DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 48, 1999

At the Crossroads of DNA Computing and Formal Languages: Characterizing Recursively Enumerable Languages Using Insertion-Deletion Systems*

Lila Kari[a], Gheorghe Păun[b], Gabriel Thierrin[a], Sheng Yu[a]
[a]Department of Computer Science, University of Western Ontario
N6A 5B7 London, Ontario, Canada
E-mails: lila@csd.uwo.ca/ gab@csd.uwo.ca/ syu@csd.uwo.ca
[b]Institute of Mathematics of the Romanian Academy
PO Box 1-764, 70700 Bucureşti, Romania
E-mail: gpaun@imar.ro

Abstract

Several characterizations of recursively enumerable (RE) languages are presented, using insertion-deletion systems. Such a system generates the elements of a language by inserting and deleting words, according to their contexts (the insertion-deletion rules are triples (u, z, v), with the meaning that z can be inserted or deleted in/from the context (u, v)). Grammars based on insertion rules have already been considered in [10] with linguistic motivation. Insertion/deletion operations are also basic in DNA and RNA processing, [5]. Our results show that these operations, even with strong restrictions on the length of the contexts and/or on the length or on the form of the inserted/deleted words are computationally complete, that is, they can simulate the work of any Turing machine. A problem formulated in [30] is solved in this context.

1 Introduction

The insertion-deletion operation is fundamental in many areas dealing with symbol processing (in spite of the fact that the formal language theory is mainly

*Research supported by Grants OGP0007877 and OGP0041630 of the Natural Sciences and Engineering Research Council of Canada and by Project 11281 of the Academy of Finland

1991 *Mathematics Subject Classification.* Primary 68Q05, 68Q42, 68Q45, 03D10.

devoted to the study of *rewriting*). Natural and fruitful from a mathematical point of view (see, e.g., [14]), the insertion (adjoining) of a string or of a pair of strings in (to) a given string, depending on certain local conditions, is a basic operation in linguistics. The string-context interplay is probably one of the most important ingredients of all descriptive linguistic theories ([17]). Several generative mechanisms based on this operation have already been considered. In contextual grammars introduced in [18] (up-to-date details can be found in the forthcoming monograph [24]) pairs of strings are inserted, depending on the bracketed string. A similar effect is accomplished by the tree adjoining operation in tree adjunct grammars, [13]. A sort of dual model has been considered in [10]: strings are inserted in given associated contexts. In all these cases only context-sensitive languages are obtained.

A natural extension of these generative mechanisms is to consider also *erasing operations*. For contextual grammars this has been done in [25]. Insertion-deletion systems, corresponding to Galiukschov grammars with deletion, were investigated in [15].

Because the families of languages generated by types of grammars with insertion rules only seem to be "small" in comparison with families in the Chomsky hierarchy (often, they do not cover the family of regular or of linear languages), it is rather surprising to find characterizations of recursively enumerable languages starting from "insertion languages" and using certain operations with languages. For instance, in [8] it is shown that each RE language can be written as the quotient by a regular language of a language generated by a contextual grammar with finitely many selection strings associated with each context (this is the most restricted class of contextual grammars). A similar result is obtained in [19] for Galiukschov grammars.

The "explanation" of such results is, roughly speaking, the following one: in general, the ability of sensing contexts, supplemented by erasing possibilities can do everything that a type-0 grammar (a Turing machine) can do. In the results mentioned above, the context sensing is ensured by the context-dependent insertion, whereas the erasing is provided by the quotient operation.

However, as we have pointed out above, the erasing feature can be introduced in the generative mechanism, aiming in this way at obtaining direct characterizations of RE languages. This has been done in [15], where a way to simulate Turing machines by insertion-deletion systems is provided. Another proof of this characterization of RE is given in [19].

In the above mentioned papers no care has been paid to the length of the contexts used by the insertion-deletion operation, or to the length or to the shape of the inserted-deleted strings. We address here this problem, proving that quite restricted insertion-deletion grammars (we call them *systems*) are still able to characterize RE.

Besides being fundamental in formal language theory, the operations of insertion and deletion have recently become of interest in connection with the topic of molecular computing. The area of molecular computing was born in 1994 when Adleman, [1], succeeded to solve an instance of the Directed Hamiltonian Path Problem solely by manipulating DNA strands. This marked the first instance where a mathematical problem could be solved by biological means and, besides further research [16], gave rise to a couple of interesting problems: a) can *any*

algorithm be simulated by means of DNA manipulation, and b) is it possible, at least in theory, to design a programmable molecular computer? To answer these questions, various models of molecular computation have been proposed, and for some of these models it has been shown that the bio-operations involved can simulate the actions of a Turing machine (see, for example, [2], [4], [12], [3], [9], [27], [33], [23]).

Besides their being theoretically interesting, one of the motivations for studying insertions and deletions is that these operations can be used as the sole primitives needed for modeling DNA computation, [15], and moreover, they are already implementable in the laboratory. Indeed, by using available reagents and a standard technique called *PCR site-specific oligonucleotide mutagenesis* [7] one can perform insertions and deletions of nucleotide sequences. (A similar operation, substitution, has been proposed in [3] as a bio-operation necessary to simulate a universal Turing machine.)

The results obtained here address a related problem: how to obtain full computational power by insertions and deletions of a single nucleotide in an RNA sequence. Indeed, it is known that the U nucleotide can be relatively easily inserted and deleted in/from RNA sequences, [5]. This has led to the open problem whether or not each Turing machine can be simulated, in a convenient encoding, just by inserting/deleting strings consisting of one symbol only, U (see [30]). We prove here that the answer is affirmative. The proof that insertions and deletions of a single nucleotide are enough to simulate the actions of a Turing machine opens thus another possible way for designing a molecular computer.

2 Insertion-deletion systems

In this section we introduce the basic model investigated in this paper, the *insdel systems*. We denote by V^* the free monoid generated by an alphabet V under the operation of concatenation; the empty string is denoted by λ and the length of $x \in V^*$ is denoted by $|x|$. By *FIN, REG, LIN, CF, CS, RE* we denote the families of finite, regular, linear, context-free, context-sensitive, and of recursively enumerable languages, respectively. For basic elements of formal language theory, we refer to [28], [29].

An *insertion-deletion* (shortly, *insdel*) *system*, [15], is a construct

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

where V is an alphabet, $T \subseteq V$, A is a finite subset of V^*, and I, D are finite subsets of $V^* \times V^* \times V^*$.

The alphabet T is the terminal alphabet of γ, A is the set of axioms, I is the set of insertion rules, and D is the set of deletion rules. An insertion/deletion rule is given in the form (u, z, v); in order to increase the readability, we sometimes write $(u, z, v)_{ins}$ in order to indicate that (u, z, v) is an insertion rule and $(u, z, v)_{del}$ for $(u, z, v) \in D$. In an insertion/deletion rule (u, z, v), u, v represent the context of insertion/deletion (i.e. the insertion/deletion will take place between u and v), whereas z represents the string to be inserted/deleted.

For $x, y \in V^*$ we write $x \Longrightarrow y$ iff one of the following two cases holds:

(i). $x = x_1uvx_2$, $y = x_1uzvx_2$, for some $x_1, x_2 \in V^*$ and $(u, z, v) \in I$ (an insertion step);

(ii). $x = x_1uzvx_2$, $y = x_1uvx_2$, for some $x_1, x_2 \in V^*$ and $(u, z, v) \in D$ (a deletion step).

Denoting by $\Longrightarrow^*$ the reflexive and transitive closure of the relation $\Longrightarrow$, the language generated by γ is defined by

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

An insdel system $\gamma = (V, T, A, I, D)$ is said to be of *weight* (n, m, p, q) if

$$\max\{|z| \mid (u, z, v) \in I\} = n,$$
$$\max\{|u| \mid (u, z, v) \in I \text{ or } (v, z, u) \in I\} = m,$$
$$\max\{|z| \mid (u, z, v) \in D\} = p,$$
$$\max\{|u| \mid (u, z, v) \in D \text{ or } (v, z, u) \in D\} = q.$$

Intuitively, n is an upper bound for the length of the inserted string, while m is an upper bound for the length of the left and right context. The constants p and q represent the analogous upper bounds for deletion.

We denote by $INS_n^m DEL_p^q$, $n, m, p, q \geq 0$, the family of languages $L(\gamma)$ generated by insdel systems of weight (n', m', p', q') such that $n' \leq n$, $m' \leq m$, $p' \leq p$, $q' \leq q$. Because insertion/deletion of empty strings changes nothing, we ignore such rules; namely, when $n = 0$ we also assume that $m = 0$, and when $p = 0$ we also assume that $q = 0$. 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. When one of the parameters n, m, p, q is not bounded, we replace it by ∞. Thus, the family of all insdel languages is $INS_\infty^\infty DEL_\infty^\infty$.

3 Preliminary results

The families $INS_\infty^m DEL_0^0$ are those generated by Galiukschov grammars. Proofs of the following results can be found in [10], [20], [21], [31]:

(i). $FIN \subset INS_\infty^0 DEL_0^0 \subset INS_\infty^1 DEL_0^0 \subset INS_\infty^2 DEL_0^0 \subset \ldots$
$\subset INS_\infty^\infty DEL_0^0 \subset CS$.

(ii). REG is incomparable with all families $INS_\infty^m DEL_0^0, m \geq 0$, and $REG \subset INS_\infty^\infty DEL_0^0$.

(iii). $INS_\infty^1 DEL_0^0 \subset CF$, but CF is incomparable with all families $INS_\infty^m DEL_0^0, m \geq 2$, and with $INS_\infty^\infty DEL_0^0$; $INS_\infty^2 DEL_0^0$ contains non-semilinear languages.

(iv). LIN is incomparable with all families $INS_\infty^m DEL_0^0, m \geq 0$, and with $INS_\infty^\infty DEL_0^0$.

(v). All families $INS_\infty^m DEL_0^0, m \geq 0$, are anti-AFL's (that is, they are closed under none of the following operations: union, concatenation, Kleene closure, direct and inverse morphisms, intersection with regular languages).

(vi). Each regular language is the morphic image of a language in the family $INS_\infty^1 DEL_0^0$.

Moreover, in [19] it is proved that

(vii). Each language $L \in RE$ can be written in the form $L = g(h^{-1}(L'))$, for a morphism h, a weak coding g, and $L \in INS_\infty^7 DEL_0^0$. (In fact, from the proof in [19], we can see that $L' \in IND_4^7 DEL_0^0$.) As a corollary, we get the fact that L can be also written as $L = R \backslash L'$, for $R \in REG$ and $L' \in INS_\infty^7 DEL_0^0$ (in fact, from the proof we get $L' \in INS_5^7 DEL_0^0$).

From the definitions, we obviously have

Theorem 1. *$INS_n^m DEL_p^q \subseteq INS_{n'}^{m'} DEL_{p'}^{q'}$ for all $0 \leq n \leq n'$, $0 \leq m \leq m'$, $0 \leq p \leq p'$, $0 \leq q \leq q'$.*

A proof of the following result can be found in [15].

Theorem 2. *$RE = INS_\infty^\infty DEL_\infty^\infty$.*

In fact, from the proof in [15] we have $RE = INS_3^6 DEL_2^7$. Another proof of the result in Theorem 2 is given in [19]. In order to see the power of insertion/deletion operations, we recall here the construction in [19], making also explicit the weight of the obtained insdel system.

Theorem 3. *$RE = INS_3^2 DEL_3^0$.*

Proof. Take a language $L \subseteq T^*$ generated by a grammar $G = (N, T, S, P)$ in Kuroda normal form, hence with P containing context-free rules of the form $X \to x, |x| \leq 2$, and non-context-free rules of the form $XY \to UZ$, for $X, Y, U, Z \in N$. We construct the insdel system

$$\begin{aligned}
\gamma &= (N \cup T \cup \{E, K_1, K_2\}, T, \{SEE\}, I, D), \\
I &= \{(X, K_1 x, \alpha_1 \alpha_2) \mid X \to x \in P, \alpha_1, \alpha_2 \in N \cup T \cup \{E\}\} \\
&\cup \{(XY, K_2 UZ, \alpha_1 \alpha_2) \mid XY \to UZ \in P, \alpha_1, \alpha_2 \in N \cup T \cup \{E\}\} \\
D &= \{(\lambda, XK_1, \lambda) \mid X \in N\} \\
&\cup \{(\lambda, XYK_2, \lambda) \mid X, Y \in N\} \\
&\cup \{(\lambda, EE, \lambda)\}.
\end{aligned}$$

The symbol E is a dummy symbol used when checking the context $\alpha_1 \alpha_2$ at the end of the string. The symbols K_1, K_2 are "killers": K_1 removes one symbol, namely the one placed immediately to its left, and K_2 removes two symbols, those placed immediately to its left. Making use of these symbols, the rules in I simulate the rules in P. Symbols already marked by the "killers" K_1, K_2 cannot be used as contexts of rules in I. Consequently we get $L(G) = L(\gamma)$. □

4 Restricting the weight

A natural problem in this framework is to look for characterization results $RE = INS_n^m DEL_p^q$ with as small as possible values for n, m, p, q. We give here three

results proving the surprising power of "small" insdel systems: systems of weight (1, 2, 1, 1), or (2, 1, 2, 0), or (1, 2, 2, 0) are able to simulate any type-0 grammar.

These results do not settle the above formulated problem. For instance, we do not know the power of insdel systems of weight (n, m, p, q) strictly smaller than (1, 2, 1, 1) and than (2, 1, 2, 0) and than (1, 2, 2, 0), or of weight (n, m, p, q) incomparable to these quadruples. Particularly interesting are the systems of weights (1, 1, 1, 1), (1, 1, 2, 0), (2, 1, 1, 1), and (1, 2, 1, 0). Are such systems generating non-context-free languages ? We *conjecture* that the answer is negative.

Theorem 1. $RE = INS_1^2 DEL_1^1$.

Proof. According to the Church-Turing thesis, we only have to prove the inclusion $\subseteq$.

Consider a language $L \subseteq T^*, L \in RE$, generated by a grammar $G = (N, T, S, P)$ in the Penttonen normal form, [26], that is containing context-free rules $X \to x$ with $|x| \leq 2$, and non-context-free rules of the form $XY \to XZ$, for $X, Y, Z \in N$.

Without loss of generality we may assume that in each rule $X \to \alpha_1\alpha_2 \in P$ we have $X \neq \alpha_1$, $X \neq \alpha_2$, $\alpha_1 \neq \alpha_2$. (If necessary, we replace $X \to \alpha_1\alpha_2$ with $X \to X'$, $X' \to \alpha_1\alpha_2'$, $\alpha_2' \to \alpha_2$, where X', α_2' are new symbols.) Similarly, we may assume that for each rule $XY \to XZ \in P$ we have $X \neq Y$, $X \neq Z$, $Y \neq Z$. Moreover, by replacing each rule $X \to \alpha \in P$, $\alpha \in N \cup T$, by $X \to \alpha Z, Z \to \lambda$, we obtain an equivalent grammar. Hence, we may assume that the rules in P are of the following three forms:

1. $X \to \alpha_1\alpha_2,\ \alpha_1, \alpha_2 \in N \cup T, X \neq \alpha_1, X \neq \alpha_2, \alpha_1 \neq \alpha_2$
2. $X \to \lambda$,
3. $XY \to XZ,\ X, Y, Z \in N, X \neq Y, X \neq Z, Y \neq Z$.

Moreover, we assume the rules of P labelled in an one-to-one manner

We construct the insdel system

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

where

$$\begin{aligned} V &= N \cup T \cup \{[r], (r \quad r \text{ is the label of a rule in } P\} \\ &\cup \{B, E\}, \\ A &= \{BSE\}, \end{aligned}$$

and the sets I, D are constructed as follows.

(i). For each rule $r : X \to \alpha_1\alpha_2 \in P, \alpha_1, \alpha_2 \in N \cup T$, of type 1, we consider the following insertion/deletion rules:

$$\begin{aligned} &(r.1.) \quad (\beta_1, [r], X\beta_2)_{ins}, \beta_1 \in N \cup T \cup \{B\}, \beta_2 \in N \cup T \cup \{E\} \\ &(r.2.) \quad ([r]X, (r), \beta)_{ins}, \beta \in N \cup T \cup \{E\}, \\ &(r.3.) \quad ([r], X, (r))_{del}. \end{aligned}$$

$(r.4.)$ $([r], \alpha_1, (r))_{ins}$
$(r.5.)$ $(\alpha_1, \alpha_2, (r))_{ins}$
$(r.6.)$ $(\lambda, [r], \alpha_1)_{del}$,
$(r.7.)$ $(\alpha_2, (r), \lambda)_{del}$.

(ii). For each rule $r \quad X \to \lambda \in P$ of type 2, we introduce the deletion rule

$$(r.1.) \quad (\beta_1, X, \beta_2)_{del}, \beta_1 \in N \cup T \cup \{B\}, \beta_2 \in N \cup T \cup \{E\}$$

(iii). For each rule $r : XY \to XZ \in P$, $X, Y, Z \in N$, of type 3, we consider the following insertion/deletion rules:

$(r.1.)$ $(\beta_1 X, [r], Y\beta_2)_{ins}, \beta_1 \in N \cup T \cup \{B\}, \beta_2 \in N \cup T \cup \{E\}$
$(r.2.)$ $([r]Y, (r), \beta)_{ins}, \beta \in N \cup T \cup \{E\}$,
$(r.3.)$ $([r], Y, (r))_{del}$,
$(r.4.)$ $([r], Z, (r))_{ins}$,
$(r.5.)$ $(X, [r], Z)_{del}$,
$(r.6.)$ $(Z, (r), \lambda)_{del}$.

(iv). We also consider the deletion rules

$$(\lambda, B, \lambda)$$
$$(\lambda, E, \lambda)$$

We claim that the equality $L(G) = L(\gamma)$ holds.

$(\subseteq)$ Each derivation step $w \Longrightarrow w'$ in G is simulated in γ by a derivation $BwE \Longrightarrow^* Bw'E$, using the rules $(r.1.) - (r.i.)$ associated as above to the rule in P used in $w \Longrightarrow w'$. For instance, assume that $w = w_1 X w_2$, $w' = w_1\alpha_1\alpha_2 w_2$, for $r : X \to \alpha_1\alpha_2 \in P$. Then we successively obtain:

$$\begin{aligned}
Bw_1Xw_2E &\Longrightarrow Bw_1[r]Xw_2E && \text{by the rule } (r.1.)\\
&\Longrightarrow Bw_1[r]X(r)w_2E && \text{by the rule } (r.2.)\\
&\Longrightarrow Bw_1[r](r)w_2E && \text{by the rule } (r.3.)\\
&\Longrightarrow Bw_1[r]\alpha_1(r)w_2E && \text{by the rule } (r.4.)\\
&\Longrightarrow Bw_1[r]\alpha_1\alpha_2(r)w_2E && \text{by the rule } (r.5.)\\
&\Longrightarrow Bw_1\alpha_1\alpha_2(r)w_2E && \text{by the rule } (r.6.)\\
&\Longrightarrow Bw_1\alpha_1\alpha_2w_2E && \text{by the rule } (r.7.)\\
&= Bw'E.
\end{aligned}$$

We proceed in a similar way when $w \Longrightarrow w'$ is obtained by using a rule $r : XY \to XZ$. The details are left to the reader.

A derivation starts from BSE; at any moment, the markers B, E can be removed. Thus, any terminal string generated by G is in $L(\gamma)$.

$(\supseteq)$ Consider a string BwE; initially we have $w = S$. We can apply to it a rule $(r.1.)$ from group 1, or a deletion rule (β_1, X, β_2), or a rule $(r.1.)$ from

group 3. Assume that we apply $(\beta_1, [r], X\beta_2)_{ins}$ for some $r \quad X \to \alpha_1\alpha_2 \in P$ We have

$$Bw_1Xw_2E \Longrightarrow Bw_1[r]Xw_2E.$$

Because the rules in P are labelled in an one-to-one way, because $X \neq \alpha_1$, and because rules of the form of $(r.1.)$ in groups 1 and 3 have a left context checking the symbol placed immediately to the left of X (the same for the deletion rules in group 2), the only rule which can use the symbol X is $(r.2.)$. Eventually this rule must be applied, otherwise the derivation cannot lead to a terminal string. Thus, the substring $[r]X$ of $Bw_1[r]Xw_2$ leads to $[r]X(r)$. Again there is only one possible continuation, by the rule $(r.3.)$, which erases the symbol X. Only after inserting α_1 in-between $[r]$ and (r) we can remove the symbol $[r]$. In the presence of α_1 and of (r) we can introduce α_2, too, by the rule $(r.5.)$. Because (r) is introduced after $[r]$, and $X \neq \alpha_1$, the symbol α_1 used by this rule $(r.5.)$ as a left context should be introduced at a previous step, by the corresponding rule $(r.4.)$. After introducing α_2, which is different from both α_1 and X, we can delete (r), by the rule $(r.7.)$. Due to the contexts, no other rule can use the mentioned symbols as contexts or can delete any of them. Thus, after using $(r.1.)$, we have to use all rules $(r.i.), 2 \leq i \leq 7$, associated with $r : X \to \alpha_1\alpha_2$, simulating the use of $X \to \alpha_1\alpha_2$.

In the same way, after using a rule $(\beta_1 X, [r], Y\beta_2)$ associated with $r : XY \to XZ \in P$, we have to continue with $(r.i.), 2 \leq i \leq 6$ (possibly not immediately or at consecutive steps, but using the same symbols of the current string), hence we have to simulate the rule $XY \to XZ$.

The deletion rules (β_1, X, β_2) directly correspond to erasing rules in P. The markers B, E can be deleted at any step. Consequently, γ can generate only strings in $L(G)$. □

Theorem 2. $RE = INS_2^1 DEL_2^0$.

Proof. For every $L \subseteq T^*, L \in RE$, there is a grammar $G = (N, T, S, P)$ such that $L = L(G)$ and P contains two types of rules: context-free rules and rules of the form $XY \to \lambda, X, Y \in N$ (see [11], [22], [32]). Without loss of generality, we may assume that the context-free rules in P are of the form $X \to u$ with $|u| \leq 2$. Moreover, from the proofs in [11], [22], [32] we see that the symbols X, Y in rules $XY \to \lambda$ do not appear in the left hand members of the context-free rules in P.

Consider such a grammar G, where the rules in P of the form $X \to \alpha_1\alpha_2$ have been labelled in an one-to-one manner, and construct the insdel system $\gamma = (V, T, A, I, D)$ with

$$\begin{aligned}
V &= N \cup T \cup \{F_r, F_r' \mid r : X \to \alpha_1\alpha_2 \in P, \alpha_1, \alpha_2 \in N \cup T\} \\
&\cup \{B, K\}, \\
A &= \{BS\}, \\
I &= \{(\beta, \alpha_1 F_r, X),\ (\beta, \alpha_2 K, F_r),\ (K, F_r', F_r) \mid \\
&\qquad r : X \to \alpha_1\alpha_2 \in P, \alpha_1, \alpha_2 \in N \cup T, \beta \in N \cup T \cup \{B\}\} \\
&\cup \{(\beta, \alpha K, X) \mid X \to \alpha \in P, \alpha \in N \cup T, \beta \in N \cup T \cup \{B\}\}, \\
D &= \{(\lambda, F_r' F_r, \lambda) \mid r : X \to \alpha_1\alpha_2, \alpha_1, \alpha_2 \in N \cup T\} \\
&\quad \{(\lambda, KX, \lambda) \mid X \to u \in P\}
\end{aligned}$$

$$\cup \quad \{(\lambda, XY, \lambda) \mid XY \to \lambda \in P\}$$
$$\cup \quad \{(\lambda, B, \lambda)\}.$$

We obtain the equality $L(G) = L(\gamma)$.

($\subseteq$) Consider a derivation step in G, $w \Longrightarrow w'$. If the used rule is of the form $XY \to \lambda$, then clearly $Bw \Longrightarrow Bw'$ in γ, by using the rule $(\lambda, XY, \lambda) \in D$. If the used rule is $X \to \alpha, \alpha \in N \cup T$, then

$$\begin{aligned} Bw = Bw_1Xw_2 &\Longrightarrow Bw_1\alpha KXw_2 && \text{by the rule } (\beta, \alpha K, X)_{ins} \\ &\Longrightarrow Bw_1\alpha w_2 && \text{by the rule } (\lambda, KX, \lambda)_{del} \\ &= Bw'. \end{aligned}$$

If the used rule is $r : X \to \alpha_1\alpha_2$, then we proceed as follows

$$\begin{aligned} Bw = Bw_1Xw_2 &\Longrightarrow Bw_1\alpha_1F_rXw_2 && \text{by the rule } (\beta, \alpha_1F_r, X)_{ins} \\ &\Longrightarrow Bw_1\alpha_1\alpha_2KF_rXw_2 && \text{by the rule } (\beta, \alpha_2K, F_r)_{ins} \\ &\Longrightarrow Bw_1\alpha_1\alpha_2KF_r'F_rXw_2 && \text{by the rule } (K, F_r', F_r)_{ins} \\ &\Longrightarrow Bw_1\alpha_1\alpha_2KXw_2 && \text{by the rule } (\lambda, F_r'F_r, \lambda)_{del} \\ &\Longrightarrow Bw_1\alpha_1\alpha_2w_2 && \text{by the rule } (\lambda, KX, \lambda)_{del} \\ &= Bw'. \end{aligned}$$

Consequently, each derivation $S \Longrightarrow^* x$ in G can be reproduced in γ as $BS \Longrightarrow^* Bx$; then B can be removed by the rule $(\lambda, B, \lambda) \in D$. If $x \in T^*$, then $x \in L(\gamma)$.

($\supseteq$) Consider a string of the form Bw with $w \in (N \cup T)^*$. (Initially we have $w = S$.) If a rule $(\lambda, XY, \lambda) \in D$ can be applied to w, then it corresponds to the use of the rule $XY \to \lambda$ for rewriting w. No other deletion rule can be used.

If we use an insertion rule $(\beta, \alpha K, X)$, then $w = w_1Xw_2$ (w_1 can be λ, and then $\beta = B$). We get the string $Bw_1\alpha KXw_2$. The symbol X in the right hand of K cannot be used by any other rule of γ excepting $(\lambda, KX, \lambda) \in D$ (the symbol K marks the symbols which cannot be used in further insertion rules). Eventually, this deletion rule should be used, hence we simulate in this way the rule $X \to \alpha \in P$.

Another possibility is to use the insertion rule (β, α_1F_r, X), for some $r : X \to \alpha_1\alpha_2 \in P$. We get a string of the form $Bw_1\alpha_1F_rXw_2$. No rule in γ can use the symbols F_r and X, excepting $(\beta, \alpha_2K, F_r) \in I$. Eventually, this rule is used, hence α_1F_rX is replaced by $\alpha_1\alpha_2KF_rX$. The only way to remove the nonterminals K, F_r, X is by first introducing F_r', by the rule $(K, F_r', F_r) \in I$ (hence obtaining $\alpha_1\alpha_2KF_r'F_rX$), then erasing $F_r'F_r$ and KX, by using the rules $(\lambda, F_r'F_r, \lambda)_{del}$ and $(\lambda, KX, \lambda)_{del}$, respectively. This corresponds to the use of the rule $X \to \alpha_1\alpha_2$.

The symbol B ensures the fact that we can always check the left context β of insertion rules.

Therefore, $L(\gamma) = L(G)$. □

Combining the ideas of the previous proofs, we also get

Theorem 3. $RE = INS_1^2DEL_2^0$.

Proof. Start as in the proof of Theorem 2 from a grammar $G = (N, T, S, P)$ with P containing context-free rules of the form $X \to u$, with $|u| = 2$ or $u = \lambda$, and non-context-free rules of the form $XY \to \lambda$. Without loss of generality we may assume that if $X \to \alpha_1\alpha_2 \in P$, then $\alpha_1 \neq X$, $\alpha_2 \neq X$, and that X, Y in rules of the form $XY \to \lambda$ do not appear as left hand members of context-free rules.

Then we construct the insdel system $\gamma = (V, T, A, I, D)$, with

$$\begin{aligned} V &= N \cup T \cup \{[r], (r) \mid r \quad X \to \alpha_1\alpha_2 \in P\} \\ &\cup \{B, E, K\}, \\ A &= \{BSE\}, \end{aligned}$$

and the sets I, D are constructed as follows

(i). For each rule $r : X \to \alpha_1\alpha_2 \in P$ we consider the following rules:

$$\begin{aligned} &(\beta_1, [r], X\beta_2)_{ins}, \ \beta_1 \in N \cup T \cup \{B\}, \beta_2 \in N \cup T \cup \{E\}, \\ &([r]X, (r), \beta)_{ins}, \ \beta \in N \cup T \cup \{E\}, \\ &([r]X, K, (r))_{ins}, \\ &(\lambda, XK, \lambda)_{del}, \\ &([r], \alpha_1, (r))_{ins}, \\ &([r]\alpha_1, \alpha_2, (r))_{ins}, \\ &([r], K, \alpha_1\alpha_2)_{ins}, \\ &(\lambda, [r]K, \lambda)_{del}, \\ &(\alpha_1\alpha_2, K, (r))_{ins}, \\ &(\lambda, K(r), \lambda)_{del}. \end{aligned}$$

(They simulate the rule r in the same way as the rules $(r.1.) - (r.7.)$ in the proof of Theorem 1, with the difference that the control symbols $[r], (r)$ are deleted in the presence of the "killer" K, which is introduced only after replacing X by $\alpha_1\alpha_2$.)

(ii). For each rule $r : X \to \lambda \in P$, we introduce the rules

$$\begin{aligned} &(X, K, \beta)_{ins}, \ \beta \in N \cup T \cup \{E\}, \\ &(\lambda, XK, \lambda)_{del}. \end{aligned}$$

(We cannot introduce two or more occurrences of K for the same X, due to the presence of the symbol β.)

(iii). For each rule $r : XY \to \lambda \in P$ we introduce the rule

$$(\lambda, XY, \lambda)_{del}$$

As in the previous proofs, one can see that $L(G) = L(\gamma)$ □