ml} generates L_{ml}.
Remark. Note that L_{ml} is generated with weight just 2 $(1, 1; 0, 0)$.

3.1 Modelling Natural Language Constructs and Bio-molecular Structures

In this section, we represent some of the natural language constraints such as *crossed dependency*, *copy language*, *triple agreement*, *quadruple agreement*, *center-embedded structure* and discuss their coherences with bio-molecular structures.

In the next lemma, we represent crossed dependency structure in natural languages which models pseudoknot structure in bio-molecules.

Lemma 1. $L_{cd} = \{a^n b^m c^n d^m \mid m, n \geq 1\} \in \Upsilon_{cd}$.

Proof. The language L_{cd} can be generated by the Matrix insertion-deletion system $\Upsilon_{cd} = (\{a, b, c, d\}, \{a, b, c, d\}, \{abcd\}, \{R_{I_1} = [(a, \lambda/a, \lambda), (c, \lambda/c, \lambda)]$, $R_{I_2} = [(b, \lambda/b, \lambda), (d, \lambda/d, \lambda)]\ \})$. A sample derivation can be given as follows: $a^{\downarrow}bc^{\downarrow}d \Longrightarrow_{R_{I_1}} a^{\downarrow}\underline{a}bc^{\downarrow}\underline{c}d \Longrightarrow_{R_{I_1}} a\underline{a}ab^{\downarrow}c\underline{c}cd^{\downarrow} \Longrightarrow_{R_{I_2}} aaab\underline{b}cccd\underline{d} \Longrightarrow^{*}_{R_{I_1/I_2}} a^n b^m c^n d^m$.

The language L_{cd} resembles the pseudoknot structure language $L_{ps} = \{uv\bar{u}^R\bar{v}^R \mid u, v \in \Sigma^*_{DNA}\}$ in gene sequences as shown in Fig. 2(a). The language L_{ps} can be generated by the Matrix insertion-deletion system $\Upsilon_{ps} = (\{b, \bar{b}, \dagger_1, \dagger_2, \dagger_3, \dagger_4\}, \{b, \bar{b}\}, \{\lambda, \dagger_1\dagger_2 \dagger_3\dagger_4\}, R)$, where $b \in \{a, t, g, c\}$, $\bar{b}$ is complement of b and R is given as: $R_{I_1} = [(\lambda, \lambda/b, \dagger_1), (\lambda, \lambda/\bar{b}, \dagger_3)]$, $R_{I_2} = [(\lambda, \lambda/b, \dagger_2), (\lambda, \lambda/\bar{b}, \dagger_4)]$, $R_{D_1} = [(\lambda, \dagger_1/\lambda, \lambda), (\lambda, \dagger_3/\lambda, \lambda)]$, $R_{D_2} = [(\lambda, \dagger_2/\lambda, \lambda), (\lambda, \dagger_4/\lambda, \lambda)]$
A sample derivation is given as follows:

$^{\downarrow}\dagger_1\dagger_2^{\downarrow}\dagger_3\dagger_4 \Longrightarrow_{R_{I_1}} \underline{a}\dagger_1^{\downarrow}\dagger_2\underline{t}\dagger_3^{\downarrow}\dagger_4 \Longrightarrow_{R_{I_2}} a\dagger_1\underline{g}^{\downarrow}\dagger_2 t\dagger_3\underline{c}^{\downarrow}\dagger_4 \Longrightarrow_{R_{I_2}} a\dagger_1 g\underline{a}\dagger_2 t\dagger_3 c\underline{t}\dagger_4 \Longrightarrow_{R_{D_1}} a^{\Downarrow}ga\dagger_2 t^{\Downarrow}ct\dagger_4 \Longrightarrow_{R_{D_2}} aga^{\Downarrow}tct^{\Downarrow}$ □

In the next lemma, we model copy language in natural languages which resembles attenuator structure in gene sequences.

Lemma 2. $L_{cp} = \{ww \mid w \in \{a, b\}^*\} \in \Upsilon_{cp}$.

Proof. The language L_{cp} can be generated by the Matrix insertion-deletion system $\Upsilon_{cp} = (\{a, b, \dagger_1, \dagger_2\}, \{a, b\}, \{\lambda, \dagger_1\dagger_2\}, \{R_{I_1} = [(\lambda, \lambda/a, \dagger_1), (\lambda, \lambda/a, \dagger_2)]$, $R_{I_2} = [(\lambda, \lambda/b, \dagger_1), (\lambda, \lambda/b, \dagger_2)], R_{D_1} = [(\lambda, \dagger_1/\lambda, \lambda), (\lambda, \dagger_2/\lambda, \lambda)]\ \})$. A sample derivation is given as follows: $^{\downarrow}\dagger_1^{\downarrow}\dagger_2 \Longrightarrow_{R_{I_1}} \underline{a}^{\downarrow}\dagger_1\underline{a}^{\downarrow}\dagger_2 \Longrightarrow_{R_{I_1}} a\underline{a}^{\downarrow}\dagger_1 a\underline{a}^{\downarrow}\dagger_2 \Longrightarrow_{R_{I_2}} aa\underline{b}\dagger_1 aa\underline{b}\dagger_2 \Longrightarrow_{R_{D_1}} aab^{\Downarrow}aab^{\Downarrow}$.

The language L_{cp} resembles the attenuator language $L_{an} = \{u\bar{u}^R u\bar{u}^R \mid u \in \Sigma^*_{DNA}\}$ in gene sequences as shown in Fig.2(b). The language L_{an} can be generated by the Matrix insertion-deletion system $\Upsilon_{an} = (\{a, t, g, c, \dagger_1, \dagger_2\}, \{a, t, g, c\}, \{\lambda, \dagger_1\dagger_2\}, R)$, where R is given as: $R_{I_1} = [(\lambda, \lambda/a, \dagger_1), (\dagger_1, \lambda/t, \lambda), (\lambda, \lambda/a, \dagger_2), (\dagger_2, \lambda/t, \lambda)]$, $R_{I_2} = [(\lambda, \lambda/t, \dagger_1), (\dagger_1, \lambda/a, \lambda), (\lambda, \lambda/t, \dagger_2), (\dagger_2, \lambda/a, \lambda)]$, $R_{I_3} = [(\lambda, \lambda/c, \dagger_1), (\dagger_1, \lambda/g, \lambda), (\lambda, \lambda/c, \dagger_2), (\dagger_2, \lambda/g, \lambda)]$, $R_{I_4} = [(\lambda, \lambda/g, \dagger_1), (\dagger_1, \lambda/c, \lambda), (\lambda, \lambda/g, \dagger_2), (\dagger_2, \lambda/c, \lambda)]$, $R_{D_1} = [(\lambda, \dagger_1/\lambda, \lambda), (\lambda, \dagger_2/\lambda, \lambda)]$
A sample derivation is given as follows:

$^{\downarrow}\dagger_1^{\downarrow}\,^{\downarrow}\dagger_2^{\downarrow} \Longrightarrow_{R_{I_1}} \underline{a}^{\downarrow}\dagger_1^{\downarrow}\underline{t}\,\underline{a}^{\downarrow}\dagger_2^{\downarrow}\underline{t} \Longrightarrow_{R_{I_2}} at\underline{t}^{\downarrow}\dagger_1^{\downarrow}\underline{a}tat\underline{t}^{\downarrow}\dagger_2^{\downarrow}\underline{a}t \Longrightarrow_{R_{I_3}} atc\underline{c}^{\downarrow}\dagger_1^{\downarrow}\underline{g}atatc\underline{c}^{\downarrow}\dagger_2^{\downarrow}$
$\underline{g}at \Longrightarrow_{R_{I_4}} atcg\dagger_1\underline{c}gatatcg\dagger_2\underline{c}gat \Longrightarrow_{R_{D_1}} atcg^{\Downarrow}cgatatcg^{\Downarrow}cgat$ □

In the next lemma, we show the relevance between triple, quadruple agreement language to their corresponding structure in bio-molecules.

Lemma 3. $L_{ta} = \{a^n b^n c^n \mid n \geq 1\} \in \Upsilon_{ta}$.

Proof. The language L_{ta} can be generated by the Matrix insertion-deletion system $\Upsilon_{ta} = (\{a, b, c\}, \{a, b, c\}, \{abc\}, \{R_{I_1} = [(a, \lambda/ab, b), (b, \lambda/c, c)]\})$. A sample derivation can be given as follows: $a^{\downarrow}b^{\downarrow}c \Longrightarrow_{R_{I_1}} a\underline{a^{\downarrow}}\underline{b}b^{\downarrow}\underline{c}c \Longrightarrow_{R_{I_1}} aa\underline{ab}bb\underline{c}cc \Longrightarrow^*_{R_{I_1}} a^n b^n c^n$. The triple agreement language has relevances to triple-stranded DNA (triple helix structure) [9]. If we include d in V, T, replace the axiom as $abcd$ and R_{I_1} as $[(a, \lambda/ab, b), (c, \lambda/cd, d)]$ in Υ_{ta} we can see that Υ_{ta} generates quadruple agreement language $\{a^n b^n c^n d^n \mid n \geq 1\}$. It is mentioned in [9] that the quadruple agreement language can be regarded as a quadruple-stranded DNA (quadruple helix structure). □

In the next lemma, we show that center-embedded structure found in natural languages [19] can be viewed as a RNA stem in bio-molecules.

Lemma 4. *The center-embedded language* $L_{ce} = \{a^n b^m b^m a^n \mid n, m \geq 0\}$ *can be represented by Matrix insertion-deletion system.*

Proof. The language L_{ce} can be generated by the Matrix insertion-deletion system $\Upsilon_{ce} = (\{a, b, \dagger_1, \dagger_2, \dagger_3, \dagger_4\}, \{a, b\}, \{\lambda, \dagger_1\, \dagger_2\, \dagger_3 \dagger_4\}, R)$, where R is given as: $R_{I_1} = [(\dagger_1, \lambda/a, \lambda), (\dagger_4, \lambda/a, \lambda)]$ $R_{I_2} = [(\dagger_2, \lambda/b, \lambda), (\dagger_3, \lambda/b, \lambda)]$, $R_{D_1} = [(\lambda, \dagger_1/\lambda, \lambda), (\lambda, \dagger_4/\lambda, \lambda)]$ $R_{D_2} = [(\lambda, \dagger_2/\lambda, \lambda), (\lambda, \dagger_3/\lambda, \lambda)]$. It is easy to see that Υ_{ce} generates L_{ce}. The center-embedded language resembles the stem construct $\{u_1 u_2 u_3 \bar{u_3}^R \bar{u_2}^R \bar{u_1}^R \mid u_1, u_2, u_3 \in \Sigma^*_{DNA}\}$ in gene sequences as shown in Fig.1(a). □

Note. The dumbbell language $L_{db} = \{u\bar{u}^R v\bar{v}^R \mid u, v \in \Sigma^*_{DNA}\}$ (refer to Fig.1(c).) is found relevance with the natural language $L = \{a^n b^n c^m d^m \mid n, m \geq 0\}$. It is easy to generate this context-free language using Matrix insertion-deletion system.

4 Computational Completeness

In this section, we prove that the family of recursively enumerable languages can be characterized by Matrix insertion-deletion grammars with weight 3 $(1, 1; 1, 0)$. This universality result is achieved by simulating matrix grammars with appearance checking in strong binary normal form (SBNF).

Theorem 1. $MATINS^1_1 DEL^0_1 = RE$.

Proof. The part $MATINS^1_1 DEL^0_1 \subseteq RE$ is obvious. We now prove the other part. For each language $L \in RE$ there is a matrix grammar G with appearance checking in SBNF, such that $L(G) = L$. A matrix grammar G is given as (N, T, S, M, F) where N is a set of non-terminals, T is set of terminals, S is the start symbol, M is a finite set of matrices and F is a set of rules (used for appearance checking). A matrix grammar G is said to be in SBNF if $N = N_1 \cup N_2 \cup \{\$, \#\}$ where these three sets are mutually disjoint and the matrices in M are in one of the following forms:

1. $[S \rightarrow XA]$ $X \in N_1, A \in N_2$
2. $[X \rightarrow Y, A \rightarrow x]$ $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*, |x| \leq 2; x = x_1 x_2$
3. $[X \rightarrow Y, A \rightarrow \#]$ $X, Y \in N_1, A \in N_2$

4. $[X \to \lambda, A \to x]\ X \in N_1, A \in N_2, x \in T^*, |x| \leq 2$

There is only one matrix of type 1 and F consists exactly of all rules $A \to \#$ appearing in matrices of type 3, # is a trap symbol once used in the derivation it cannot be removed and the rules of matrix 4 are applied only in the last step of the derivation. Observe that there can be at most only one non-terminal symbol from N_1 at any point of the derivation.

Next, we construct a Matrix insertion-deletion system Υ which will simulate the matrix grammars in strong binary normal form with appearance checking. Let $\Upsilon = (V, T, A, R)$, where $V = N_1 \cup N_2 \cup \{S,]_{XA},]_Y, [_{x_1x_2},]_{x_1x_2}, [_\#,]_\#, \# \mid X, Y \in N_1, A \in N_2, x_1, x_2 \in (N_2 \cup T)\}$, $T = T$ (terminal set in matrix grammars), $A = S$, and R is given in simulation process.

The Matrix of type 1 can be simulated by the following Matrix insertion-deletion system rules $R_{I_1/D_1} = [(S, \lambda/]_{XA}, \lambda), (\lambda, S/\lambda, \lambda), (]_{XA}, \lambda/A, \lambda), (]_{XA}, \lambda/$
$X, A), (\lambda,]_{XA}/\lambda, \lambda) \mid [S \to XA],\ X \in N_1, A \in N_2]$. Here after, we omit the corresponding condition matrix rule (here $[S \to XA]$) as we are giving the simulation for each type of the matrix at the appropriate place. The simulation of type 1 is given as follows:

$$S^{\downarrow} \Longrightarrow S]_{\underline{XA}} \Longrightarrow^{\Downarrow}]_{XA}{}^{\downarrow} \Longrightarrow]_{XA}{}^{\downarrow}\underline{A} \Longrightarrow]_{XA}\underline{X}A \Longrightarrow XA$$

The Matrix of type 2 can be simulated by the following Matrix insertion-deletion system rules $R_{I_2/D_2} = [(X, \lambda/]_Y, \lambda), (\lambda, X/\lambda, \lambda), (]_Y, \lambda/Y, \lambda), (\lambda,]_Y/\lambda, \lambda),$
$(\lambda, \lambda/[_{x_1x_2}, A), (A, \lambda/]_{x_1x_2}, \lambda), (\lambda, A/\lambda, \lambda), ([_{x_1x_2}, \lambda/x_2,]_{x_1x_2}), ([_{x_1x_2}, \lambda/x_1, x_2),$
$(\lambda, [_{x_1x_2}/\lambda, \lambda), (\lambda,]_{x_1x_2}/\lambda, \lambda)]$
The simulation is given as follows:

$$X^{\downarrow} \Longrightarrow X]_{\underline{Y}} \Longrightarrow^{\Downarrow}]_Y{}^{\downarrow} \Longrightarrow]_Y\underline{Y} \Longrightarrow^{\Downarrow} Y$$

$${}^{\downarrow}A \Longrightarrow \underline{[_{x_1x_2}}A^{\downarrow} \Longrightarrow [_{x_1x_2}A]_{\underline{x_1x_2}} \Longrightarrow [_{x_1x_2}{}^{\Downarrow\ \downarrow}]_{x_1x_2} \Longrightarrow [_{x_1x_2}\ {}^{\downarrow}\underline{x_2}]_{x_1x_2} \Longrightarrow$$
$$[_{x_1x_2}\underline{x_1}x_2]_{x_1x_2} \Longrightarrow^{\Downarrow} x_1x_2]_{x_1x_2} \Longrightarrow x_1x_2{}^{\Downarrow}$$

Since there is only one symbol from N_1 at any derivation, the rule $(\lambda, X/\lambda, \lambda)$ will delete the used $X \in N_1$ correctly. Note that we introduce another $Y \in N_1$ only after deleting X. Similarly, the rule $(\lambda, A/\lambda, \lambda)$ can delete any $A \in N_2$, the (next) rule $([_{x_1x_2}, \lambda/x_2,]_{x_1x_2})$ makes sure that the used $A \in N_2$ is deleted. If any other A is deleted, the rule $([_{x_1x_2}, \lambda/x_2,]_{x_1x_2})$ cannot be applied as the used A will be in the middle of $[_{x_1x_2}$ and $]_{x_1x_2}$ and the derivation stops.

The Matrix of type 3 can be simulated by the following Matrix insertion-deletion system rules $R_{I_3/D_3} = [(X, \lambda/]_Y, \lambda), (\lambda, X/\lambda, \lambda), (]_Y, \lambda/Y, \lambda), (\lambda,]_Y/\lambda, \lambda),$
$(\lambda, \lambda/[_\#, A), (A, \lambda/]_\#, \lambda), (\lambda, A/\lambda, \lambda), ([_\#, \lambda/\#,]_\#), (\lambda, [_\#/\lambda, \lambda), (\lambda,]_\#/\lambda, \lambda)]$
The simulation is given as follows:

$$X^{\downarrow} \Longrightarrow X]_{\underline{Y}} \Longrightarrow^{\Downarrow}]_Y{}^{\downarrow} \Longrightarrow]_Y\underline{Y} \Longrightarrow^{\Downarrow} Y$$

$${}^{\downarrow}A \Longrightarrow \underline{[_\#}A^{\downarrow} \Longrightarrow [_\#A]_{\underline{\#}} \Longrightarrow [_\#{}^{\Downarrow\ \downarrow}]_\# \Longrightarrow [_\#\underline{\#}]_\# \Longrightarrow^{\Downarrow} \#]_\# \Longrightarrow \#^{\Downarrow}$$

Note that the appearance checking symbol # is not deleted in the derivation.

The Matrix of type 4 can be simulated by the following Matrix insertion-deletion system rules R_{I_4/D_4} $[(\lambda, X/\lambda, \lambda), (\lambda, \lambda/[_{x_1x_2}, A), (A, \lambda/]_{x_1x_2}, \lambda)$, $(\lambda, A/\lambda, \lambda), ([_{x_1x_2}, \lambda/x_2,]_{x_1x_2}), ([_{x_1x_2}, \lambda/x_1, x_2), (\lambda, [_{x_1x_2}/\lambda, \lambda), (\lambda,]_{x_1x_2}/\lambda, \lambda)]$
The simulation is given as follows:

$$X \Longrightarrow \lambda$$

$$^{\downarrow}A \Longrightarrow \underline{[_{x_1x_2}}A^{\downarrow} \Longrightarrow [_{x_1x_2}A]_{x_1x_2} \Longrightarrow [_{x_1x_2}\ ^{\Downarrow}\ ^{\downarrow}]_{x_1x_2} \Longrightarrow [_{x_1x_2}\ ^{\downarrow}\underline{x_2}]_{x_1x_2} \Longrightarrow [_{x_1x_2}\underline{x_1}x_2]_{x_1x_2} \Longrightarrow^{\Downarrow} x_1x_2]_{x_1x_2} \Longrightarrow x_1x_2\ ^{\Downarrow}$$

Note that $[_{x_1x_2},]_{x_1x_2},]_Y, [_\#,]_\#$ are considered to be single non-terminals. As the maximal length of inserted string is 1 (i.e., $n = 1$), the maximal length of the context used in insertion rules is 1 (i.e., $m = 1$), the maximum length of deleted string is 1 (i.e., $p = 1$) and no context is used in deletion rules (i.e., $q = 0$), the matrix grammars in SBNF can be simulated by Matrix insertion-deletion systems with a total weight of 3. This ends the proof. □

5 Conclusions

In this paper, we discussed the Matrix insertion-deletion grammar systems and using the system we have generated some context-free and non-context-free languages which are having some structural relevances with bio-molecules. Thus we identified a promising grammar system which can suit for both natural language representation and bio-molecular structural modelling. We have proved that Matrix insertion-deletion systems with weight 3 $(1, 1; 1, 0)$ can characterize all recursively enumerable languages and the system uses no contexts for deletion operation. As the insertion rules are context dependent (as we use context for insertion), they are more like context-sensitive and since deletions are done without looking any context, they are more like context-free. Therefore, this system uses the nature of both context-sensitiveness and context-freeness, it seems to be a promising model for representing natural languages. Thus analyzing this model more towards the properties of MCS formalisms is necessary and is left as a future work.

As the family of recursively enumerable languages is recognized with a total weight of 3, the context-free languages should be characterized with weight less than 3 in matrix insertion-deletion systems. As non-terminals are used in the context-free grammars, the simulated matrix insertion-deletion system must use deletion rules to delete the introduced non-terminals, it looks not possible to characterize context-free languages by Matrix insertion-deletion systems with weight less than 3. The same holds true for even regular languages, thus we reached to an hierarchical collapse. To avoid this hierarchical collapse, we can introduce two more new weights in Matrix insertion-deletion systems, namely s, t such that s denotes the total number of matrices and t denotes the maximum number of rules among all matrices. With the new weights the Matrix insertion-deletion systems can be represented as $MAT_s^t INS_n^m DEL_p^q$ and we believe that regular and context-free languages can be characterized with less weights counting the above said matrix measures. This leads to an interesting future work from generative power perspective.

Acknowledgements

The authors would like to thank Prof. Kamala Krithivasan for motivating us to work in insertion-deletion systems.

References

1. Akihiro Takaharai., Takashi Yokomori.: On the computational power of insertion-deletion systems, Natural Computing 2, pp. 321–36: Kluwer Academic Publishers, (2003)
2. Arto Salomaa.: Formal languages, New York and London: Academic Press, (1973)
3. Aravind K. Joshi.: An introduction to tree adjoining grammars. In Manaster-Ramer (ed.), Mathematics of Languages, John Benjamins, Amsterdam, Philadelphia, pp. 87–114, (1988)
4. Bar-Hillel., Shamir E.: Finite state languages: formal representations and adequacy problems. In: Bar Hillel (ed.), Lang. and Infn.: Addison-Wesley, Reading, MA, pp. 87–98, (1964)
5. Boullier P.: Range concatenation grammars. Proceedings of Sixth International Workshop on Parsing Technologies (IWPT), pp. 53–64 (2000)
6. Cristian S. Calude., Gheorghe Paŭn.: Computing with cells and atoms, An introduction to Quantum, DNA and Membrane Computing, London: Taylor and Francis, (2001)
7. David B. Searls.: Representing genetic information with formal grammars. In: Proceedings of the National Conference on Artificial Intelligence. pp. 386–391, (1988)
8. David B. Searls.: The linguistics of DNA. American Scientist. pp. 579–591, (1992)
9. David B. Searls.: The computational linguistics of biological sequences. In: Hunter, L.(ed.) Artificial Intelligence and Molecular Biology, AAAI Press, pp. 47–120, (1993)
10. Elena Rivas., Sean R. Reddy.: The language of RNA: A formal grammar that includes pseudoknots, Bioinformatics, vol 16. pp. 334–340, (2000)
11. Gheorghe Păun., Grzegorz Rozenberg., Arto Salomaa.: DNA Computing, New Computing Paradigms. Springer, (1998)
12. Gheorghe Păun.: Membrane Computing-An introduction. Springer, (2002)
13. John E. Hopcroft., Rajeev Motwani., Jeffrey D. Ullman.: Introduction to Automata Theory, Languages and Computation: Addison-Wesley, (2006)
14. Lakshmanan Kuppusamy., Anand Mahendran., Krishna S.: Matrix Insertion-Deletion Systems for Bio-molecular Structures, Submitted (to ICDCIT-2011), Aug (2010)
15. Lakshmanan Kuppusamy., Krishna S.N., Rama R.: Internal contextual grammars for mildly context sensitive languages, Research on Language and Computation, 5, pp. 181-197, (2007)
16. Lila Kari., Gheorghe Păun., G. Thierrin., S. Yu.: At the crossroads of DNA computing and formal languages: Characterizing RE using insertion-deletion systems. Proc. of 3rd DIMACS Workshop on DNA Based Computing, Philadelphia, pp. 318333, (1997)
17. Maurice Margenstern., Gheorghe Păun., Yurii Rogozhin., Sergey Verlan.: Context-free insertiondeletion systems, Theoretical Computer Science, 330, pp.339-348,(2005)
18. Rozenberg., Arto Salomaa.: Handbook of formal languages, Vol 1,2,3, Springer, (1997)
19. Solomon Marcus., Carlos Martin-Vide., Gheorghe Păun.: Contextual grammars as generative models of natural languages, Volume 24 , Issue 2, pp. 245–274, (1998)
20. Yasuo Uemura., Aki Hasegawa., Satoshi Kobayashi., Takashi Yokomori.: Tree adjoining grammars for RNA structure prediction. Theoretical Computer Science, 210: pp. 277–303, (1999)