Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

Minimizing of the only-insertion insdel systems

MIN Yong (闵 勇)[†1], JIN Xiao-gang (金小刚)[†‡1], SU Xian-chuang (苏先创)[2], PENG Bo (彭 博)[1]

([1]AI Institute, School of Computer Science, Zhejiang University, Hangzhou 310027, China)
([2]School of Software Engineering, Zhejiang University, Hangzhou 310027, China)
[†]E-mail: myong@zju.edu.cn; xiaogangj@cise.zju.edu.cn

Received Mar. 10, 2005; revision accepted June 8, 2005

Abstract: A more recent branch of natural computing is DNA computing. At the theoretical level, DNA computing is powerful. This is due to the fact that DNA structure and processing suggest a series of new data structures and operations, and to the fact of the massive parallelism. The insertion-deletion system (insdel system) is a DNA computing model based on two genetic operations: insertion and deletion which, working together, are very powerful, leading to characterizations of recursively enumerable languages. When designing an insdel computer, it is natural to try to keep the underlying model as simple as possible. One idea is to use either only insertion operations or only deletion operations. By helping with a weak coding and a morphism, the family $\mathrm{INS}_4^7\mathrm{DEL}_0^0$ is equal to the family of recursively enumerable languages. It is an open problem proposed by Martin-Vide *et al.* on whether or not the parameters 4 and 7 appearing here can be replaced by smaller numbers. In this paper, our positive answer to this question is that $\mathrm{INS}_2^4\mathrm{DEL}_0^0$ can also play the same role as insertion and deletion. We suppose that the $\mathrm{INS}_2^4\mathrm{DEL}_0^0$ may be the least only-insertion insdel system in this situation. We will give some reasons supporting this conjecture in our paper.

Key words: DNA computing, Insdel, Only insertion
doi:10.1631/jzus.2005.A1021 **Document code:** A **CLC number:** TP384

INTRODUCTION

In formal language theory, most language generative devices are based on the operation of rewriting, but, two other operations can also play the same role: insertion and deletion operation, originally motivated mainly from linguistics. Because the phenomena that is similar to these operations has been found in DNA operations, these two language operations have come to the special attention of researchers in the DNA computing area.

For a given string $xuvy$, an insertion operation (u,w,v) (where u is former context, v is back context, w is the inserted string), produces a new string $xuwvy$. Conversely, by a deletion operation of (u,w,v), we obtain $xuvy$ from $xuwvy$. Theoretically, both insertion and deletion operations can be performed by biomolecular techniques. The computational powers of insertion-deletion systems have been researched; you can find major results in this field in (Kari and Thierrin, 1996; Kari *et al.*, 1997; Ehrenfeucht *et al.*, 1998; Takahara and Yokomori, 2002; Margenstern *et al.*, 2005).

This paper will focus on the theoretical issues on the computational capability of insertion-deletion systems that use only insertion operation, called only-insertion insdel system, and solve the open problems proposed in (Martin-Vide *et al.*, 1998).

PRELIMINARIES

An insdel system is a construct:

$$\Gamma = (V, T, AX, R) \tag{1}$$

where V is an alphabet, $T\subseteq V$, AX is a finite language 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 terminal symbols, those of AX

[‡] Corresponding author

are axioms, and the triples in R are insertion-deletion rules. The meaning of $(u,\lambda/\beta,v)$ is that β can be inserted in between u and v; the meaning of $(u,\alpha/\lambda,v)$ is that α can be deleted from the context (u,v). Stated otherwise, $(u,\lambda/\beta,v)$ corresponds to the rewriting rule $uv\rightarrow u\beta v$, and $(u,\alpha/\lambda,v)$ corresponds to the rewriting rule $u\alpha v\rightarrow uv$. Consequently, for $x, y\in V^*$ we can write $x\Rightarrow y$ if y can be obtained from x by using an insertion or a deletion rule.

The language generated by Γ is defined by:

$$L(\Gamma)=\{\omega\in T^* \mid x\Rightarrow^* \omega, \text{ for some } x\in AX\} \qquad (2)$$

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

We say that an insdel system $\Gamma=(V, T, AX, R)$ is of weight $(n,m;p,q)$ if

$$\begin{aligned}
&n=\max\{|\beta| \mid (u,\lambda/\beta,v)\in R\},\\
&m=\max\{|u| \mid (u,\lambda/\beta,v)\in R \text{ or } (v,\lambda/\beta,u)\in R\},\\
&p=\max\{|\alpha| \mid (u,\alpha/\lambda,v)\in R\},\\
&q=\max\{|u| \mid (u,\alpha/\lambda,v)\in R \text{ or } (v,\alpha/\lambda,u)\in R\} \qquad (3)
\end{aligned}$$

We denote by $\mathrm{INS}_n^m\mathrm{DEL}_p^q$, for $n, m, p, q\geq 0$, the family of languages $L(\Gamma)$ generated by insdel systems of weight $(n',m';p',q')$ such that $n'\leq n$, $m'\leq m$, $p'\leq p$, $q'\leq q$. The meaning of INS_0^0 is that no insertion rule is used, and the meaning of DEL_0^0 is that no deletion rule is used.

Before precisely stating our main result, we need to introduce the following result of only insertion system in (Martin-Vide *et al.*, 1998):

Theorem 1 Each language $L\in RE$ can be written in the form $L=g(h^{-1}(L'))$, where g is a weak coding, h is a morphism, and $L'\in\mathrm{INS}_4^7\mathrm{DEL}_0^0$.

MAIN RESULT

Now, we present our main result on the generative power of only-insertion system of weight (2,4; 0,0):

Theorem 2 Each language $L\in RE$ can be written in the form $L=g(h^{-1}(L'))$, where g is a weak coding, h is a morphism, and $L'\in\mathrm{INS}_2^4\mathrm{DEL}_0^0$.

Proof Consider a language $L\subseteq T^*$, $L\in RE$ generated by a type-0 grammar $G=(N,T,P,S)$ in the Penttonen normal form, that is containing context-free rules $A\rightarrow x$ with $|x|\leq 2$, and non-context-free rules of the form $AB\rightarrow AC$, for $A,B,C\in N$. Without loss of generality we may assume that in each rule $A\rightarrow x_1x_2\in P$ we have $A\neq x_1$, $A\neq x_2$, $x_1\neq x_2$ (If necessary, we replace $A\rightarrow x_1x_2$ with $A\rightarrow A'$, $A'\rightarrow x_1x_2'$, $x_2'\rightarrow x_2$, where A', x_2' are new symbols). Similarly, we may assume that for each rule $AB\rightarrow AC\in P$ we have $A\neq B$, $A\neq C$, $B\neq C$. Moreover, by replacing each rule $A\rightarrow x\in P$, $x\in N\cup T$, by $A\rightarrow xC$, $C\rightarrow\lambda$, we obtain an equivalent grammar. Hence, we may assume that the rules in P are of the following three forms (Păun *et al.*, 1998):

1. $A\rightarrow x_1x_2$, for $x_1,x_2\in N\cup T$ such that $A\neq x_1$, $A\neq x_2$, $x_1\neq x_2$,
2. $A\rightarrow\lambda$,
3. $AB\rightarrow AC$, for $A,B,C\in N$ such that $A\neq B$, $A\neq C$, $B\neq C$.

We construct the only-insertion system with smaller parameters $\mathrm{INS}_2^4\mathrm{DEL}_0^0$: $\Gamma=(V, T, AX, R)$, where V is an alphabet, T is the set of terminal symbols, $T\subseteq V$, AX is set $\{\theta\theta\theta\theta S\theta\theta\theta\theta\}$, R is a rule set. And $\theta\in V$ is initial symbol for providing context; $\#\in V$, if followed by #, the nonterminal symbols have been deleted; we denote $H=\{[r_x], (r_x), \backslash r_x\backslash\}$ $(x=1,2,3,4)$, $H\subseteq V$, the new symbols $[r_x]$, (r_x), $\backslash r_x\backslash$ are assistant symbols for using rules (x=1,2,3,4 is the index of rule group); N is the set of nonterminal symbols except for #, θ and elements in H. Let $M=N\cup H$ and $W=N\cup T$.

We also suppose that A, B, C are the arbitrary elements from N, α_i is an arbitrary element from V, β is an arbitrary element from M, and δ is an arbitrary element from W in the following text. The set R is constructed as follows.

Rule group 1 For each rule $A\rightarrow\lambda\in P$, we consider the rule:

$$(\mathrm{r.1.1}): (\alpha_1A, \lambda/\#, \alpha_2\alpha_3),$$

Rule group 2 For each rule $AB\rightarrow AC$ in the Penttonen normal form with $A,B,C\in N$, we consider the following insertion rules:

$$(\mathrm{r.2.1}): (\alpha_1AB, \lambda/\#C, \alpha_2\alpha_3),$$

In the rules 1 and 2, the α_1, α_2, α_3 must satisfy the following conditions: (1) $\alpha_1\notin\{[r_3], [r_4], (r_4)\}$; (2)

$\alpha_2 \neq \#$; (3) if $\alpha_2 \in \{(r_3), \backslash r_3 \backslash\}$, then α_3 must be #.

Rule group 3 For each rule $A \rightarrow x_1x_2$ in the Penttonen normal form with $A \in N$ and $x_1, x_2 \in N \cup T$, we consider the following rules:

(r.3.1): $(\alpha_1 A, \lambda/(r_3), \alpha_2)$, for $\alpha_1 \notin \{[r_4], (r_4)\}, \alpha_2 \notin \{(r_3), \#\}$;
(r.3.2): $(\alpha_1, \lambda/[r_3]\backslash r_3\backslash, A(r_3))$, for $\alpha_1 \neq \backslash r_3 \backslash$;
(r.3.3): $([r_3], \lambda/x_1x_2, \backslash r_3 \backslash A)$;
(r.3.4): $(x_1x_2\backslash r_3\backslash A, \lambda/\#, (r_3))$;
(r.3.5): $(\#(r_3), \lambda/\#, \alpha_1)$, for $\alpha_1 \neq \#$;
(r.3.6): $(\backslash r_3 \backslash, \lambda/\#, A\#(r_3)\#)$;
(r.3.7): $([r_3], \lambda/\#, x_1x_2\backslash r_3\backslash\#)$.

Rule group 4 we construct swapping rules as followings:

(r.4.1): $(\alpha_1\alpha_2, \lambda/[r_4], A\beta\#)$, for $\alpha_2 \notin \{[r_3], [r_4]\}$, $\alpha_1\alpha_2 \neq [r_3]\delta$;
(r.4.2): $([r_4]A\beta\#, \lambda/(r_4)A, \alpha_1\alpha_2)$, for $\alpha_1\alpha_2 \neq (r_4)\delta$;
(r.4.3): $([r_4]A, \lambda/\#\backslash r_4\backslash, \beta\#(r_4))$;
(r.4.4): $(\backslash r_4 \backslash \beta\#(r_4), \lambda/\#, \alpha_1)$, for $\alpha_1 \neq \#$;
(r.4.5): $(\backslash r_4 \backslash, \lambda/\#, \beta\#(r_4)\#)$;
(r.4.6): $([r_4], \lambda/\#, A\#\backslash r_4\backslash\#)$.

For $\omega \in W$, we consider string "$\omega\#$" as a symbol b_ω. Let K be the set of these symbols. We define the morphism (Păun *et al.*, 1998):

h: $(K \cup T \cup \{\theta\})^* \rightarrow V^*$, by:
$h(b_\omega) = \omega\#$;
$h(a) = a$, $a \in T$;
$h(\theta) = \theta$.

Consider also the weak coding (Păun *et al.*, 1998):

g: $(K \cup T \cup \{\theta\})^* \rightarrow T^*$, by:
$g(b_\omega) = \lambda$;
$g(a) = a$, $a \in T$;
$g(\theta) = \lambda$.

We obtain: $L(G) = g(h^{-1}(L(\Gamma)))$.

The rule groups 1, 2 and 3 simulate the three forms in the Penttonen normal form. The fourth rule is used for the purpose of preparing the current string for using of other rule groups in future. This is done as follows:

For string "$A\beta\#B$", we cannot apply rule of group 2 for it directly. In fact, "$A\beta\#B$" is equal to "AB", because β is marked deletion. The rules of group 4 can move an unmarked nonterminal symbol A across marked blocks (deleted) to right. Once pairs AB can be created, the rule group 2 can be applied.

In order to assure the performing of rules form r.3.1 to r.3.7 and form r.4.1 to r.4.6 is continuous and complete, we use assistant symbols $[r_x]$, (r_x), $\backslash r_x \backslash$. We call the area charging in applying a group of rules is a frame. $[r_x]$, (r_x), $\backslash r_x \backslash$ respectively marking the head, tail and middle of frame. These symbols that are temporary, when the executing of rule group is over, are marked by #.

In order to prove the equality $L(G) = g(h^{-1}(L(\Gamma)))$ we shall first prove that rules are doing what we have said that they are supposed to do (in this way we obtain the inclusion $\subseteq$), then we shall prove that they cannot do anything else (that is, also $\supseteq$ is true).

$L(G) \subseteq g(h^{-1}(L(\Gamma)))$

Claim 1 When using rule group 1, the occurrence of A in the derived string is unmarked, but it is marked with # in the resulting string.

The fact that A is unmarked in the original string to which the rule is applied is ensured by α_2, which is different from #. After the rule has been applied, the marker # is inserted, so the A is marked with #.

Claim 2 When using rule group 2, the occurrence of B in the derived string is unmarked, but it is marked with # in the resulting string, where each symbol of C is also unmarked.

Because α_2 is not #, the A in the original string and C in the resulting string are unmarked. As we obtain the substring "$AB\#C$", obviously, B has been marked.

Claim 3 When using rule group 3 (r.3.*i*), $1 \leq i \leq 7$, associated with a rule: $A \rightarrow x_1x_2$ in the Penttonen normal form, then the occurrence of A in the derived string is unmarked, but it is marked with # in the resulting string, where x_1 and x_2 are unmarked.

Starting with string $\alpha_1\alpha_2 A\alpha_3\alpha_4$, by using the third rule orderly fashion, the track of changing of this string is (String between the | is the frame of the rule):

$$\alpha_1\alpha_2|A|\alpha_3\alpha_4$$
$$\downarrow$$
$$\alpha_1\alpha_2|A(r_3)|\alpha_3\alpha_4$$
$$\downarrow$$

$$\begin{array}{c}\alpha_1\alpha_2|[r_3]\backslash r_3\backslash A(r_3)|\alpha_3\alpha_4\\ \downarrow\\ \alpha_1\alpha_2|[r_3]x_1x_2\backslash r_3\backslash A(r_3)|\alpha_3\alpha_4\\ \downarrow\\ \alpha_1\alpha_2|[r_3]x_1x_2\backslash r_3\backslash A\#(r_3)|\alpha_3\alpha_4\\ \downarrow\\ \alpha_1\alpha_2|[r_3]x_1x_2\backslash r_3\backslash A\#(r_3)\#|\alpha_3\alpha_4\\ \downarrow\\ \alpha_1\alpha_2|[r_3]x_1x_2\backslash r_3\backslash\#A\#(r_3)\#|\alpha_3\alpha_4\\ \downarrow\\ \alpha_1\alpha_2[r_3]\#x_1x_2\backslash r_3\backslash\#A\#(r_3)\#\alpha_3\alpha_4\end{array}$$

Therefore, the unmarked symbol A in the original string has been marked in the resulting string, and x_1x_2 has been inserted into the resulting string.

Claim 4 Starting from a legal string, the rule group 4 (r.4.*i*), $1\leq i\leq 6$, can replace substring "$A\beta\#$" with the block $(\beta\#)^*$ and ending with the symbol A.

Starting with the legal initial string "$A\beta\#$", we can easily prove this claim by listing its changing track (String between the | is the frame of the rule):

$$\begin{array}{c}\alpha_1\alpha_2|A\beta\#|\alpha_3\alpha_4\\ \downarrow\\ \alpha_1\alpha_2|[r_4]A\beta\#|\alpha_3\alpha_4\\ \downarrow\\ \alpha_1\alpha_2|[r_4]A\beta\#(r_4)A|\alpha_3\alpha_4\\ \downarrow\\ \alpha_1\alpha_2|[r_4]A\#\backslash r_4\backslash\beta\#(r_4)A|\alpha_3\alpha_4\\ \downarrow\\ \alpha_1\alpha_2|[r_4]A\#\backslash r_4\backslash\beta\#(r_4)\#A|\alpha_3\alpha_4\\ \downarrow\\ \alpha_1\alpha_2|[r_4]A\#\backslash r_4\backslash\#\beta\#(r_4)\#A|\alpha_3\alpha_4\\ \downarrow\\ \alpha_1\alpha_2[r_4]\#A\#\backslash r_4\backslash\#\beta\#(r_4)\#A\alpha_3\alpha_4\end{array}$$

Thus, starting from a legal string (initially, we have $\theta^4S\theta^4$), the rules of Γ can simulate the rules of G, producing legal strings. We use $umk(\Phi)$ to denote the string of the unmarked symbols in a legal string Φ generated by Γ.

Claim 5 If $x\Rightarrow^* y$ by using a rule in the first or second group or all seven rules in the third group, then $umk(x)\Rightarrow umk(y)$ by the corresponding rule in G.

Claim 6 If $x=g(h^{-1}(y))$, for some $y\in L(\Gamma)$, then y is a legal string and $x=umk(y)$, $y\in T^*$. Conversely, if $y\in L(\Gamma)$ and $umk(y)\in T^*$, then $umk(y)=g(h^{-1}(y))$.

The above two claims can be immediately deduced from the definitions of the morphisms g and weak coding h. These claims prove the inclusion $L(G)\subseteq g(h^{-1}(L(\Gamma)))$.

$L(G)\supseteq g(h^{-1}(L(\Gamma)))$

Claim 7 After using a rule (r.3.1), no other rule but (r.3.2) can be applied. Then, after (r.3.2), only (r.3.3) can be used. The following sequence must be (r.3.4), (r.3.5), (r.3.6) and (r.3.7).

First, we list the context string of every rule:

(c.1): $\alpha_1A\alpha_2$, for $\alpha_1\notin\{[r_4],(r_4)\},\alpha_2\notin\{(r_3),\#\}$;
(c.2): $\alpha_1A(r_3)$, for $\alpha_1\neq\backslash r_3\backslash$;
(c.3): $[r_3]\backslash r_3\backslash A$;
(c.4): $x_1x_2\backslash r_3\backslash A(r_3)$;
(c.5): $\#(r_3)\alpha_1$, for $\alpha_1\neq\#$;
(c.6): $\backslash r_3\backslash A\#(r_3)\#$;
(c.7): $[r_3]x_1x_2\backslash r_3\backslash\#$.

We name frame before using rule (r.3.*i*) *frame.i* (these frames can be found in proving of Claim 3), at the same time, we call context string of applying rule (r.3.*i*) c.*i*, $2\leq i\leq 7$. Comparing the list of context string with the list of frame, we find the c.*i* cannot be substring of any *frame.j*, except $i=j$, $2\leq i\leq 7$. So, the execution of rules is ordinal and unrepeatable.

We call string that contains no fragment of unfinished executing of rule groups 3 and 4 clear string. In the clear string, any assistant symbol τ, $\tau\in H$ has been marked with #. We can easily find only (r.3.1) can be applied in the clear string, because using (r.3.*i*), $2\leq i\leq 7$ need at least one unmarked assistant symbol. Therefore, the execution of rule group 3 must be from one to seven.

Claim 8 After using a rule (r.4.1), no other rule but (r.4.2) can be applied. Then, after (r.4.2), only (r.4.3) can be used. The following sequence must be (r.4.4), (r.4.5) and (r.4.6).

We can use the same method in proving Claim 7 to prove this claim, so, we do not repeat here.

Claim 9 The rule group one and two cannot apply in any frame in the processing of applying rule groups 3 and 4.

Consider three restrictions of rules 1 and 2, we can easily prove this claim by trying to apply rule in any frame of rule groups 3 and 4. You can find these frames during proving of Claims 3 and 4.

Claim 10 The execution of rule groups 3 and 4 cannot intermix.

We cannot apply (r.3.i), $2\leq i\leq 7$ in any frame of rule group 4, because using (r.3.i), $2\leq i\leq 7$ need one or more unmarked symbols: $[r_3]$, (r_3) or $\backslash r_3\backslash$. We also cannot apply (r.3.1) in any frame of rule group 4 because the symbol before A cannot be $[r_4]$ and (r_4). So, neither (r.3.i), $2\leq i\leq 7$ nor (r.3.1) can be used in the frames of rule group 4. For the same reason, neither (r.4.i), $2\leq i\leq 6$ nor (r.4.1) can be used in the frames of rule group 3. So, the execution of rule groups 3 and 4 cannot intermix.

From Claims 7 and 8, we know the execution of rule groups 3 and 4 is continuous. From Claims 9 and 10, we prove the execution of rule groups 3 and 4 is unbreakable. So, our insdel system cannot do anything that we do not want it to do. Therefore, $L(G)\supseteq g(h^{-1}(L(\Gamma)))$.

Now, $L(G)=g(h^{-1}(L(\Gamma)))$ has been proved.

CONCLUSION

Within the framework of only-insertion insdel systems of weight $(m, n; 0, 0)$, a natural problem is to find as small as possible values for $(m, n; 0, 0)$, such that the *RE* (the family of recursively enumerable languages) can be characterized by $\mathrm{INS}_n^m\mathrm{DEL}_0^0$ with the help of other operations. We have shown that the families of languages $\mathrm{INS}_2^4\mathrm{DEL}_0^0$, with the helping of a weak coding and a morphism, coincide with *RE*. In fact, the characterization result gives answers to the open questions posed in (Martin-Vide *et al.*, 1998).

The $\mathrm{INS}_2^4\mathrm{DEL}_0^0$ may be the least only-insertion insdel system in this situation (deletion by marking with the help of a weak coding and a morphism). We also give some reasons supporting this conjecture. The source of the restriction of weight is the swapping rule which is necessary in this situation.

Conjecture 1 $\min(m)=2$

When you want to copy a symbol to the new position in the string, in order to judge the symbol is inserted (state after copying) or original (state before copying), you must insert an assistant symbol at the same time. So, $m\geq 2$.

Conjecture 2 $\min(n)=4$

To ensure the executing sequence, we need to know if the two symbols after one are removed or not. The two removed symbols are represented by "$\beta\#\beta\#$", therefore, $n\geq 4$.

Here, we cannot give a strict proof of these conjectures. Moreover, if there are new methods for building only-insertion system, a more effective system may be found.

References

Ehrenfeucht, A., Păun, G., Rozenberg, G., 1998. On representing RE languages by internal contextual languages. *Theoretical Computer Science*, **205**(1-2):61-83.

Kari, L., Thierrin, G., 1996. Contextual insertion/deletion and computability. *Information and Computation*, **131**(1): 47-61.

Kari, L., Păun, G., Thierrin, G., Yu, S., 1997. At the Crossroads of DNA Computing and Formal Languages: Characterizing RE using Insertion-Deletion Systems. Proceedings of 3rd DIMACS Workshop on DNA Based Computing, Philadelphia, p.318-333.

Margenstern, M., Păun, G., Rogozhin, Y., Verlan, S., 2005. Context-free insertion-deletion systems. *Theoretical Computer Science*, **330**(2):339-348.

Martin-Vide, C., Păun, G., Salomaa, A., 1998. Characterizations of recursively enumerable language by means of insertion grammars. *Theoretical Computer Science*, **205**(1-2):195-205.

Păun, G., Salomaa, A., Rozenberg, G., 1998. DNA Computing: New Computing Paradigms. Springer-Verlag, p.187-216.

Takahara, A., Yokomori, T., 2002. On the Computational Powers of Insertion-Deletion Systems. 8th International Meeting on DNA Based Computers, Sapporo, p.139-150.