A Note on Restricted Insertion-Deletion Systems

István Katsányi*

Abstract

In the past few years several papers showed, that various generative mechanisms in formal language theory that use insertion and deletion operations are capable of generating any recursively enumerable languages [1, 2, 3, 4, 5, 6, 7, 8, 9]. Since such systems are also models of molecular computing, for practical reasons it is important to examine these systems in a restricted case, in which the number of symbols in the model of the alphabet is limited. In [4] it is showed that we can define the generated language of an insertion-deletion system in such a way, that a two-letter alphabet is enough to generate any recursively enumerable language. In this note we complete this result by showing that the same generative capacity can be obtained even if we define the generated language the traditional way.

1 Introduction

The insertion grammars (or semi-contextual grammars) were introduced in [10] (see also [11]) as a model of the constructions of natural languages. It is an important model of formal languages of its own right, but it gained even more significance by the emerging of the field of DNA computing, since using a standard laboratory technique called *PCR site-specific oligonucleotide mutagenesis* insertions or deletions of nucleotide sequences into or from the strands of DNA molecules are possible. Hence by inspecting the practical applicability of the formal models we may gain functioning molecular computers. However, it is important to keep the constructions as simple as possible.

*Eötvös Loránd University, Department of Algorithms and Applications, 1117 Budapest, Pázmány Péter sétány 1/C, e-mail: kacsa@ludens.elte.hu

2 Preliminaries

An *insertion-deletion system* (or shortly an *insdel system*) is a construct $\gamma = (V, T, A, I, D)$, where V is a finite *alphabet*, $T \subseteq V$ is the *terminal alphabet*, $A \subseteq V^*$ is the finite set of *axioms* and $I, D \subseteq V^* \times V^* \times V^*$ are the finite sets of *insertion and deletion rules*, respectively. For two words $x, y \in V^*$ the relation $x \Longrightarrow_\gamma y$ holds when either $x = x_1uvx_2$, $y = x_1uzvx_2$, $x_1, x_2 \in V^*$ and $(u, z, v) \in I$, or $x = x_1uzvx_2$, $y = x_1uvx_2$, $x_1, x_2 \in V^*$ and $(u, z, v) \in D$. Let $\Longrightarrow_\gamma^*$ be the reflexive, transitive closure of $\Longrightarrow_\gamma$. The language generated by γ is $L(\gamma) = \{w \in T^* \mid x \Longrightarrow_\gamma^* w, x \in A\}$. In papers [1, 2, 4, 6, 7, 9] we can find different proofs for even stronger variants of the following theorem:

Theorem 1. *The family of languages generated by insertion-deletion systems equals to the family of recursively enumerable languages.*

In [4] a variant of the former construct is defined: $\gamma = (\{a, c\}, T, h, A, I, D)$, where a and c are two specified symbols, T is the (finite) *terminal alphabet*, $h : T^* \to V^*$ is a λ-free morphism (λ is the empty word), $A \subseteq \{a, c\}^*$ is the finite set of *axioms* and $I, D \subseteq \{a, c\}^* \times c^* \times \{a, c\}^*$ are the finite sets of *insertion and deletion rules*, respectively. The relation $\Longrightarrow_\gamma$ is defined the usual way. The language generated by γ is

$$L(\gamma) = h^{-1}(\{w \in \{a, c\}^* \mid z(aca)^n \Longrightarrow_\gamma^* (aca)^m w, \text{ for some } n, m \geq 0, z \in A\}).$$

This language definition may look a little bit strange, but it has some biological motivations. Large part of the human genome consists of short repeated sequences having no known function. A possible hypothesis can be that this junk DNA builds a *workspace* for computation. Contexts of $(aca)^*$ may refer to this workspace. In spite of the restrictions, these systems were shown in [4] to be as powerful as the regular insertion-deletion systems.

3 Restricted insertion-deletion systems

Here we define a new kind of insertion-deletion system which has additional constraints comparing to the regular model described earlier and show that it is capable of universal computation. A *restricted insertion-deletion system* is a construct $\gamma = (V, T, h, A, I, D)$, where V is an *alphabet* consisting of two letters, T is a finite alphabet called the *terminal alphabet*, $h : T^* \to V^*$ is a λ-free morphism, A is a finite subset of V^*, the set of *axioms*, I and D are finite subsets of $V^* \times V^* \times V^*$, the *insertion and deletion rules*, respectively. The role of V, A, I and D coincides with the regular model. The relation $\Longrightarrow_\gamma$ is also defined the usual way. The

morphism h is needed to define languages over an arbitrary finite alphabet. The language generated by γ is $L(\gamma) = h^{-1}(\{w \in V^* \mid z \Longrightarrow^* w, \text{ where } z \in A\})$.

It is easy to see, that for every (regular) insdel system $\gamma = (V, T, A, I, D)$ there exists a restricted insdel system γ' for which $L(\gamma) = L(\gamma')$. Hint: Suppose that $V = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, and define the morphism $g : V^* \to \{a, c\}^*, \quad g(\alpha_i) = ac^i a$ $(1 \leq i \leq n)$, where a and c are arbitrary symbols, $n \geq 1$. Let h be a mapping from T^* to $\{a, c\}^*$ such that for all $u \in T^*$ $h(u) = g(u)$. We may extend g the usual way to map *sets* of words, and even sets of *triplets* of words. The $\gamma' = (\{a, c\}, T, h, g(A), g(I), g(D))$ restricted insdel system is equivalent with γ, that is for each $u, v \in V^*$ the relation $u \Longrightarrow_\gamma v$ holds exactly when $g(u) \Longrightarrow_{\gamma'} g(v)$, hence $L(\gamma) = L(\gamma')$. Using this observation the proof of the next theorem is trivial. However, in the following proof we will show another approach and simulate a type-0 Chomsky grammar directly. The simulation works clearly differently than the above and may be of interest.

Theorem 2. *The family of languages generated by restricted insertion-deletion systems equals to the family of recursively enumerable languages.*

Proof. Let T be an arbitrary alphabet and let $L \subseteq T^*$ be an arbitrary recursively enumerable language. Let us suppose, that L is generated by the grammar $G = (N, T, S, P)$ in Penttonen normal form. Moreover, let us substitute all rules of the form $X \to A \in P$ $(A \in N \cup T)$ by rules $X \to AE$, $E \to \lambda$, where E is a new symbol. Hence the rules of P have one of the following forms: $X \to AB$, $X \to \lambda$, $XY \to XZ$, where $X, Y, Z \in N$ and $A, B \in N \cup T$.

Let us denote the elements of the set $N \cup T$ by the symbols $\alpha_1, \alpha_2, \ldots, \alpha_n$, where $S = \alpha_1$. Similarly, let us enumerate the rules of G: $P = \{r_1, r_2, \ldots, r_s\}$. The symbols of G will be coded by the morphism g:

$$g : (N \cup T)^* \to \{a, c\}^*, \quad g(\alpha_i) = ac^i a \quad (1 \leq i \leq n).$$

Let $h(\alpha_i) = g(\alpha_i) \quad (\alpha_i \in T)$. The following restricted insertion-deletion system will generate L: $\gamma = (\{a, c\}, T, h, \{aca\}, I, D)$, where the elements of I and D are listed below $(i, j, k \in [1, n], q \in [1, s])$:

1. For each rule $r_q : \alpha_i \to \alpha_j \alpha_k$ we have two rules: $(ac^i, c^{q+n-i} aac^k, a) \in I$ and $(ac^j, c^{q+n-j}, a) \in D$.
2. For each rule $r_q : \alpha_i \to \lambda$ we have $(\lambda, ac^i a, \lambda) \in D$.
3. For $r_q : \alpha_i \alpha_j \to \alpha_i \alpha_k$ we have $(ac^i aac^j, c^{q+n-j}, a) \in I$ and $(ac^k, c^{q+n-k}, a) \in D$.

The idea of the construction is that we may simulate every derivation step of G by either a single deletion rule or by an insertion rule followed by a deletion rule. After such an insertion we get the subword $ac^{n+q}a$, where $q \in [1, s]$ uniquely

determines the used rule. Since there is no word over T that h maps to such a word, this subword must be eliminated somehow in order to reach a word that has effect of the generated language. The insertion and deletion rules are constructed in a way that this subword can only be changed by the deletion rule that belongs to the qth rule. After the usage of this deletion rule the simulation of the derivation step of G is completed. It is possible that the deletion rule does not follow immediately its pair, but since no other rule has effect of the above-mentioned subword, each derivation in γ that ends in a word in the domain of h must use the rule on this subword, and the result of the derivation does not change if we perform the application of this rule earlier, immediately after the usage of its insertion pair. Due to the size restrictions of this note the details of the verification are left to the reader. □

4 Conclusions

In this note we showed that insertion-deletion systems remain powerful enough to generate any recursively enumerable languages even if we allow an alphabet of two letters only, provided that we use an inverse morphism to map the words of this alphabet to the alphabet of the given language. We do not use extra workspace, as an earlier approach did. Since it is assumed, that context-dependent insertions and deletions can be performed on DNA strands, we get a new proof that DNA computation is universal.

References

[1] Carlos Martin-Vide, Gheorghe Păun, and Arto Salomaa. Characterizations of recursively enumerable languages by means of insertion grammars. *Theoretical Computer Science*, 205:195–205, 1998.

[2] Lila Kari and Gabriel Thierrin. Contextual insertions/deletions and computability. *Information and Computation*, 131(1):47–61, November 25 1996.

[3] Mark Daley, Lila Kari, Greg Gloor, and Rani Siromoney. Circular contextual insertions/deletions with applications to biomolecular computation. In *Proceedings of SPIRE'99, the 6th Symposium on String Processing and Information Retrieval, Cancun, Mexico, 21–24 September, 1999*, pages 47–54.

[4] Lila Kari, Gheorghe Păun, Gabriel Thierrin, and Sheng Yu. At the crossroads of DNA computing and formal languages: Characterizing recursively enumerable languages using insertion-deletion systems. In *Proceedings of the 3rd DIMACS Workshop on DNA Based Computers, University of Pennsylvania, June 23–25, 1997*, pages 318–333.

[5] Lila Kari. *On Insertion and Deletion in Formal Languages*. PhD thesis, University of Turku, 1991.

[6] Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa. *DNA Computing. New Computing Paradigms*. Springer-Verlag, Berlin, 1998.

[7] A. Takahara and T.Yokomori. On the computational power of insertion-deletion systems. In *Proceedings of DNA'8, the 8th International Meeting on DNA-based Computers, Sapporo, Japan June 10–13, 2002, also in LNCS vol. 2568, 2003, pages 269–280*.

[8] Maurice Margenstern, Gheorghe Păun, and Yurii Rogozhin. On the power of (molecular) crowd: Set-conditional string processing. In *Proceedings of AFL'02, the 10th International Conference on Automata and Formal Languages, Debrecen, Hungary, August 13–18, 2002*.

[9] Maurice Margenstern, Gheorghe Păun, Yurii Rogozhin, and Sergey Verlan. Context-free insertion-deletion systems. *Proceedings of DCFS 2003, the 5th Workshop on Descriptional Complexity of Formal Systems, Budapest, Hungary, July 12–14, 2003*.

[10] B. S. Galiukschov. Semicontextual grammars (in russian). *Mat. Logica i Mat. Ling.*, pages 38–50, 1981.

[11] S. Marcus. Contextual grammars. *Rev. Roum. Math Pures Appl.*, 14:1525–1534, 1969.