Annales Univ. Sci. Budapest., Sect. Comp. 25 (2005) 67-74

RESTRICTED INSERTION–DELETION SYSTEMS

I. Katsányi (Budapest, Hungary)

Abstract. In the past few years several paper showed that various generative mechanisms in formal language theory that used context-dependent insertions and deletions are capable of generating any recursively enumerable language [1-9]. Since such systems are also models of molecular computing, for practical reasons it is important to examine these systems in a restricted case, when 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 paper 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] as the model of construction of natural languages. It is an important model of formal languages on its own right, but it gained even more importance by 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.

2. Preliminaries

When not stated otherwise, we will use the standard notations used in the theory of formal languages (see for example [11, 12]). As usual, we denote the free monoid generated by a finite set X by X^*. We call X as *alphabet*, elements of X as *letters* or *symbols*, and elements of X^* as *words*. The length of a word u is denoted by $|u|$. The notation of the empty word is λ. We will use the following lemma (Penttonen normal form, [13]):

Lemma 1. *For each G=(N,T,S,P) grammar there exists a grammar* $G' = (N', T', S', P')$ *for which* $L(G) = L(G')$*, and the contextfree rules of* P' *have the form* $X \to x$*, where* $X \in N'$*,* $x \in (N' \cup T)^*$*,* $|x| \leq 2$*, and the non-contextfree rules of* P' *have the form* $XY \to XZ$*, where* $X, Y, Z \in N'$*.*

An *insertion-deletion system* is a construction

$$\gamma = (V, T, A, I, D),$$

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 exactly when *one* of the following two cases occurs:

- $x = x_1 u v x_2, \quad y = x_1 u z v x_2, \quad x_1, x_2 \in V^* \quad \text{and} \quad (u, z, v) \in I,$
- $x = x_1 u z v x_2, \quad y = x_1 u v x_2, \quad x_1, x_2 \in V^* \quad \text{and} \quad (u, z, v) \in D.$

Let $\Longrightarrow_\gamma^*$ be the reflexive, transitive closure of $\Longrightarrow_\gamma$. The language generated by γ is

$$L(\gamma) = \{w \in T^* | x \Longrightarrow_\gamma^* w, x \in A\}.$$

In papers [1,2,4,6,7,9] it is shown that every recursively enumerable language can be generated by an insertion-deletion system.

In addition to the former construction, in [4] the *restricted insertion-deletion systems* are 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, $A \subseteq \{a, c\}^*$ 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 differences between the regular insertion-deletion systems are the following:

- the alphabet V consists of two symbols only,
- the terminal alphabet T is independent of the alphabet of the systems, the inverse of morphism h is used to map the words of the system to words of the terminal alphabet.
- The insertion and deletion rules are of the form (u, c^i, v), where $u, v \in \in \{a, c\}^*, i \geq 0$.

The relation $\Longrightarrow_\gamma$ is defined on the usual way. The language generated by γ is

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

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. That is why we do not mind contexts of $(aca)^*$ in this model.

In [4] it was shown that this construction is also very powerful: systems of this kind can generate any recursively enumerable language.

3. Restricted insertion-deletion systems

In this section we define a new kind of insertion-deletion system that has additional constraints comparing to the regular model described earlier. Although it differs from the existing restricted insertion-deletion system, since from now on we deal only with our constructions, it will not cause ambiguity. After the definitions it is showed, that in spite of the restrictions, this model is capable of universal computation, too.

A *restricted insertion-deletion system* is a construction

$$\gamma = (V, T, h, A, I, D),$$

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 on the usual way. When there is no chance of ambiguity, we use $\Longrightarrow$ instead of $\Longrightarrow_\gamma$. The morphism h is needed to define languages over an arbitrary finite alphabet. The language generated by γ is

$$L(\gamma) = h^{-1}(\{w \in V^* | z \Longrightarrow^* w, \quad \text{where} \quad z \in A\}).$$

The main result of this paper is the following

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

Proof. Being a consequence of the Church thesis, we do not prove that the family of languages generated by restricted insertion-deletion system is contained in the family of recursively enumerable languages. For the other inclusion, 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 $G = (N, T, S, P)$. By Lemma 1, we may assume that G is in the normal form of Penttonen. Moreover, let us substitute all rules of the form $X \to \alpha \in P$ $(\alpha \in N \cup T)$ by rules $X \to \alpha Z$, $Z \to \lambda$, where Z is a symbol that appeared nowhere earlier. Hence the rules of P have one of the following forms:

1. $X \to YZ$, where $X \in N$ and $Y, Z \in N \cup T$,
2. $X \to \lambda$, where $X \in N$,
3. $XY \to XZ$, where $X, Y, Z \in N$.

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. Let $P = \{r_1, r_2, \ldots, r_s\}$. 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 $h(\alpha_i) = g(\alpha_i)$ $(\alpha_i \in T)$. The following restricted insertion-deletion system will generate L:

$$\gamma = (\{a, c\}, Th, \{aca\}, I, D),$$

where the elements of I and D are listed below $(i, j, k \in [1, n], q \in [1, s])$:

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

3. For each rule $r_q : \alpha_i\alpha_j \to \alpha_i\alpha_k$
   a. $(ac^iaac^j, c^{q+n-j}, a) \in I$, and
   b. $(ac^k, c^{q+n-k}, a) \in D$.

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. In that latter case we get somewhere 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 a hence obtained 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 connected 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 usage of its insertion pair.

To prove that $L(G) = L(\gamma)$, let us consider the following lemmas.

Lemma 2. *For each* $u \in (N \cup T)^*, S \Longrightarrow_G^* u$ *implies* $aca \Longrightarrow_\gamma^* g(u)$.

This lemma has the consequence that $L(G) \subseteq L(\gamma)$, because $L(G)$ consists of words $u \in T^*$, for which $S \Longrightarrow_G^* u$, and for such u the morphisms h and g are equal. The lemma itself may be proved by induction on the number of derivation steps of u. Since $g(S) = g(\alpha_1) = aca$, the assertion is true for derivations of length zero. Now let us suppose that it is true for derivations not longer than $m \geq 0$. Let us consider a derivation of length $m + 1$:

$$S \Longrightarrow_G^m u_1u_2u_3 \Longrightarrow_G u_1u_2'u_3,$$

where the last used rule is $u_2 \to u_2'$. By our induction hypothesis

$$aca \Longrightarrow_\gamma^* g(u_1u_2u_3) = g(u_1)g(u_2)g(u_3).$$

We have three cases:

1. If the last used rule was of the form $\alpha_i \to \alpha_j\alpha_k$, then $u_2 = \alpha_i, u_2' = \alpha_j\alpha_k$, $g(u_2) = ac^ia$. Using $(ac^i, c^{q+n-i}aac^k, a) \in I$ and $(ac^j, c^{q+n-j}, a) \in D$ we get

$$g(u_1)ac^iag(u_3) \Longrightarrow_\gamma g(u_1)ac^{q+n}aac^kag(u_3) \Longrightarrow_\gamma$$

$$\Longrightarrow_\gamma g(u_1)ac^jaac^kag(u_3) = g(u_1u_2'u_3).$$

2. If the last used rule was of the form $\alpha_i \to \lambda$, then $u_2 = \alpha_i$, $u_2' = \lambda$, $g(u_2) = ac^i a$. Using the rule $(\lambda, ac^i a, \lambda) \in D$ we get

$$g(u_1)ac^i ag(u_3) \Longrightarrow_\gamma g(u_1)g(u_3) = g(u_1 u_2' u_3).$$

3. If the last used rule was of the form $\alpha_i\alpha_j \to \alpha_i\alpha_k$, then $u_2 = \alpha_i\alpha_j$, $u_2' = \alpha_i\alpha_k$, $g(u_2) = ac^i aac^j a$. Using rules $(ac^i aac^j, c^{q+n-j}, a) \in I$ and $(ac^k, c^{q+n-k}, a) \in D$ we get

$$g(u_1)ac^i aac^j ag(u_3) \Longrightarrow_\gamma g(u_1)ac^i aac^{q+n} ag(u_3) \Longrightarrow_\gamma$$

$$\Longrightarrow_\gamma g(u_1)ac^i aac^k ag(u_3) = g(u_1 u_2' u_3).$$

By that $aca \Longrightarrow_\gamma^* g(u_1 u_2' u_3)$, and the proof of the Lemma 2 is finished.

The following lemma is a formalization of the idea given after the definition of γ.

Lemma 3. *For each derivation sequence in γ that ends in a word in the domain of the morphism g, there exists also a derivation in γ that ends in the same word, such that the usage of an insertion rule belonging to a rule of G is immediately followed by a deletion rule belonging to the same rule.*

Let us consider an arbitrary derivation sequence $aca \Longrightarrow_\gamma \ldots \Longrightarrow_\gamma v$ such that there exists $u = g^{-1}(v)$. Suppose that by the usage of an insertion rule belonging to a rule $r_q \in P (q \in [1, s])$ the subword $ac^{n+q}a$ appears in the derived word. It cannot be a subword of v, because there is no word that g maps to a word that has a subword like this. Hence, this subword must be altered somehow. By checking the rules of γ it is clear, that no rules depend on a context containing this subword, and no deletion rule may alter this subword with the exception of the deletion rule belonging to r_q. Hence we must use this rule on this subword, and the result of the derivation will not change, if we do this immediately after the usage of the rule that introduced this subword.

The following lemma has the immediate consequence that $L(G) \supseteq L(\gamma)$.

Lemma 4. *If $aca \Longrightarrow_\gamma^* v$ such that there exists $u = g^{-1}(v)$, then $S \Longrightarrow_G^* \Longrightarrow_G^* u$.*

Since by Lemma 3 there is a derivation sequence $aca \Longrightarrow_\gamma \ldots \Longrightarrow_g v$, where the usage of an insertion rule belonging to a rule of G is immediately followed by a deletion rule belonging to the same rule, we can prove this lemma by mathematical induction very similar to the one used in Lemma 2. The details are omitted here.

4. Conclusions

In this paper we showed that the insertion-deletion systems remain powerful enough to generate any recursively enumerable language 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] **Martin-Vide C., Păun G. and Salomaa A.,** Characterizations of recursively enumerable languages by means of insertion grammars, *Theoretical Computer Science,* **205** (1998), 195-205.

[2] **Kari L. and Thierrin G.,** Contextual insertions/deletions and computability, *Information and Computation,* **131** (1) (1996), 47-61.

[3] **Daley M., Kari L., Gloor G. and Siromoney R.,** Circular contextual insertions/deletions with applications to biomolecular computation, *SPIRE/CRIWG,* 1999, 47-54.

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

[5] **Kari L.** *On insertion and deletion in formal languages,* PhD thesis, University of Turku, 1991.

[6] **Păun G., Rozenberg G. and Salomaa A.,** *DNA computing. New computing paradigms,* Springer, 1998.

[7] **A. Takahara and T. Yokomori,** On the computational power of insertion-deletion systems, *Proc. of the 8th Int. Workshop on DNA-based Computers, Sapporo, Japan, June 10-13, 2002,* and LNCS **2568**, 2003, 269-280.

[8] **Margenstern M., Păun G., Rogozhin Y.,** On the power of (molecular) crowd: Set-conditional string processing, *Proc. of AFL'02, Budapest, Hungary, August 13-18, 2002.*

[9] **Margenstern M., Păun G., Rogozhin Y. and Verlan S.,** Context-free insertion-deletion systems, *Proc. of DCFS 2003, The 5th Workshop on Descriptional Complexity of Formal Systems, Budapest, Hungary, July 12-14, 2003,* and *Theoretical Computer Science,* **330** (2) (2005), 339-348.

[10] **Галюкшов Б.С.,** Полуконтекстные грамматики, *Мат. логика и мат. лингвистика (Калинин),* 1981, 38-50. (Galiukshov B.S., Semi-contextual grammars, *Mat. logika i mat. ling.,* 1981, 38-50. (in Russian))

[11] **Rozenberg G. and Salomaa A.,** *Handbook of formal languages,* Springer, 1997.

[12] **Salomaa A.,** *Formal languages,* Academic Press, New York, 1973.

[13] **Penttonen M.,** One-sided and two-sided context in formal grammar, *Information and Control,* **25** (4) (1974), 371-392.

(Received March 1, 2004)

I. Katsányi
Department of Algorithms and Applications
Eötvös Loránd University
Pázmány Péter s. 1/C
H-1117 Budapest, Hungary
kacsa@ludens.elte.hu