On the Ambiguity and Complexity Measures of Insertion-Deletion Systems

Kamala Krithivasan[1], Lakshmanan Kuppusamy[2], Anand Mahendran[2], and Khalid M.[2]

[1] Department of Computer Science and Engineering, IIT Madras, Chennai-600 036, India
kamala@iitm.ac.in

[2] School of Computing Science and Engineering, VIT University, Vellore-632 014, India
{klakshma,manand,mkhalid}@vit.ac.in

Abstract. In DNA processing and RNA editing, gene insertion and deletion are considered as the basic operations. Based on the above evolutionary transformations, a computing model has been formulated in formal language theory known as *insertion-deletion* systems. In this paper we study about ambiguity and complexity measures of these systems. First, we define the various levels of ambiguity ($i = 0, 1, 2, 3, 4, 5$) for insertion-deletion systems. Next, we show that there are inherently i-ambiguous insertion-deletion languages which are j-unambiguous for the combinations $(i, j) \in \{(5, 4), (4, 2), (3, 1), (3, 2), (2, 1), (0, 1)\}$. Further, We prove an important result that the ambiguity problem of insertion-deletion system is undecidable. Finally, we define three new complexity measures $TLength-Con$, $TLength-Ins$, $TLength-Del$ for insertion-deletion systems and analyze the trade-off between the newly defined ambiguity levels and complexity measures.

Keywords: DNA processing, insertion-deletion systems, inherently ambiguous languages, unambiguous grammar, complexity measures.

1 Introduction

In the last few years, Natural Computing which includes biologically inspired computing is an area which is pursued with interest. It includes DNA computing, membrane computing and evolutionary computing among other topics. The developments which have taken place in DNA computing have inspired the definition and study of new theoretical models in formal language theory, *sticker systems, splicing systems, Watson-Crick automata, insertion-deletion systems* [2], [6] are some of the theoretical models inspired by the behaviour of DNA strands in biology. In [4] insertion operation was only considered and in [14] insertion-deletion systems were introduced. They have opened a new avenue in formal language theory as a new model for generating languages. Informally, the insertion and deletion operations of an insertion-deletion system is defined as follows: If a string α is inserted between two parts w_1 and w_2 of a string w_1w_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_1w_2, we call the operation as deletion.

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 [13] 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 [6], this result was improved with weight 4 (of the combinations $(1, 1; 1, 1)$ and $(1, 1; 2, 0)$ respectively).

Insertion-deletion operations have some relevances to some phenomena in human genetics. In the following Fig.1. we try to show how the insertion-deletion systems are applied in the field of genetics. Consider a single strand DNA sequence $S_1 = xuvyz$, where x, u, v, y, z are all strings. Add a single stranded DNA sequence $u'w'v'$ to the sequence $xuvyz$, where $u'v'$ are the Watson-Crick complements of the strings u, v and w' is the complement of some string w, see Fig.1(a). First, annealing will take place such that u' will stick to u and v' to v, thus we obtain the scenario as in Fig.1(b). Then a cut by a restriction enzyme to the double stranded DNA sequence uv in Fig.1(c) and by adding a primer z', we obtain a double stranded sequence as in Fig.1(d). Finally, by melting the double stranded sequence the two strands will be separated, hence we obtain two strings. One string will be of the form $S_2 = xuwvyz$ (as in Fig.1(e)). The string S_2 implies that the string w is inserted between u and v. Thus, the string S_2 obtained from S_1 shows the use of insertion operation in DNA sequences. A similar annealing can be theoretically performed for deletion operation.

Ambiguity is considered as one of the fundamental problems in formal language theory. A grammar is said to be ambiguous, if there exists more than one distinct derivation of the words in the generated language. As we have seen above that the insertion-deletion system can be applied theoretically in DNA processing, the ambiguity in DNA processing (which uses the insertion-deletion system) can happen in the following manner. Let W_1W_2 be a DNA strand and suppose we want to insert $W_3W_4W_5$ between W_1 and W_2 to obtain another DNA strand $W_1W_3W_4W_5W_2$. This can be done first by inserting W_3 between W_1 and W_2, followed by inserting W_4 between W_3 and W_2, followed by inserting W_5 between W_4 and W_2. The other sequence would be first by inserting W_5 between W_1 and W_2, followed by inserting W_4 between W_1 and W_5, followed by inserting W_3 between W_1 and W_4. This shows that ambiguity in gene sequences is also possible (i.e., starting from one sequence we are able to get another sequence in more than one way such that the intermediate sequences are different). This motivates us to define formally the ambiguity for insertion-deletion systems. Based on the components used for insertion-deletion system, different levels of ambiguity are

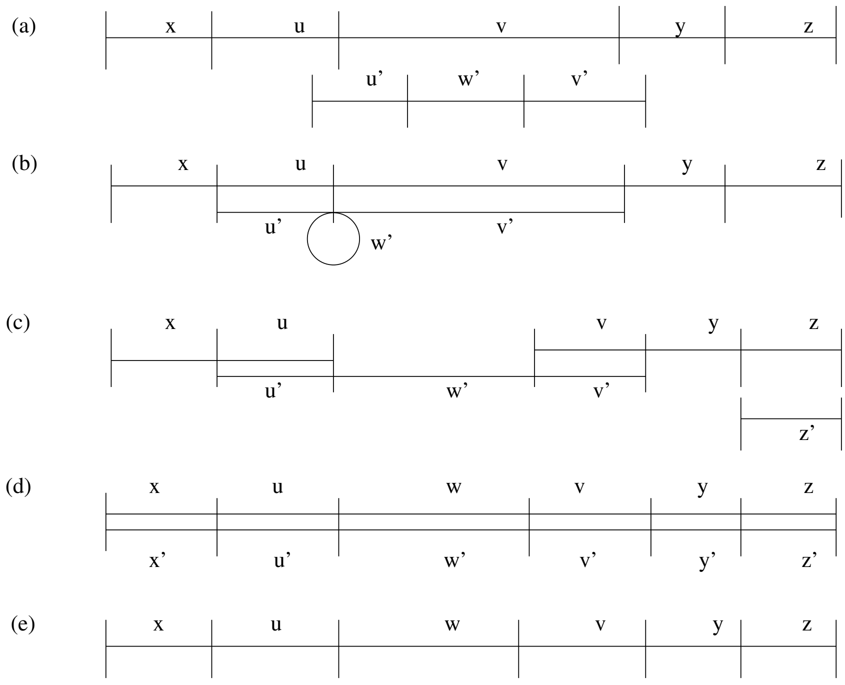

Fig. 1. Insertion by annealing

defined. Study of this concept of ambiguity may be useful in considering inheritance properties and phylogenetic trees [17]. More specifically, when these intermediate sequences are represented as phylogenetic trees, we can see that the trees are different and thus it might help us to identify the inheritance properties.

As insertion-deletion system is a counterpart for *contextual grammars* defined by S. Marcus [15] in 1969, here we briefly recall about the work done on ambiguity in contextual grammars. Unlike context-free grammars, defining ambiguity for contextual grammars is not so obvious since the derivation of contextual grammars consists of many components such as *axioms, contexts* and *selectors.* In [7], the notion of ambiguity was considered for the first time in this field, for external contextual grammars. Then, in [10,16], several levels of ambiguity were defined for internal contextual grammars by considering the components used in the derivation. There were many open problems formulated in [10,16] on different aspects of ambiguity and some of them have been solved in [11]. For more details on contextual grammars, we refer to [8].

As Fig.1. shows the applicability of insertion-deletion systems in gene sequences, storing of such sequences would be economical if the corresponding insertion-deletion system is economical. Such an economical system can be identified by means of descriptional complexity measures. The basic measures for insertion systems are defined in [8]: $Ax, MAx, TAx, Prod, Symbol.$ As the insertion-deletion systems is the slight modification of insertion systems, the measures $Ax, MAx, TAx, Prod$ are even applicable to insertion-deletion systems. Here economical system means the corresponding system should be minimal with respect to some measure. Once the ambiguity and complexity measures are defined the trade-off between them can be analyzed by identifying languages whose corresponding grammars are ambiguous and minimal in measure M, but unambiguous and not minimal in measure M.

Since insertion-deletion systems is an intermediate model between contextual grammars and context sensitive grammars, analyzing the ambiguity and developing new complexity measures of such systems would be more interesting. In Section 3, we define the various ambiguity levels ($i = 0, 1, 2, 3, 4, 5$) for insertion-deletion systems. Next, we show that there are inherently i-ambiguous insertion-deletion languages which are j-unambiguous for the combinations $(i, j) \in \{(5, 4), (4, 2), (3, 1), (3, 2), (2, 1), (0, 1)\}$. Next, we discuss an example that shows how the ambiguity level defined for insertion-deletion systems can be interpreted in gene sequences. Further, we also prove that there is no solution (i.e., no algorithmic procedure) to decide whether a given arbitrary insertion-deletion system is ambiguous or not. In other words, the ambiguity problem for an insertion-deletion system is undecidable. Finally, in Section 4, we introduce three new complexity measures $TLength - Con$ (total length of contexts used in insertion/deletion rules), $TLength - Ins$ (total length of the contexts used in insertion rules plus the length of the strings to be inserted), $TLength - Del$ (total length of the contexts used in deletion rules plus the length of the strings to be deleted) and we analyze the trade-off between the newly defined ambiguity levels and the above complexity measures of insertion-deletion systems.

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 is called an alphabet. We denote by V^*, the free monoid generated by V, by λ it identity or the empty string, and by V^+ the set $V^* - \{\lambda\}$. The elements of V^* are called *words* or *strings*. For any word $w \in V^*$, we denote the length of w by $|w|$.

Next, we will look into the basic definitions of insertion-deletion systems. Given an insertion-deletion (in short ins-del) 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 triples of the form $(u, \alpha/\beta, v)$, where $(u, v) \in V^*$, $(\alpha, \beta) \in (V^+ \times \{\lambda\}) \cup (\{\lambda\} \times V^+)$. 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 $uv \to u\alpha v$, and $(u, \beta/\lambda, v)$ corresponds to the rewriting rule $u\beta v \to uv$.

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/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

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

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

Next, we will introduce the various descriptional complexity measures of ins-del systems. Given a ins-del system $\gamma = (V, T, A, R)$, the basic measures of ins-del systems are defined as follows:

$$Ax(\gamma) = \text{card}(A), MAx(\gamma) = \max_{w \in A} |w|, TAx(\gamma) = \sum_{w \in A} |w|,$$
$$Prod(\gamma) = \text{card}(R),$$

where Ax denotes the number of axioms, MAx denotes the maximum length of an axiom, TAx denotes total length of all axioms, $Prod$ denotes the number of insertion-deletion rules. For $M \in \{Ax, MAx, TAx, Prod\}$ and for a language L, we define $M(L) = \min\{M(\gamma) \mid L = L(\gamma)\}$. We call γ as a minimal system for L with respect to the measure M or we simply say γ is minimal in M for L. For a measure M and a language L, we define $M^{-1}(L) = \{\gamma \mid L(\gamma) = L$ and $M(\gamma) = M(L)\}$. Here, $M^{-1}(L)$ denotes the set of all optimal systems for L with respect to the measure M. Two measures M_1, M_2 are said to be *incompatible* if there exists a language L such that $M_1^{-1}(L) \cap M_2^{-1}(L) = \emptyset$. Otherwise, M_1 and M_2 are said to be *compatible*. For more details on complexity measures, we refer to [8].

3 Various Ambiguity Levels

In this section, we define various ambiguity levels for ins-del system based on the components used in the derivation.

Consider the following derivation step in a ins-del system γ, $\delta : w_1 \Longrightarrow w_2 \Longrightarrow \ldots \Longrightarrow w_m, m \geq 1$, such that $w_1 \in A$ and the following scenarios can happen (1) $w_j = x_{1,j} u_j v_j x_{2,j}$ and $w_{j+1} = x_{1,j} u_j \alpha_j v_j x_{2,j}$, when an insertion rule $(u_j, \lambda/\alpha_j, v_j)$ is used, where $x_{1,j}, u_j, v_j, x_{2,j} \in V^*$. (2) $w_j = x_{1,j} u_j \beta_j v_j x_{2,j}$ and $w_{j+1} = x_{1,j} u_j v_j x_{2,j}$, when a deletion rule $(u_j, \beta_j/\lambda, v_j)$ is used, where $x_{1,j}, u_j, v_j, x_{2,j} \in V^*$. The sequence which consists of used axiom, strings to be inserted/deleted is called as *Control Sequence* which is given as follows: $w_1, \alpha_1/\beta_1,\ \alpha_2/\beta_2, \alpha_3/\beta_3, \ldots, \alpha_{m-1}/\ \beta_{m-1}$. The sequence which consists of used axiom, the string (α_j/β_j) to be inserted/deleted and the used contexts (u_j, v_j) is called *Complete Control Sequence* which is given as follows: $w_1, (u_1, \alpha_1/\beta_1, v_1)$, $(u_2, \alpha_2/\beta_2, v_2), (u_3, \alpha_3/\beta_3, v_3), \ldots, (u_{m-1}, \alpha_{m-1}/\beta_{m-1}, v_{m-1})$. The position where insertion (α) /deletion (β) takes place can be given by the *description* of δ, as follows: $w_1, x_{1,1} u_1 (\alpha_1\ /\beta_1) v_1\ x_{2,1}, x_{1,2} u_2 (\alpha_2/\beta_2) v_2 x_{2,2}, x_{1,3} u_3 (\alpha_3/\beta_3) v_3 x_{2,3}, \ldots,$ $x_{1,m-1} u_{m-1} (\alpha_{m-1}/\beta_{m-1}) v_{m-1} x_{2,m-1}$.

Definition 1. *1. A ins-del system γ, is said to be 0-ambiguous, if there exist at least two different axioms, $w_1, w_2 \in A$, $w_1 \neq w_2$, such that they both derive the same word z, i.e., $w_1 \Longrightarrow^+ z$, $w_2 \Longrightarrow^+ z$.*

2. A ins-del system γ, is said to be 1-ambiguous, if there are two different unordered control sequences which derives the same word.

3. A ins-del system γ, is said to be 2-ambiguous, if there are two different unordered complete control sequences which derives the same word.

4. A ins-del system γ, is said to be 3-ambiguous, if there are two different ordered control sequences which derives the same word.

5. A ins-del system γ, is said to be 4-ambiguous, if there are two different ordered complete control sequences which derives the same word.

6. A ins-del system γ, is said to be 5-ambiguous, if there are two different descriptions which derives the same word.

More precisely, an ins-del system γ to be 2 or 4 ambiguous, it should have at least two distinct contexts and to be 1 or 3 ambiguous it should have at least two distinct strings used for insertion/deletion. A system which is not i-ambiguous, for some $i = 0, 1, 2, 3, 4, 5$, is said to be *i-unambiguous.* A language L is *inherently i-ambiguous* if every system γ generating L is i-ambiguous. A language for which a i-unambiguous system exists is called *i-unambiguous.* From the above definitions, it is easy to see that each inherently i-ambiguous language is inherently j-ambiguous for $(i, j) \in \{(1, 2), (1, 3), (2, 4), (3, 4), (4, 5)\}$. However, the converse is not true. Therefore, whenever we show that a language is inherently j-ambiguous, we want to make sure that the result is not followed by the fact that the language is inherently i-ambiguous for the above i and j. In fact, in the following theorems, we do the same.

3.1 Inherently Ambiguous Ins-del Languages

In this section, we show that there exist inherently i-ambiguous ins-del languages for $i = 0, 2, 3, 4, 5$.

Theorem 1. *There are inherently 5-ambiguous ins-del languages which are 4-unambiguous.*

Proof. Consider the language $L_1 = \{a^n bba^m \mid n, m \geq 1\}$. The language L_1 can be generated by the following ins-del system γ_1.

$$\gamma_1 = (\{a, b\}, \{a, b\}, \{abba\}, \{(a, \lambda/a, \lambda)\}).$$

As, the system γ_1 uses only string a for both insertion/deletion and only one context (a, λ) obviously, the system γ_1 is 4-unambiguous. Hence, the language L_1 is 4-unambiguous.

To prove the language L_1 is inherently 5-ambiguous, consider an arbitrary system γ_1' which generates L_1. The axiom in the system γ_1' should be of the form $a^r bba^s, r, s \geq 1$. The contexts used in insertion/deletion rule could be in one of the forms (a^{r_1}, b), (a^{r_1}, bb), (a^{r_1}, a^{r_2}), (a^{r_1}, λ), (λ, a^{r_2}), (b, a^{r_2}), $(bb, a^{r_2}), r_1, r_2 \geq 1$. The string which will be inserted/deleted should be of the form $a^p, p \geq 1$. To prove the system γ_1' is ambiguous, consider a word $a^k bba^l \in L_1$. From this word, the word $a^{k'} bba^{l'}$ (for large values of k' and l') can be obtained by two different descriptions which differs by the position where the string a is inserted/deleted. First by getting $a^{k'}$ from a^k and then $a^{l'}$ from a^l or the other way round. Hence, the system γ_1' is ambiguous. Any ins-del system generating this language will have this property and hence it is ambiguous. Therefore, the language L_1 is inherently 5-ambiguous. □

Theorem 2. *There are inherently 4-ambiguous ins-del languages, which are 2-unambiguous.*

Proof. Consider the language $L_2 = \{a^n b^n c^m d^m \mid n, m \geq 1\}$. The language L_2 can be generated by the following ins-del system γ_2.

$$\gamma_2 = (\{a, b, c, d\}, \{a, b, c, d\}, \{abcd\}, \{(a, \lambda/ab, b), (c, \lambda/cd, d)\}).$$

From the system γ_2 it is easy to see that both the contexts (a, b) and (c, d) are required to the generate the language. In order to generate equal number of a's and b's in the language L_2 the system has to use the context (a, b). Similarly, to generate equal number of c's and d's in the language L_2 the system has to use the context (c, d). Since no other alternate contexts are available in the system γ_2 to generate equal number of a's and b's and equal number of c's and d's the system γ_2 is 2-unambiguous. Therefore, the language L_2 is 2-unambiguous.

To prove the language L_2 is inherently 4-ambiguous, consider an arbitrary system γ_2' which generates L_2. The system γ_2' should have the axiom of the form $a^i b^i c^j d^j, i, j \geq 1$. In order to generate the strings of the form $a^n b^n cd, n \geq 1$, the system should have the context of the form $(a^{r_1}, b^{r_2}), r_1, r_2 \geq 1$ and the string to be inserted/deleted should be of the form $a^{t_1} b^{t_1}$, $t_1 \geq 1$. Similarly, to generate the strings of the form $abc^m d^m, m \geq 1$, the system should have the context of the form $(c^{s_1}, d^{s_2}), s_1, s_2 \geq 1$ and the string to be inserted/deleted should be of the form $c^{u_1} d^{u_1}$, $u_1 \geq 1$. To prove the system γ_2' is ambiguous, consider a word $a^{r_1} b^{r_1} c^{s_1} d^{s_1}, r_1, s_1 \geq 1 \in L_2$. From this word, the word $a^p b^p c^q d^q$ can be obtained by two different ordered complete control sequences. In one sequence the required number of a's and b's can be inserted/deleted by using the context of the form (a^{r_1}, b^{r_2}), followed by the insertion/deletion of c's and d's by using the context of the form (c^{s_1}, d^{s_2}). In another sequence, first the required number of c's and d's can be inserted/deleted by using the context of the form (c^{s_1}, d^{s_2}), followed by the insertion/deletion of a's and b's by using the context of the form (a^{r_1}, b^{r_2}). Hence the system γ_2' is ambiguous. Any ins-del system generating this language can have at least two different complete control sequences. Therefore, the language L_2 is inherently 4-ambiguous. □

Theorem 3. *There are inherently 3-ambiguous ins-del languages which are 1 and 2 unambiguous.*

Proof. Consider a language $L_3 = \{ca^n b^m d \mid n, m \geq 1\}$. The language L_3 can be generated by the following ins-del system γ_3.

$$\gamma_3 = (\{a, b, c, d\}, \{a, b, c, d\}, \{cabd\}, \{(a, \lambda/a, \lambda), (b, \lambda/b, \lambda)\}).$$

From the system γ_3, it is easy to see that both the contexts and strings are required to generate the language. In order to generate the language L_3, the context (a, λ) and the string a has to be used $n - 1$ times and similarly the context (b, λ) and the string b has to be used $m - 1$ times. Therefore, the system γ_3 is 1 and 2-unambiguous. Hence, the language L_3 is 1 and 2-unambiguous.

To prove the language L_3 is inherently 3-ambiguous, consider an arbitrary system γ_3' which generates L_3. The system γ_3' should have an axiom of the form $ca^ib^jd, i, j \geq 1$. In order to generate L_3, insertion rules should be of the form, $(c^i, \lambda/a^j, \lambda)$, $(a^i, \lambda/a^j, \lambda)$, $(a^i, \lambda/a^j, a^k)$, $(c^i, \lambda/a^j, a^k)$, $(\lambda, \lambda/a^j, a^k)$, $(a^i, \lambda/a^j, b^k)$, $(b^i, \lambda/b^j, b^k)$, $(b^i, \lambda/b^j, \lambda)$, $(\lambda, \lambda/b^j, b^k)$, $(a^i, \lambda/b^j, b^k)$, $(\lambda, \lambda/b^j, d^k)$, $(b^i, \lambda/b^j, d^k)$, where $i, j, k \geq 1$. Similarly, the system should have deletion rules of the form, $(c^i, a^j/\lambda, \lambda)$, $(a^i, a^j/\lambda, \lambda)$, $(a^i, a^j/\lambda, a^k)$, $(c^i, a^j/\lambda, a^k)$, $(\lambda, a^j/\lambda, a^k)$, $(a^i, a^j/\lambda, b^k)$, $(b^i, b^j/\lambda, b^k)$, $(b^i, b^j/\lambda, \lambda)$, $(\lambda, b^j/\lambda, b^k)$, $(a^i, b^j/\lambda, b^k)$, $(\lambda, b^j/\lambda, d^k)$, $(b^i, b^j/\lambda, d^k)$, where $i, j, k \geq 1$. The string used for the insertion/deletion should be of the form $a^i, i \geq 1$, $b^j, j \geq 1$. From this, we can derive two different ordered control sequences which derives the same word. Consider an arbitrary word $ca^ib^jd \in L_3$. From this word, the word ca^tb^sd can be derived in two different ordered control sequences. In one sequence, the required number of a's can be inserted/deleted, followed by the insertion/deletion of required number of b's. In another sequence, first the required number of b's can be inserted/deleted, followed by the insertion/deletion of required number of a's. Therefore, the system γ_3' is ambiguous. Any ins-del system generating L_3 will have this property. Hence, the language L_3 is inherently 3-ambiguous. □

Theorem 4. *There are inherently 2-ambiguous ins-del languages which are 1-unambiguous.*

Proof. Consider the language $L_4 = \{ba^n \mid n \geq 0\} \cup \{a^nc \mid n \geq 0\} \cup \{ba^nc \mid n \geq 0\}$. The language L_4 can be generated by the following ins-del system γ_4.

$$\gamma_4 = (\{a, b, c\}, \{a, b, c\}, \{b, c, bc\}, \{(b, \lambda/a, \lambda), (\lambda, \lambda/a, c)\}).$$

As, the system γ_4 is having only one string which is used for insertion, the system γ_4 is 1-unambiguous. Therefore, the language L_4 is 1-unambiguous.

To prove the language L_4 is inherently 2-ambiguous, consider an arbitrary system γ_4' which generates L_4. The system γ_4' must have three axioms of the form ba^i, a^jc, ba^kc, $i, j, k \geq 0$. If the system γ_4' is having less than three axioms, it will generate strings not in the language. In order to generate the first part of the language the system must have an insertion rule of the form $(b, \lambda/a^i, \lambda), i \geq 1$. Similarly, the deletion rule must be of the form $(b, a^i/\lambda, \lambda), i \geq 1$. To generate the second part of the language the system must have an insertion rule of the form $(\lambda, \lambda/a^j, c), j \geq 1$. Similarly, the deletion rule must be of the form $(\lambda, a^j/\lambda, c), j \geq 1$. The string to be inserted/deleted should be of the form a^p, $p \geq 1$. To prove the system γ_4' is ambiguous, consider a word $ba^lc \in L_4$, where $l = k + ij$. This word can be derived from the axiom ba^kc by two different unordered complete control sequences. In one sequence the context (b, λ) can be used j times and in another sequence the context (λ, c) can be used i times. Thus obtaining two different unordered complete control sequences. Hence, the system γ_4' is 2-ambiguous. Any ins-del system generating the language L_4 will have this property. Therefore, the language L_4 is inherently 2-ambiguous. □

Theorem 5. *There are inherently 0-ambiguous ins-del languages without context, which are 1-unambiguous with finite context.*

Proof. Consider the language $L_5 = \{a, b\}^+$. The language L_5 can be generated by the following ins-del system γ_5.

$$\gamma_5 = (\{a, b\}, \{a, b\}, \{a, b\}, \{(a, \lambda/a, \lambda), (a, \lambda/b, \lambda), (b, \lambda/b, \lambda), (b, \lambda/a, \lambda)\}).$$

Since, the string a or b is inserted only to the right of the axiom by using the contexts (a, λ) or (b, λ), any word in the language can be generated in a unique manner. Hence, the system γ_5 is 1-unambiguous. Therefore, the language L_5 is 1-unambiguous. Moreover, any word in L_5 can be generated from only one axiom, therefore γ_5 is 0-unambiguous.

To prove the language L_5 is inherently 0-ambiguous if no contexts is used, consider an arbitrary system γ_5' which generates L_5. The system must have two axioms of the form $a^{r_1}, r_1 \geq 1$ and $b^{r_2}, r_2 \geq 1$. The string to be inserted/deleted should be of the form a^{p_1} and $b^{p_2}, p_1, p_2 \geq 1$. To prove γ_5' is ambiguous, consider a word $b^i a^j b^k \in L_5$ for large values of i, j, k. This word can be derived from two axioms a^{r_1} and b^{r_2} as follows:

$$a^{r_1} \Longrightarrow^*_{ins} b^i a^{r_1} \Longrightarrow^*_{del} b^i a^j \Longrightarrow^*_{ins} b^i a^j b^{s_1} \Longrightarrow^*_{del} b^i a^j b^k$$
$$b^{r_2} \Longrightarrow^*_{ins} b^{r_2} a^j \Longrightarrow^*_{del} b^{r_2} a^j b^{s_2} \Longrightarrow^*_{ins} b^{r_2} a^j b^k \Longrightarrow^*_{del} b^i a^j b^k$$

Since the system is without contexts, insertion/deletion can happen at any place. Any system without contexts which generates the language L_5 will have this property and hence it is ambiguous. Therefore, the language L_5 is inherently 0-ambiguous if no contexts is considered. □

3.2 Ambiguity Issues in Gene Sequences

In this subsection, we show an example for how the level 5-ambiguity discussed for ins-del systems can be interpreted in gene sequences. Consider an *orthodox string* available in gene sequences. A string w over a complementary alphabet Σ is called orthodox iff it is (i) the empty string ϵ, or (ii) the result of inserting two adjacent complementary element $b\bar{b}$, for some $b \in \Sigma$, anywhere in an orthodox string [3]. A language is orthodox iff it contains only orthodox strings. The orthodox language L_{od} can be generated by the ins-del system $\gamma_{od} = (\{b, \bar{b}\}, \{b, \bar{b}\}, \{\lambda\}, R)$, where $b \in \{a, t, g, c\}$, $\bar{b}$ is complement of b (i.e $\bar{a} = t,\ \bar{g} = c,\ \bar{t} = a,\ \bar{c} = g$) and R is given as $R = [(\lambda, \lambda/b\bar{b}, \lambda)]$. Consider the following string in orthodox language *gctagcat*. This string can be derived in two different descriptions by γ_{od} The two different descriptions are given as follows:

$$\textit{Description } 1 :^{\downarrow} ta \Longrightarrow \underline{gc}ta^{\downarrow} \Longrightarrow gcta\underline{gc}^{\downarrow} \Longrightarrow gctagc\underline{at}$$
$$\textit{Description } 2 : ta^{\downarrow} \Longrightarrow {}^{\downarrow}ta\underline{gc} \Longrightarrow \underline{gc}tagc^{\downarrow} \Longrightarrow gctagc\underline{at}$$

Note that the axiom, order of insertion of strings, order of contexts (here (λ, λ)) all are same in both derivations, but the position of insertion is different in each derivation. Therefore, the grammar γ_{od} is 5-ambiguous. Thus, starting from same (gene) sequence, we are able to get the same sequence, but the inter-mediate gene sequences are different. This suggests that there may be more than one way that a gene sequence can be processed.

3.3 Decidability of Ambiguity for Ins-del Systems

First, we will slightly brief about Post Correspondence Problem(PCP). Let Σ be an alphabet set containing at least two symbols. An instance of PCP has two ordered sets of strings $x = (x_1, ..., x_n)$ and $y = (y_1, ..., y_n)$ over Σ, we say that this instance of PCP has a solution if there is a sequence $i_1, ..., i_m$, $m \geq 1$, with $1 \leq i_j \leq n$ for each $1 \leq j \leq m$, such that $x_{i_1}...x_{i_m} = y_{i_1}...y_{i_m}$. It has been proved that the PCP problem is not decidable in the sense that there cannot exist an algorithm which will take an arbitrary instance of PCP as input and say whether this instance of PCP has a solution or not.

Decidability is one of the basic questions to be answered in formal language theory. *Emptiness*, *Finiteness* are some of the examples for decidability problems. For more details on decidability problems, we refer to [9]. Once the ambiguity is defined for the ins-del system, one question naturally arises on the ambiguity of ins-del systems: *Does there exist an algorithm to decide whether a given arbitrary ins-del system γ is ambiguous or not?.* In the next theorem, we prove that ambiguity problem of ins-del systems is undecidable by using the Post Correspondence Problem (this is called reducing PCP to the ambiguity problem).

Theorem 6. *The ambiguity* (*of any* i, $i = 0, 1, 2, 3, 4, 5$) *of ins-del systems is undecidable.*

Proof. Let $\Sigma = \{a', b'\}$. Consider two arbitrary words $(x_1, ..., x_n)$ and $(y_1, ..., y_n)$ over $\Sigma = \{a', b'\}$. Construct an ins-del system γ.

$$\gamma = (\{a, a', b, b', c, d, e\}, \{a, a', b, b', c, d, e\}, \{ccccecd, ccdcccd\}, I)$$

where the insertion rules (I) is given as follows:

$I_1 = (cc, \lambda/d, cce)$
$I_2 = (ccccec, \lambda/x_{i_1}a^{i_1}b, d)$
$I_3 = (x_{i_k}, \lambda/x_{i_{k+1}}a^{i_{k+1}}b, a^{i_k}b), \quad 1 \leq i_k \leq n$
$I_4 = (dcc, \lambda/e, c)$
$I_5 = (ccdccc, \lambda/y_{j_1}a^{j_1}b, d)$
$I_6 = (y_{j_l}, \lambda/y_{j_{l+1}}a^{j_{l+1}}b, a^{j_l}b), \quad 1 \leq j_l \leq n$

Let w_1 be the word derived from the axiom *ccccecd* (The $\downarrow$ denotes the position where the string is inserted in the next derivation step and the number at the suffix in each derivation symbol '$\Longrightarrow$' indicates which insertion rule is applied).

$$ccccec^{\downarrow}d \Longrightarrow_{I_2} ccccecx_{i_1}^{\downarrow}a^{i_1}bd \Longrightarrow_{I_3}^{*} cc^{\downarrow}ccecx_{i_1}...x_{i_k}a^{i_k}b...a^{i_1}bd$$
$$\Longrightarrow_{I_1} ccdccecx_{i_1}...x_{i_k}a^{i_k}b...a^{i_1}bd.$$

Let w_2 be the word derived from the axiom *ccdcccd*.

$$ccdccc^{\downarrow}d \Longrightarrow_{I_5} ccdcccy_{j_1}^{\downarrow}a^{j_1}bd \Longrightarrow_{I_6}^{*} ccdcc^{\downarrow}cy_{j_1}...y_{j_l}a^{j_l}b...a^{j_1}bd$$
$$\Longrightarrow_{I_1} ccdccecy_{j_1}...y_{j_l}a^{j_l}b...a^{j_1}bd.$$

By comparing w_1 and w_2, they are both are equal iff $k = l$ and $i_1 = j_1, ..., i_k = j_k$ and $x_{i_1}...x_{i_k} = y_{i_1}...y_{i_k}$. So, there exists a Post Correspondence Solution (PCP) for the instance $i_1, ..., i_k$ for the strings $(x_1, ..., x_n)$ and $(y_1, ..., y_n)$. We can see that, the system is ambiguous as it derives the same word from two different axioms.

Conversely, if PCP for $(x_1, ..., x_n)$ and $(y_1, ..., y_n)$ has a solution for the instance $i_1, ..., i_k$, such that $x_{i_1}...x_{i_k} = y_{i_1}...y_{i_k}$, then clearly the system γ is 0-ambiguous (i.e., from two different axioms, we are able to get two words w_1 and w_2, such that $w_1 = w_2$.).

Therefore, if the ambiguity problem of ins-del systems is decidable, then PCP is said to be decidable by the above reduction method which is not the case. Therefore, 0-ambiguity problem for ins-del system is not decidable. In a similar fashion, we see that i-ambiguity problem ($i = 1, 2, 3, 4, 5$) for ins-del system is undecidable. □

4 New Complexity Measures

In this section, we define some new complexity measures for ins-del systems. Given a ins-del system $\gamma = (V, T, A, R)$, the basic complexity measures have been discussed in the preliminary section. Let the system γ contains the insertion rules of the form $(u_1, \lambda/\alpha_1, v_1), ..., (u_m, \lambda/\alpha_m, v_m)$ and similarly the deletion rules of the form $(u_1, \beta_1/\lambda, v_1), ..., (u_m, \beta_m/\lambda, v_m)$. Based on the above rules, the new measures are defined as follows:

$$TLength - Con(\gamma) = \sum_{(u,v)\in R} |uv|,$$
$$TLength - Ins(\gamma) = \sum_{(u,\alpha,v)\in R} |u\alpha v|,$$
$$TLength - Del(\gamma) = \sum_{(u,\beta,v)\in R} |u\beta v|.$$

The measure $TLength - Con$ specifies the total length of the contexts used in insertion/deletion rules, $TLength - Ins$ specifies the total length of the contexts used in insertion rules plus the length of the strings (α) to be inserted and $TLength - Del$ specifies the total length of the contexts used in deletion rules plus the length of the strings (β) to be deleted.

4.1 Trade-off between Ambiguity and Measures

In this section, we analyze the trade-off between the newly defined ambiguity levels and various complexity measures. We show that when a minimal measure M is chosen for a language L, then all the corresponding systems which generates www$L$ are ambiguous. On the other hand, when an unambiguous system is chosen for L the system is not minimal in M. The corollary is the consequences of the theorem and by the fact that the measures Ax, MAx, TAx are pairwise compatible [5]. Before, we proceed to the results, let us adapt the notion of 'pseudo inherently ambiguous' introduced in [12]. A system γ is said to be restricted if γ satisfies some conditions imposed on it. Here, we impose the condition minimal measure (with respect to a measure M) to systems and we say that γ is restricted with respect to M.

Definition 2. *A language L is said to be pseudo inherently ambiguous (when considered minimal with respect to M) if every restricted system (with respect to M) generating L is ambiguous.*

In the next theorem, we show that if we want to store the language L_6 with the corresponding system which is minimal in terms of *Prod*, then any such system is (5-) ambiguous. On the other hand, if we want to store L_6 with the corresponding system which is unambiguous, then any such system is not minimal with respect to *Prod*. A similar result in discussed in Theorem 8. Thus, in the following section we make a trade-off between ambiguity and complexity measures in choosing a system for a language.

Theorem 7. *There are pseudo inherently 5-ambiguous ins-del languages when Prod is considered minimal, but there are 5-unambiguous systems which are not minimal in Prod.*

Proof. Consider the language L_6=$\{d(abb)^n \mid n \geq 0\} \cup \{(abb)^n c \mid n \geq 0\}$. The language L_6 can be generated by the following 5-ambiguous ins-del system γ_6.

$$\gamma_6 = (\{a, b, c, d\}, \{a, b, c, d\}, \{d, dabb, c, abbc\}, \{(abb, \lambda/abb, \lambda)\}).$$

The system γ_6 is minimal with respect to *Prod*. From γ_6, it is easy to see that $Prod(L_6) = 1$. Any system which generates L_6 must have at least one insertion/deletion rule. Hence, the language L_6 is minimal in *Prod*.

Consider any system γ_6' which generates L_6 for which $Prod(\gamma_6') = 1$. Since different *abb* is chosen for insertion/deletion in deriving the words $d(abb)^i$ and $(abb)^j c$ for large value of i and j, the position where the string *abb* is inserted/deleted differs, but derives the same word. Hence, the system γ_6' is ambiguous. Note that the system γ_6' is 4-unambiguous.

However, the language L_6 is 5-unambiguous as we can give a 5-unambiguous $\gamma_6'' = (\{a,b,c,d\}, \{a,b,c,d\}, \{d,c\}, \{(d,\lambda/abb,\lambda), (\lambda,\lambda/abb,c)\})$ which generates L_6. The first part of the language can be generated by inserting the string abb to the right of d. Similarly, the second part of the language can be generated by inserting the string abb to the left of c. As the position of inserting the string never changes in both parts of the language, the system γ_6'' is unambiguous. Note that, the system γ_6'' is not minimal in $Prod$, since $Prod(\gamma_6'') = 2$. □

Theorem 8. *There are pseudo inherently 4-ambiguous ins-del languages when Ax is considered minimal, but there are 4-unambiguous systems which are not minimal in Ax.*

Proof. Consider the language $L_7 = \{b(a)^{3n} \mid n \geq 0\} \cup \{c(a)^{3n} \mid n \geq 0\} \cup \{ba^{3n}ca^{3m} \mid n,m \geq 0\}$. The language L_7 can be generated by the following 4-ambiguous ins-del system γ_7.

$$\gamma_7 = (\{a,b,c\}, \{a,b,c\}, \{b,c,bc\}, \{(b,\lambda/aaa,\lambda), (c,\lambda/aaa,\lambda)\}).$$

The system γ_7 is minimal with respect to Ax. From γ_7, it is easy to see that $Ax(L_7) \leq 3$. Any system which generates L_7 must have at least three axioms corresponding to the three parts of the language. If the system γ_7 is having less than three axioms, it will generate strings not in the language. Hence, $Ax(L_7) \geq 3$, which implies $Ax(L_7) = 3$.

Consider any system γ_7' which generates L_7 for which $Ax(\gamma_7') = 3$. Since the string $ba^{3i}, i \geq 0 \in L_7$, the system should have insertion/deletion rule of the form $(b,\lambda/aaa,\lambda)$ $/(b,aaa/\lambda,\lambda)$. Similarly, since the string $ca^{3j}, j \geq 0 \in L_7$, the system should have insertion/deletion rule of the form $(c,\lambda/aaa,\lambda)$ $/(c,aaa/\lambda,\lambda)$. Consider a word $ba^{3r}ca^{3s}, r,s \geq 0$, this word can be derived from the axiom of the form $ba^{3p}ca^{3q},\ p,q \geq 0$ by two different ordered complete control sequences. In one sequence the insertion/deletion $(b,\lambda/aaa,\lambda)$ $/(b,aaa/\lambda,\lambda)$ rules can be used first, followed by the insertion/deletion rules $(c,\lambda/aaa,\lambda)$ $/(c,aaa/\lambda,\lambda)$. In another derivation, the insertion/deletion rules $(c,\lambda/aaa,\lambda)$ $/(c,aaa/\lambda,\lambda)$ can be used first, followed by the insertion/deletion rules $(b,\lambda/aaa,\lambda)$ $/(b,aaa/\lambda,\lambda)$. Note that since the string aaa to be inserted/deleted is same in both the derivations, the system is 1 and 3-unambiguous.

However, the language L_7 is 4-unambiguous as we can give a 4-unambiguous $\gamma_7'' = (\{a,b,c\}, \{a,b,c\}, \{b, baaa, c, caaa, bc, baaac, bcaaa, baaacaaa\}, \{(a,\lambda/aaa,\lambda)\})$ which generates L_7. Note that the system γ_7'' is not minimal in Ax. □

Corollary 1. *There are pseudo inherently 4-ambiguous ins-del languages when Ax, MAx, TAx are minimal, but there are 4-unambiguous systems which are not minimal with respect Ax, MAx, TAx.*

5 Conclusion

Insertion-deletion systems were defined and motivated by the way DNA strands are inserted and deleted. In this paper, we defined various ambiguity levels

($i = 0, 1, 2, 3, 4, 5$) for insertion-deletion systems. Next, we showed that there are inherently i-ambiguous insertion-deletion languages which are j-unambiguous for the following combinations $(i, j) \in \{(5, 4), (4, 2), (3, 1), (3, 2), (2, 1), (0, 1)\}$. We have not proved the result for the pair $(1, 0)$ and is left as open. We have also shown an example that how the ambiguity levels defined for ins-del systems can be interpreted in gene sequences. Then, we proved that the ambiguity problem for insertion-deletion systems is undecidable. Further, we defined three new complexity measures $TLength - Con$, $TLength - Ins$, $TLength - Del$. We discussed the trade-off between the newly defined ambiguity levels and complexity measures in insertion-deletion systems. We aimed to show that there are pseudo inherently i-ambiguous insertion-deletion languages which are i-unambiguous. A few more complexity measures like maximal length of an inserted and deleted strings can be defined and the trade-off between the ambiguity levels and new complexity measures can be analyzed as a future work.

Acknowledgment. The second author would like to acknowledge the project SR/S3/EECE/054/2010, Department of Science and Technology, New Delhi, India. The third author would like to acknowledge the scheme "Mentoring of Engineering Teachers by INAE Fellows" of Indian National Academy of Engineering, New Delhi, India and his work was partially carried out during the author's visit to Indian Institute of Technology Madras, Chennai, India.

References

1. Takaharai., A., Yokomori, T.: On the computational power of insertion-deletion systems. In: Natural Computing, vol. 2, pp. 321–336. Kluwer Academic Publishers (2003)
2. Calude, C.S., Paŭn, G.: Computing with cells and atoms. An introduction to Quantum, DNA and Membrane Computing. Taylor and Francis, London (2001)
3. Searls, D.B.: The computational linguistics of biological sequences. In: Hunter, L.(ed.) Artificial Intelligence and Molecular Biology, pp. 47–120. AAAI Press (1993)
4. Galiukschov, B.S.: Semicontextual grammars. Mat. Logical. Mat. Ling., Talinin University, 38–50 (1981) (in Russian)
5. Georgescu, G.: The Syntactic Complexity of Internal Contextual Grammars and Languages. In: Martin-Vide, C. (ed.) II Intern. Conf. Mathematical Linguistics, Amsterdam, pp. 17–28 (1997)
6. Păun, G., Rozenberg, G., Salomaa, A.: DNA Computing, New Computing Paradigms. Springer (1998)
7. Păun, G.: Contextual Grammars. The Publ. House of the Romanian Academy of Sciences, Bucuresti (1982)
8. Păun, G.: Marcus Contextual Grammars. Kluwer Academic Publishers (1997)
9. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (2006)
10. Ilie, L.: On Ambiguity in Internal Contextual Languages. In: Martin-Vide, C. (ed.) II Intern. Conf. Mathematical Linguistics, Tarragona, 1996, pp. 29–45. John Benjamins, Amsterdam (1997)

11. Lakshmanan, K.: A note on Ambiguity of Internal Contextual Grammars. Theoretical Computer Science, 436–441 (2006)
12. Lakshmanan, K., Anand, M., Krithivasan, K.: On the Trade-off Between Ambiguity and Measures in Internal Contextual Grammars. In: Cămpeanu, C., Pighizinni, G., (eds.) DCFS 2008, Charlottetown, Canada, pp. 216–223 (2008)
13. Kari, L., Păun, G., Thierrin, G., Yu, S.: At the crossroads of DNA computing and formal languages: Characterizing RE using insertion-deletion systems. In: Proc. of 3rd DIMACS Workshop on DNA Based Computing, Philadelphia, pp. 318–333 (1997)
14. Kari, L.: On insertion/deletion in formal languages. Ph.D Thesis, University of Turku (1991)
15. Marcus, S.: Contextual Grammars. Rev. Roum. Pures. Appl. (1969)
16. Martin-Vide, C., Miguel-Verges, J., Păun, G., Salomaa, A.: Attempting to Define the Ambiguity in Internal Contextual Languages. In: Martin-Vide, C. (ed.) II Intern. Conf. Mathematical Linguistics, Tarragona 1996, pp. 59–81. John Benjamins, Amsterdam (1997)
17. Setubal, Meidanis: Introduction to Computational Molecular Biology. PWS Publishing Company (1997)