LETTER

On Computational Power of Insertion-Deletion Systems without Using Contexts

Sadaki HIROSE^{†a)} and Satoshi OKAWA^{††}, Members

SUMMARY An Insertion-Deletion system, first introduced in [1], is a theoretical computing model in the DNA computing framework based on insertion and deletion operations. When insertion and deletion operations work together, as expected, they are very powerful. In fact, it has been shown that even the very restricted Insertion-Deletion systems can characterize the class of recursively enumerable languages [1]–[4]. In this paper, we investigate the computational power of Insertion-Deletion systems and show that they preserve the computational universality without using contexts.

key words: Insertion-Deletion system, DNA computing, formal language, automaton

1. Introduction

There are obvious limits to miniaturize the current silicon-based computers. For a drastic innovation, it has been suggested that the basic components should go to the molecular level. DNA computing is one of the recent developments arising from this suggestion.

The Insertion-Deletion system is a theoretical computing model in the DNA computing framework based on two operations, insertion and deletion, which have been considered in formal language theory. It was first introduced in [1], following the systematic study of insertion and deletion operations in [5] (where, however, generative mechanisms based on these two operations were not considered).

By applying an insertion operation denoted by $(u, \lambda/\beta, v)$, β can be inserted in between contexts u and v . Conversely, by applying a deletion operation denoted by $(u, \alpha/\lambda, v)$, α can be deleted from a substring $u\alpha v$, where u and v are contexts.

When insertion and deletion operations work together, as expected, they are very powerful. In fact, it has been shown that even the very restricted Insertion-Deletion systems can characterize the class of recursively enumerable languages [1]–[4].

In this paper, we investigate the computational power of Insertion-Deletion systems and show that they preserve the computational universality without using contexts.

2. Definition of Insertion-Deletion System

An Insertion-Deletion system is a construct

$$\gamma = (V, T, A, R),$$

where V is an alphabet, T is a subset of V , A is a finite set of strings over V , and R is a finite set of triples of the form $(u, \alpha/\beta, v)$, where $u, v \in V^*$, $(\alpha, \beta) \in (V^+ \times \{\lambda\}) \cup (\{\lambda\} \times V^+)$. The elements of T are the terminal symbols, those of A are axioms, the triples in R are insertion/deletion rules. Intuitively, an insertion rule $(u, \lambda/\beta, v)$ corresponds to a rewriting rule $uv \rightarrow u\beta v$, while a deletion rule $(u, \alpha/\lambda, v)$ corresponds to a rewriting rule $u\alpha v \rightarrow uv$.

For $x, y \in V^*$, we can write $x \Rightarrow_\gamma y$ if y can be obtained from x by applying an insertion rule or a deletion rule as defined above. Explicitly, this means that one of the following case holds:

1) $x = x_1 u v x_2$, $y = x_1 u \beta v x_2$, for some $x_1, x_2 \in V^*$ and $(u, \lambda/\beta, v) \in R$,

2) $x = x_1 u \alpha v x_2$, $y = x_1 u v x_2$, for some $x_1, x_2 \in V^*$ and $(u, \alpha/\lambda, v) \in R$.

We denote by $\Rightarrow_\gamma^*$ the reflexive and transitive closure of the relation $\Rightarrow_\gamma$.

The language generated by γ is defined by

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

We say that an Insertion-Deletion system $\gamma = (V, T, A, R)$ is of weight $(m, n; p, q)$ if

$$m = \max\{|\beta| \mid (u, \lambda/\beta, v) \in R\},$$

$$n = \max\{|\alpha|, |\lambda| \mid (u, \lambda/\beta, v) \in R\},$$

$$p = \max\{|\alpha| \mid (u, \alpha/\lambda, v) \in R\},$$

$$q = \max\{|\lambda|, |\alpha| \mid (u, \alpha/\lambda, v) \in R\}.$$

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

Note that $INS_{m'}^{n'} DEL_{p'}^{q'} \subseteq INS_m^n DEL_p^q$ clearly holds for all $0 \leq m' \leq m$, $0 \leq n' \leq n$, $0 \leq p' \leq p$, $0 \leq q' \leq q$.

3. Main Result

It is known that the very restricted (reduced weight) Insertion-Deletion systems can characterize the class of

Manuscript received December 16, 2004.

[†]The author is with the Faculty of Engineering, Toyama University, Toyama-shi, 930-8555 Japan.

^{††}The author is with the Faculty of Computer Science and Engineering, The University of Aizu, Aizu-wakamatsu-shi, 965-8580 Japan.

a) E-mail: hirose@iis.toyama-u.ac.jp

DOI: 10.1093/ietisy/e88-d.8.1993

recursively enumerable languages (RE). In [2], it is shown that $INS_1^2 DEL_1^1 = RE$, $INS_2^1 DEL_2^0 = RE$, and $INS_3^2 DEL_2^0 = RE$ hold. In [3], it is shown that $INS_1^1 DEL_2^0 = RE$ holds.

It was not known how large the family $INS_1^1 DEL_1^1$ is, but it was conjectured in [2] that at least one of the parameters n (insertion contexts), p (deleted strings), or q (deletion contexts) must be at least two in order to generate non-context-free languages. However, recently in [4], it is shown that $INS_1^1 DEL_1^1 = RE$ holds.

In this paper, as our main result, we shall prove that $INS_3^0 DEL_2^0 = RE$ holds. It means that Insertion-Deletion systems preserve the computational universality without using contexts.

At first, we shall prove the following theorem.

Theorem 1: $INS_4^0 DEL_2^0 = RE$.

Proof. Of course, we have to prove only the relation $INS_4^0 DEL_2^0 \supseteq RE$.

Let $L \subseteq T^*$ be an arbitrary recursively enumerable language, and $G = (N, T, S, P)$ be a phrase structure grammar such that $L = L(G)$. Without loss of generality, we suppose the grammar G is given in the Kuroda's normal form.

In Kuroda's normal form, the set of production rules P is constructed from the following four types of rules:

1. $r : X \rightarrow YZ$,
2. $r : XY \rightarrow ZW$,
3. $r : X \rightarrow a$,
4. $r : X \rightarrow \lambda$,

where $X, Y, Z, W \in N$, $a \in T$, and r is a label of the production rule. We assume the rules are labeled in a one-to-one manner. So, each production rule can be distinguished from the others by the label r .

We construct the Insertion-Deletion system

$$\gamma = (V, T, A, R),$$

where

$$V = N \cup T \cup \{\check{X} | X \in N\},$$

$$A = \{S\},$$

and the set R is constructed as follows.

- $r_1 : (\lambda, \lambda/\check{X}YZ, \lambda)$
(for $r : X \rightarrow YZ \in P$),
- $r_2 : (\lambda, \lambda/\check{Y}\check{X}ZW, \lambda)$
(for $r : XY \rightarrow ZW \in P$),
- $r_3 : (\lambda, \lambda/\check{X}a, \lambda)$
(for $r : X \rightarrow a \in P$),
- $r_4 : (\lambda, \lambda/\check{X}, \lambda)$
(for $r : X \rightarrow \lambda \in P$),
- $r_5 : (\lambda, X\check{X}/\lambda, \lambda)$
(for $X \in N$).

Clearly, the weight of the Insertion-Deletion system γ defined above is $(4, 0; 2, 0)$.

To complete the proof, we shall show the equality $L(G) = L(\gamma)$, that is, we shall prove $L(G) \subseteq L(\gamma)$ and $L(\gamma) \subseteq L(G)$.

(The proof of $L(G) \subseteq L(\gamma)$)

Each derivation step $w \Rightarrow_G w'$ in G is simulated in γ by a derivation $w \xRightarrow{*}_{\gamma} w'$, using rules r_i associated as above with the production rule in P used in the derivation $w \Rightarrow_G w'$.

1. Assume that $w = w_1 X w_2$, $w' = w_1 Y Z w_2$, for $r : X \rightarrow YZ \in P$. Then we successively obtain:

$$\begin{aligned} w_1 X w_2 & \xRightarrow{\gamma} w_1 X \check{X} Y Z w_2 && \text{by rule } r_1 \\ & \xRightarrow{\gamma} w_1 Y Z w_2 && \text{by rule } r_5. \end{aligned}$$

This shows that γ can simulate the production rule $r : X \rightarrow YZ$ in P .

2. Assume that $w = w_1 X Y w_2$, $w' = w_1 Z W w_2$, for $r : XY \rightarrow ZW \in P$. Similarly, we successively obtain:

$$\begin{aligned} w_1 X Y w_2 & \xRightarrow{\gamma} w_1 X Y \check{Y} \check{X} Z W w_2 && \text{by rule } r_2 \\ & \xRightarrow{\gamma} w_1 X \check{X} Z W w_2 && \text{by rule } r_5 \\ & \xRightarrow{\gamma} w_1 Z W w_2 && \text{by rule } r_5. \end{aligned}$$

This shows that γ can simulate the production rule $r : XY \rightarrow ZW$ in P .

3. Assume that $w = w_1 X w_2$, $w' = w_1 a w_2$, for $r : X \rightarrow a \in P$. Similarly, we successively obtain:

$$\begin{aligned} w_1 X w_2 & \xRightarrow{\gamma} w_1 X \check{X} a w_2 && \text{by rule } r_3 \\ & \xRightarrow{\gamma} w_1 a w_2 && \text{by rule } r_5. \end{aligned}$$

This shows that γ can simulate the production rule $r : X \rightarrow a$ in P .

4. Assume that $w = w_1 X w_2$, $w' = w_1 w_2$, for $r : X \rightarrow \lambda \in P$. We obtain:

$$\begin{aligned} w_1 X w_2 & \xRightarrow{\gamma} w_1 X \check{X} w_2 && \text{by rule } r_4 \\ & \xRightarrow{\gamma} w_1 w_2 && \text{by rule } r_5. \end{aligned}$$

This shows that γ can simulate the production rule $r : X \rightarrow \lambda$ in P .

Any terminal string generated by G is generated by γ from S in the step-by-step manner described above.

(The proof of $L(\gamma) \subseteq L(G)$)

Consider any string w generated by γ ; initially we have $w = S$. We can apply to it a rule r_i ($1 \leq i \leq 4$), associated with type i rule in P , or a deletion rule r_5 . As they do not use any contexts, a rule r_i ($1 \leq i \leq 4$) can be applied in any positions of w and a deletion rule r_5 may be applied to any occurrences $X\check{X}$ of w .

Assume that we apply r_1 , associated with the type 1

rule $r : X \rightarrow YZ$ in P , in some position of w . The rule r_1 derives $\check{X}YZ$ in the position of w . Since the terminal string should be obtained, the symbol $\check{X}$ has to be deleted, with the symbol X in the left of $\check{X}$, using a deletion rule r_5 . So the applications of rules r_1 and r_5 do not affect the string w more than the type 1 rule $r : X \rightarrow YZ$ does.

Similarly, rules r_3 and r_4 do not affect the string w more than the type 3 rule $r : X \rightarrow a$ and type 4 rule $r : X \rightarrow \lambda$ do, respectively.

When rule r_2 is applied, it derives $\check{Y}\check{X}ZW$ in w . The symbols $\check{Y}$ and $\check{X}$ have to be deleted, with the symbols Y and X , using deletion rule r_5 , respectively. Even if the deletion rules for $\check{Y}$ and $\check{X}$ might not be applied continuously, they do not affect the string w more than the type 2 rule $r : XY \rightarrow ZW$ does.

Consequently, γ can generate only strings in $L(G)$. $\square$

Introducing new symbols to two consecutive non-terminal symbols in N , we can reduce the parameter m (inserted strings) to 3.

Theorem 2: $INS_3^0 DEL_2^0 = RE$.

Proof. Basically, the idea of the proof is the same as in **Theorem 1**. So, we refine the Insertion-Deletion system γ of **Theorem 1**.

We add new symbols to V , that is, let

$$V = N \cup T \cup \{\check{X} | X \in N\} \cup \{\check{\#}_{XY} | X, Y \in N\} \cup \{\check{\#}_{XY} | X, Y \in N\}^\dagger.$$

Moreover, we replace insertion rule r_2 with the following r'_2 and r''_2 ,

$$\begin{aligned} r'_2 &: (\lambda, \lambda / \check{Y}\check{X}\check{\#}_{ZW}, \lambda), \\ r''_2 &: (\lambda, \lambda / \check{\#}_{ZW}ZW, \lambda) \\ &\quad (\text{for } r : XY \rightarrow ZW \in P), \end{aligned}$$

[†]Note that both $\check{\#}_{XY}$ and $\check{\#}_{XY}$ are one symbol corresponded with two consecutive non-terminal symbols XY in N .

and add deletion rule r_6 ,

$$\begin{aligned} r_6 &: (\lambda, \check{\#}_{XY}\check{\#}_{XY}/\lambda, \lambda) \\ &\quad (\text{for } X, Y \in N). \end{aligned}$$

The weight of the refined Insertion-Deletion system γ is $(3, 0; 2, 0)$.

It is trivial to show that insertion rules r'_2, r''_2 and deletion rule r_6 play the same role that insertion rule r_2 and deletion rule r_5 do. $\square$

It is not known how large the family $INS_2^0 DEL_2^0$ is. If $INS_2^0 DEL_2^0 = RE$ holds, we believe another technique is needed to show it.

Acknowledgement

This research has been supported by the Kayamori Foundation of Informational Science Advancement.

References

- [1] L. Kari and G. Thierrin, "Contextual insertion/deletion and computability," *Inf. Comput.*, vol.131, no.1, pp.47–61, 1996.
- [2] L. Kari, Gh. Păun, G. Thierrin, and S. Yu, "At the crossroads of DNA computing and formal languages: Characterizing RE using Insertion-Deletion systems," *Proc. 3rd DIMACS Workshop on DNA Based Computing*, pp.318–333, Philadelphia, 1997.
- [3] Gh. Păun, G. Rozenberg, and A. Salomaa, *DNA Computing*, Springer, 1998.
- [4] A. Takahara and T. Yokomori, "On the computational power of Insertion-Deletion systems," *Proc. 8th International Meeting on DNA Based Computing*, pp.139–150, Sapporo, 2002; also in *LNCS 2568*, pp.269–280, 2003.
- [5] L. Kari, *On Insertion and Deletion in Formal Languages*, Ph.D. Thesis, University of Turku, 1991.