Insertion operations: closure properties

Lila Kari
Academy of Finland and Mathematics Department[1]
Turku University
20 500 Turku, Finland

1 Introduction

The basic notions used for specifying languages are of *algorithmic* or *operational* character: automata (*accepting* devices) or grammars (*generating* devices). This duality reflects the original motivations coming from computer science or linguistics. A deeper theory and more involved proofs called for alternative definitional devices, where the operations have a classical mathematical character. Early examples of such definitional devices are the sequential functions for automata (see [14], pp.48-53) or substitutions and morphisms for grammars.

This paper goes into the fundamentals of the substitution operation, aiming thus to a better understanding of the mechanisms of generating languages. So far in the literature a substitution has been defined as an operation on an alphabet. A substitution is never applied to λ (except for the convention that λ is always mapped into λ). Our work can be viewed as an attempt to understand the substitution on the empty word.

Let L_1, L_2 be two languages over an alphabet Σ. The operation of *sequential insertion* of a language L_2 into a language L_1, can be viewed as a nonstandard modification of the notion of substitution. It maps all letters of Σ into themselves and the empty letter into L_2, with the following additional convention. For each word w, only one of the empty words occurring in it is substituted. The result of applying this "substitution" to L_1 consists of words obtained from words in L_1 in which an arbitrary word from L_2 has been inserted. Note that the sequential insertion is also a generalization of the catenation operation. The sequential insertion operation can be viewed also as a one-step rewriting rule of a Thue system (see [4], [3]). Consequently, some closure properties of the families in the Chomsky hierarchy under iterated sequential insertion can be obtained using some results about Thue systems.

[1]The work reported here is part of the project 11281 of the Academy of Finland

The operation of *parallel insertion* is closer to the classical notion of substitution. It is defined as the substitution above, but with a modified convention: For each word w, between all the letters and also at the extremities, only one λ occurs. The effect of the substitution applied to L_1 will be the insertion of words belonging to L_2 between all letters and also at the extremities of words belonging to L_1.

The exact effect of the classical substitution that maps all letters into themselves and λ into a language L_2 would be the insertion of arbitrary many words of L_2 between letters and at the extremities of words in L_1. According to the definitions mentioned above, this would amount to the parallel insertion of L_2^* into L_1.

Another way to look at operations of controlled insertion (each letter determines what may be inserted after it) is the following. Such an insertion can be viewed as a *production* of the form $a \longrightarrow aw$, where the word w comes from the language to be inserted next to a. The mode of controlled insertion determines how the productions are going to be applied. The controlled *parallel* insertion resembles, thus, the rewriting process of OL systems (see [13]). However, it gives rise to something different from OL systems because the productions are always of the special form and, on the other hand, there may be infinitely many productions. The relation between controlled *sequential* insertion and OS systems (see, for instance, [11]) is similar.

Various insertion operations and the dual deletion counterparts, together with their properties, language equations involving them and connections to cryptography, have been studied in [6], [9], [10]. In this paper we restrict ourselves to the study of algebraic and closure properties of some insertion operations.

In the following, REG, CF, CS will denote the families of regular, context-free, context-sensitive languages, respectively. For other formal language notions and notations we refer to [15].

2 Sequential and parallel insertion

The most natural generalization of catenation is the *sequential insertion* (abbreviated SIN) operation. Given two words u and v, instead of catenating v at the right extremity of u, the new operation inserts it in an arbitrary place in u:

$$u \Leftarrow v = \{u_1 v u_2 |\ u = u_1 u_2,\ u_1, u_2 \in \Sigma^*\}.$$

The result of the sequential insertion will be thus a set of words instead of a single word. The cardinality of the set is at most $n+1$, where n is the length of u. The catenation uv is an element of this set, corresponding to the particular case where the insertion is done at the right extremity of u.

For example, $cd \Leftarrow a = \{acd, cad, cda\}$, where a, c, d are letters in Σ.

The sequential insertion operation is not associative. For example, $(a \leftarrow b) \leftarrow c \neq a \leftarrow (b \leftarrow c)$. In general the following relation holds:

$$L_1 \leftarrow (L_2 \leftarrow L_3) \subseteq (L_1 \leftarrow L_2) \leftarrow L_3,$$

for any languages L_1, L_2, L_3 over an alphabet Σ.

The sequential insertion is nor commutative. For example, $a \leftarrow bc \neq bc \leftarrow a$.

Theorem 1 *Any family of languages closed under λ-free gsm mappings, λ-free substitutions and union is closed under sequential insertion.*

Proof. Let Σ be an alphabet and let $\#$ be a letter which does not occur in Σ. If we consider the λ-free gsm:

$$\begin{array}{ll} g = & (\Sigma, \Sigma \cup \{\#\}, \{s_0, s\}, s_0, \{s\}, P), \\ P = & \{s_0 a \longrightarrow a s_0 |\ a \in \Sigma\} \cup \{s_0 a \longrightarrow a \# s |\ a \in \Sigma\} \cup \\ & \{sa \longrightarrow as |\ a \in \Sigma\} \cup \{s_0 a \longrightarrow \# as |\ a \in \Sigma\} \end{array}$$

then we obviously have $g(L) = L \leftarrow \{\#\}$, for any language $L \subseteq \Sigma^+$.

If for two languages L_1, L_2 we define now the λ-free substitution

$$s' : (\Sigma \cup \{\#\})^* \longrightarrow 2^{\Sigma^*},\ s'(\#) = L_2 - \{\lambda\}, s'(a) = a, \forall a \in \Sigma,$$

then the following relations hold:

$$\begin{array}{lll} L_1 \leftarrow L_2 = & s'(g(L_1)) \cup L_1 \cup L_2, & \text{if } \lambda \in L_1 \cap L_2, \\ L_1 \leftarrow L_2 = & s'(g(L_1)) \cup L_1, & \text{if } \lambda \in L_2 - L_1, \\ L_1 \leftarrow L_2 = & s'(g(L_1)) \cup L_2, & \text{if } \lambda \in L_1 - L_2, \\ L_1 \leftarrow L_2 = & s'(g(L_1)), & \text{if } \lambda \notin L_1 \cup L_2. \end{array}$$

□

A parallel variant of the sequential insertion will be defined in the following. The *parallel insertion* (shortly PIN) of a word v into a word u is the word obtained after inserting v between all the letters of u and at the right and left extremities of u. The definition can be easily transferred to languages.

$$L_1 \Leftarrow L_2 = \bigcup_{u \in L_1} (u \Leftarrow L_2), \text{where}$$

$$\begin{array}{ll} u \Leftarrow L_2 = & \{v_0 a_1 v_1 a_2 v_2 \ldots a_k v_k |\ k \geq 0, a_j \in \Sigma, 1 \leq j \leq k, \\ & v_i \in L_2, 0 \leq i \leq k, \text{ and } u = a_1 a_2 \ldots a_k\}. \end{array}$$

The case $k = 0$ corresponds to the situation $u = \lambda$ when only one word $v_0 \in L_2$ is inserted.

The parallel insertion operation induces a monoid structure on $\mathcal{P}(\Sigma^*)$. Indeed, the operation is associative and the neutral element of the monoid is $\{\lambda\}$. The monoid $(\mathcal{P}(\Sigma^*), \Leftarrow)$ is not commutative. For example, $a \Leftarrow b = \{bab\}$, whereas $b \Leftarrow a = \{aba\}$.

Theorem 2 *Any family of languages closed under catenation and λ-free substitution is closed under parallel insertion.*

Proof. The theorem follows as for any two languages over Σ we have $L_1 \Leftarrow L_2 = L_2 s(L_1)$ where s is the λ-free substitution defined by:

$$s : \Sigma^* \longrightarrow 2^{\Sigma^*}, s(a) = aL_2, \quad \text{for every } a \in \Sigma.$$

□

3 Iterated insertion

In the same way as the sequential and parallel insertion are generalizations of the catenation operation, the *iterated sequential* and *iterated parallel insertion* are generalizations of the catenation closure. However, the iterated SIN and PIN prove to be more powerful than the iterated catenation. Indeed, the family of regular (context-free) languages is closed under catenation closure, whereas it is not closed under iterated SIN (iterated PIN) of a language into itself.

Definition 1 *Let L_1, L_2 be languages over the alphabet Σ. The insertion of order n of L_2 into L_1 is inductively defined by the equations:*

$$L_1 \leftarrow {}^0 L_2 = L_1, \quad L_1 \leftarrow {}^{i+1} L_2 = (L_1 \leftarrow {}^i L_2) \leftarrow L_2, i \geq 0.$$

The iterated sequential insertion (iterated SIN) of L_2 into L_1 is then defined as:

$$L_1 \leftarrow\!\!\!\!\leftarrow^* L_2 = \bigcup_{n=0}^{\infty} (L_1 \leftarrow {}^n L_2).$$

The iterated SIN is not a commutative operation. For example, $\lambda \leftarrow\!\!\!\!\leftarrow^* b = b^*$, whereas $b \leftarrow\!\!\!\!\leftarrow^* \lambda = b$. The operation is not associative either. In general, for L_1, L_2, L_3 arbitrary languages, the sets $L_1 \leftarrow\!\!\!\!\leftarrow^* (L_2 \leftarrow\!\!\!\!\leftarrow^* L_3)$ and $(L_1 \leftarrow\!\!\!\!\leftarrow^* L_2) \leftarrow\!\!\!\!\leftarrow^* L_3$ are incomparable.

Proposition 1 *There exist a finite language L_1 and a word w such that $L_1 \leftarrow\!\!\!\!\leftarrow^* w$ is not a regular language.*

Proof. The iterated SIN of $\{()\}$ into $\{\lambda, ()\}$ is the Dyck language of order one, which is a non-regular context-free language. □

The following lemma shows that adding the empty word to a λ-free language to be inserted does not change the result of the iterated SIN. The result can be proved by induction on the number of iterations.

Lemma 1 *If L_1, L_2 are languages over an alphabet Σ then:*

$$L_1 \leftarrow\!\!\!\!\leftarrow^* L_2 = L_1 \leftarrow\!\!\!\!\leftarrow^* (L_2 - \{\lambda\}).$$

The fact that the family of context-free languages is closed under iterated sequential insertion has been shown in [4], [3] in the context of Thue system theory and can be also directly obtained by using the closure of CF under nested iterated substitution (see [5]). For the closure of CS under iterated SIN, we give bellow the following constructive proof.

Theorem 3 *The family of context-sensitive languages is closed under iterated sequential insertion.*

Proof. Let $L_1 = L(G_1)$, $L_2 = L(G_2)$ be two languages generated by the context-sensitive grammars $G_i = (N_i, \Sigma_i, S_i, P_i)$, $i = 1, 2$. Assume that $N_1 \cap N_2 = \emptyset$. Assume further that G_i , $i = 1, 2$ satisfy the conditions (see, for example, [15], pp.19-20): (i)S_i does not occur on the right side of any production of P_i ; (ii) if $\lambda \in L(G_i)$ then the only rule which has the right side equal with λ is $S_i \longrightarrow \lambda$; (iii) every rule of P_i containing a terminal letter is of the form $A \longrightarrow a$ where $A \in N_i$ and $a \in \Sigma_i$.

According to Lemma 1 we can also assume that λ does not belong to L_2.

Let $\#$ be a new symbol which does not occur in any of the considered alphabets. Consider the context-sensitive grammar $G = (N_1 \cup N_2, \Sigma_1 \cup \Sigma_2 \cup \{\#\}, S_1, P)$ whose rules are:

$$\begin{aligned} P = \quad & P_1 \cup P_2 \cup \{S_1 \longrightarrow S_2 |\ S_1 \longrightarrow \lambda \in P_1\} \cup \\ & \{A \longrightarrow \#S_2\#a, A \longrightarrow a\#S_2\# |\ A \longrightarrow a \in P_1 \cup P_2\}. \end{aligned}$$

Define now the morphism $h : \Sigma^* \longrightarrow (\Sigma_1 \cup \Sigma_2)^*$ by $h(\#) = \lambda$, $h(a) = a$, $\forall a \in \Sigma_1 \cup \Sigma_2$. The role of the morphism h being obvious, it can be easily proved that $h(L(G)) = L_1 \leftarrow^* L_2$.

From the form of the rules of P we notice that the number of markers $\#$ in a word $u \in L(G)$, denoted $N_{\#}(u)$, depends on the number of terminals. More precisely, if by $\lg(w)$ we denote the length of a word w, we have $N_{\#}(u) \leq 2 \times \lg(h(u))$, for every word $u \in L(G)$. Consequently, $\lg(u) \leq 3 \times \lg(h(u))$, $\forall u \in L(G)$, that is, h is a 3-linear erasing with respect to $L(G)$.

The theorem now follows as the family of context-sensitive languages is closed under linear erasing.

□

The iterated parallel insertion of a language L_2 into a language L_1 is defined by replacing in the definition of iterated SIN "$\leftarrow$" with "$\Leftarrow$". The iterated PIN is neither a commutative nor an associative operation.

The next result shows that the iterated PIN is more powerful than the iterated SIN: starting only with two one-letter words, the iterated PIN can produce a non-context-free language. However the family of context-sensitive languages is still closed under iterated PIN.

Proposition 2 *There exists a singleton language L such that $L \Leftarrow^* L$ is not a context-free language.*

Proof. If b is a singleton letter, the result of iterated PIN of b into itself is $b \Leftarrow^* b = \{b^{2^k-1} |\ k > 0\}$. □

Theorem 4 *The family of context-sensitive languages is closed under iterated parallel insertion.*

Proof. Let L_1, L_2 be languages generated by the context-sensitive grammars $G_i = (N_i, \Sigma_i, S_i, P_i)$, $i = 1, 2$. Assume that the grammars satisfy the conditions stated in Theorem 3. Assume further that λ does not belong to L_2.

Let $G = (N, \Sigma, S, P)$ be the context-sensitive grammar whose components are:

$$\begin{aligned}
N &= N_1 \cup N_2 \cup \{S, X\},\\
\Sigma &= \Sigma_1 \cup \Sigma_2 \cup \{\$, \#\},\\
P &= (P_1 - \{S_1 \longrightarrow \lambda\}) \cup P_2 \cup\\
&\quad \{S \longrightarrow XS, S \longrightarrow \$S_1\#\} \cup\\
&\quad \{X\$ \longrightarrow \$S_2X\} \cup \{X\$\# \longrightarrow \$S_2\#|\ S_1 \longrightarrow \lambda \in P_1\} \cup\\
&\quad \{Xa \longrightarrow aS_2X|\ a \in \Sigma_1 \cup \Sigma_2\} \cup\\
&\quad \{Xa\# \longrightarrow aS_2\#|\ a \in \Sigma_1 \cup \Sigma_2\} \cup\\
&\quad \{S \longrightarrow \$\#|\ S_1 \longrightarrow \lambda \in P_1\},
\end{aligned}$$

where $S, X, \$, \#$, are new symbols which do not occur in any of the given vocabularies. Intuitively, the grammar works as follows. X^n represents the number of iterations, \$ and # are markers. After a sentential form of the type $X^n\$u\#$, $u \in L_1$ is reached, X starts to move to the right, producing an S_2 at the left extremity of u, and after every letter in u. When it reaches the right extermity of the sentential form, X dissapears. The introduced S_2's produce words of L_2. Thus, a word from $L_1 \Leftarrow^* L_2$ is obtained. After n iterations of the process, we obtain a word in $\$(L_1 \Leftarrow^n L_2)\#$.

Consequently, we have $\$(L_1 \Leftarrow^* L_2)\# = L(G)$.

If we consider now the morphism $h : \Sigma^* \longrightarrow \Sigma^*$ defined by $h(\$) = h(\#) = \lambda$ and $h(a) = a, \forall a \in \Sigma_1 \cup \Sigma_2$, it is easy to see that,

$$L_1 \Leftarrow^* L_2 = \begin{cases} h(L(G) - \{\$\#\}) \cup \{\lambda\}, & \text{if } \$\# \in L(G),\\ h(L(G)), & \text{if } \$\# \notin L(G). \end{cases}$$

It is obvious that h is a 3-linear erasing with respect to $L(G) - \{\$\#\}$. As the family of context-sensitive languages is closed under linear erasing and union, it follows that $L_1 \Leftarrow^* L_2$ is context-sensitive.

We have assumed until now that the language L_2 is λ-free. However, the theorem holds also if λ belongs to L_2 because in this case, $L_1 \Leftarrow^* L_2 = L_1 \leftarrow^* L_2$. □

4 Permuted and controlled insertion

A more exotic variant of insertion is obtained if we combine the ordinary insertion with the commutative variant. The commutative variant $\mathrm{com}(v)$ of a word

v is the set of all words obtained by arbitrarily permuting the letters of v. The *permuted insertion* of v into u will then consist of inserting into u all the words from the commutative variant of v:

$$u \leftsquigarrow v = u \leftarrow \mathrm{com}(v).$$

The permuted SIN is not a commutative operation. For example, $a \leftsquigarrow bc \neq bc \leftsquigarrow a$. The operation is not associative either. The permuted PIN is neither commutative nor associative.

It follows from the definition that any family of languages which is closed under SIN (respectively PIN) and commutative closure is closed under permuted SIN (respectively permuted PIN). REG and CF are not closed under these operations, as shown by the following result.

Proposition 3 *There exist two regular languages L_1, L_2 such that $L_1 \leftsquigarrow L_2$ and $L_1 \leftsquigarrow\!\!\!\!\!\!\sim\, L_2$ are not context-free.*

Proof. Let $L_1 = \{\lambda\}$ and $L_2 = (abc)^*$. The permuted sequential insertion of L_2 into L_1 is:

$$L_1 \leftsquigarrow L_2 = L_1 \leftsquigarrow\!\!\!\!\!\!\sim\, L_2 = L_1 \leftarrow (\mathrm{com}(L_2)) = \mathrm{com}(L_2) =$$

$$\{w \in \{a,b,c\}^* |\ N_a(w) = N_b(w) = N_c(w)\}$$

which is a non-context-free language. □

We have dealt so far with operations where the insertion took place in arbitrary places of a word. As a consequence, catenation is not a particular case of any of these operations, because one cannot fix the position where the insertion takes place. A natural idea of controlling the position where the insertion is performed is that every letter determines what can be inserted after it. The catenation operation will then be obtained by using a particular case of *controlled sequential insertion* (shortly, controlled SIN).

Definition 2 *Let L be a language over the alphabet Σ. For each letter a of the alphabet, let $\Delta(a)$ be a language over an alphabet Σ_a. The Δ-controlled sequential insertion into L (shortly, controlled SIN), is defined as:*

$$L \leftarrow\!\!\dashv \Delta = \bigcup_{u \in L} (u \leftarrow\!\!\dashv \Delta), \ \textit{where}$$

$$u \leftarrow\!\!\dashv \Delta = \{u_1 a v_a u_2 |\ u = u_1 a u_2, v_a \in \Delta(a)\}.$$

The function $\Delta : \Sigma \longrightarrow 2^{\Sigma'^*}$, where $\Sigma' = \bigcup_{a \in \Sigma} \Sigma_a$ is called a control function.

Example 1 Let $L = \{a^3b, bc, \lambda, d^2\}$ and let Δ be the control function defined by $\Delta(a) = \{e, \lambda\}$, $\Delta(b) = \{f\}$, $\Delta(c) = \emptyset$, $\Delta(d) = \{d\}$. Then,

$$L \leftarrow\!\!\dashv \Delta = \{a^3b, aea^2b, a^2eab, a^3eb, a^3bf, bfc, d^3\}.$$

□

All the insertion operations defined before have been binary operations between languages. The controlled insertion is an $(n+1)$-ary operation, where n is the cardinality of Σ, the domain of the control function.

If we impose the restriction that for a distinguished letter $b \in \Sigma$, $\Delta(b) = L_2$, and $\Delta(a) = \emptyset$ for any letter $a \neq b$, we obtain a special case of controlled SIN, the *sequential insertion next to the letter* b, denoted $L \overset{b}{\longleftarrow\!\!\shortmid} L_2$. The SIN next to a letter is a binary operation. The words from L which do not contain the letter b do not contribute to the result. A word in $L \overset{b}{\longleftarrow\!\!\shortmid} L_2$ is obtained from $u \in L$ which contains b, by inserting a word of L_2 after *one* of the occurrences of b in it.

In general, the following relation holds: $L \longleftarrow\!\!\shortmid \Delta = \bigcup_{a \in \Sigma} (L \overset{a}{\longleftarrow\!\!\shortmid} \Delta(a))$.

Example 2 Let L_1, L_2 be the languages $L_1 = \{a^n c^n |\ n \geq 1\}$, $L_2 = \{d^m |\ m \geq 1\}$. The sequential insertion of L_2 into L_1, next to c, is:

$$L_1 \overset{c}{\longleftarrow\!\!\shortmid} L_2 = \{a^n c^p d^m c^q |\ n, m, p \geq 1, p + q = n\}$$

whereas the uncontrolled sequential insertion between L_1 and L_2 is:

$$L_1 \longleftarrow L_2 = (L_1 \overset{c}{\longleftarrow\!\!\shortmid} L_2) \cup (L_1 \overset{a}{\longleftarrow\!\!\shortmid} L_2) \cup L_2 L_1.$$

□

In general, if L_1, L_2 are languages over an alphabet Σ, then the sequential insertion of L_2 into L_1 can be expressed as:

$$L_1 \longleftarrow L_2 = \bigcup\nolimits_{a \in \Sigma} (L_1 \overset{a}{\longleftarrow\!\!\shortmid} L_2) \cup L_2 L_1.$$

Note that the sequential insertion $L_1 \longleftarrow L_2$ can be obtained from the controlled SIN by using a marker $\#$ and a control function Δ which has the value L_2 for every letter of $\Sigma \cup \{\#\}$. Indeed,

$$L_1 \longleftarrow L_2 = h(\#L_1 \longleftarrow\!\!\shortmid \Delta),$$

where $\Delta(\#) = \Delta(a) = L_2, \forall a \in \Sigma$ and h is the morphism defined by $h(\#) = \lambda, h(a) = a, \forall a \in \Sigma$.

The catenation of the languages L_1 and L_2 can be obtained using a marker and the sequential insertion next to the marker. Indeed,

$$L_1 L_2 = h(L_1 \# \overset{\#}{\longleftarrow\!\!\shortmid} L_2),$$

where h is the morphism that erases the marker.

Note that in both the controlled SIN and the SIN next to a letter, the empty word does not occur in the result of the operation. Indeed, the notion of control implies the presence of at least one letter in the word in which the insertion is

performed. Therefore, even if the word to be inserted is λ, the result contains at least the "control" letters. As it does not contain any control letter, the presence of λ in L_1 does not affect the result of the controlled SIN,

$$(L_1 \leftarrow\!\!\dashv \Delta) = (L_1 - \{\lambda\}) \leftarrow\!\!\dashv \Delta.$$

The parallel variant of controlled insertion is defined similarily with the parallel insertion, but in the controlled case every letter defines the language that may be inserted after it.

In the case of controlled SIN or PIN, we cannot talk about commutativity or associativity. However, these notions can be studied in case of SIN (respectively PIN) next to a letter. Both operations are neither commutative nor associative.

Theorem 5 *Any family of languages closed under λ-free gsm and λ-free substitutions is closed under controlled sequential insertion.*

Proof. The proof is similar to the one of Theorem 1. One constructs the λ-free gsm g which inserts after each letter $a \in \Sigma$ a different marker $\#_a$. Then a substitution σ replacing each marker $\#_a$ with the language which corresponds to the control letter a is used.

The theorem follows as for any language L and control function Δ we have $\sigma(g(L)) = L \leftarrow\!\!\dashv \Delta$. □

Theorem 6 *Any family of languages closed under λ-free substitution is closed under controlled parallel insertion.*

Proof. Similar to the one of Theorem 2. In this case, the substitution is defined as $s(a) = a\Delta(a)$ for each control letter a in Σ and the corresponding language $\Delta(a)$. □

5 Scattered insertion

All the previously defined types of insertion were "compact". The word to be inserted was treated "as a whole". A "scattered " type of insertion can be considered as well. Instead of inserting a word, we sparsely insert its letters. If the inserted letters are in the same order as in the original, we obtain the well-known *shuffle* operation (see [12], p.156):

$$u \amalg v = \{u_1 v_1 u_2 v_2 \dots u_k v_k |\ u = u_1 \dots u_k, v = v_1 \dots v_k, k \geq 1\}.$$

Else, the *permuted scattered insertion* is obtained:

$$u \leftsquigarrow v = u \amalg \mathrm{com}(v).$$

The permuted scattered insertion induces a monoid structure on $\mathcal{P}(\Sigma^*)$.

As in the case of permuted sequential insertion, the mixture with the commutative closure implies the nonclosure of REG and CF under the new operation.

Proposition 4 *The family of regular and the family of context-free languages are not closed under permuted scattered SIN with regular languages.*

Proof. Let L_1, L_2 be two languages over an alphabet Σ. If $L_1 = \lambda$, then the permuted scattered SIN amounts to permuted SIN, therefore we can use the proof of Proposition 3. □

The closure of CS under both commutative closure and shuffle implies its closure under permuted scattered SIN.

6 Conclusions

This note summarizes algebraic and closure properties of various families of languages under insertion operations. Contrary to the customary style of column writing we have provided detailed proofs for the results, in order to point out interconnections with classical language theory.

The insertion operations, which generalize the well-known catenation of languages, can also be viewed as substitutions on the empty word, our work aiming thus to a better understanding of substitution. Various related problems have recently been investigated. Further generalizations of substitution have appeared in [2]. The dual deletion operations corresponding to the insertion ones (which generalize the left/right quotients on languages), their properties and several notions connected with them have been studied in [7], [9], [10]. In [8] the decidability of the existence of solutions to language equations involving insertion and deletion operations have been considered.

Moreover, the fact that insertion and deletion are in some sense inverse to each other has led to several non-language-theoretic applications, for example in cryptography. The connections to cryptography arised from variations of the *garbage-in-between* method, where encryption is realized by inserting in certain ways "garbage letters" in the original message, whereas decryption is made by deleting them (see [1]). The study of suitable pairs of insertion/deletion operations having the properties requested by practical cryptographical purposes is in progress.

Acknowledgements

We would like to thank Professor Arto Salomaa, Professor Grzegorz Rozenberg and Dr.Jarkko Kari for extended discussions and valuable suggestions.

References

[1] M.Andrasiu, J.Dassow, G.Paun, A.Salomaa. Language-theoretic problems arising from Richelieu cryptosystems. To appear in *Theoretical Computer Science.*

[2] A.Atanasiu, V.Mitrana. Substitutions on words and languages. To appear in *Proceedings of "Developments in language theory"*, Turku University, Finland, July 12-15, 1993.

[3] R.Book, M.Jantzen, C.Wrathall. Monadic Thue systems. *Theoretical Computer Science*, 19 (1982), pp.231-251.

[4] R.Book, M.Jantzen, C.Wrathall. Erasing strings. *Lecture Notes in Computer Science*, 104 (1981), pp.252-260.

[5] S.Greibach. Full AFL's and nested iterated substitution. *Information and Control*, 16 (1970), pp.7-35.

[6] L.Kari. On insertion and deletion in formal languages. *Ph.D. Thesis*, University of Turku, 1991.

[7] L.Kari. Deletion operations: closure properties. Submitted.

[8] L.Kari. On some language equations. Submitted.

[9] L.Kari, A.Mateescu, Gh.Paun, A.Salomaa. Deletion sets. To appear in *Fundamenta Informaticae*.

[10] L.Kari, A.Mateescu, Gh.Paun, A.Salomaa. On parallel deletions applied to a word. Submitted.

[11] H.C.M.Kleijn, G.Rozenberg. Context-free like restrictions on selective rewriting. *Theoretical Computer Science*, vol.16, 3 (1981), pp.237-269.

[12] W.Kuich, A.Salomaa. *Semirings, Automata, Languages*. Springer Verlag, Berlin, 1986.

[13] G.Rozenberg, A.Salomaa. *The Mathematical Theory of L Systems*. Academic Press, London, 1980.

[14] A.Salomaa. *Theory of Automata*. Pergamon Press, Oxford, 1969.

[15] A.Salomaa. *Formal Languages*. Academic Press, London, 1973.