Chapter 3

Sequential and parallel deletion

3.1 Deletion

In this section the operations of *sequential deletion* (introduced in [13]) and *parallel deletion* will be studied. The deletion of the word v from u is a generalization of the right and left quotients u/v, $v\backslash u$: instead of extracting the word v from the right or left extremity of u, we extract it from an arbitrary place in u.

Definition 3.1 *Let L_1, L_2 be languages over the alphabet Σ. The sequential deletion (abbreviated SD in the sequel) of L_2 from L_1 is defined as:*

$$L_1 \rightarrow L_2 = \bigcup_{u \in L_1, v \in L_2} (u \rightarrow v)$$

where

$$u \rightarrow v = \quad \{w \in \Sigma^* |\ u = w_1 v w_2, w = w_1 w_2,\ w_1, w_2 \in \Sigma^*\}.$$

While the sequential insertion operation is a total operation in the sense that essentially all the words from both languages contribute to the result, the sequential deletion is a partial operation in this sense. The words from L_1 which do not contain any word of L_2 as a subword, as well as the words from L_2 which are not subwords of any word of L_1, do not contribute to the

result. Indeed, the pairs of words $u \in L_1$, $v \in L_2$ where v is not a subword of u, give the empty set as their contribution to the union.

The sequential insertion and the sequential deletion are not inverse operations. In general, if $L_1 \leftarrow\!\!\!\cdot\, L_2 = L_3$ then $L_1 \subseteq L_3 \rightarrow\!\!\!\cdot\, L_2$ and if $L_1' \rightarrow\!\!\!\cdot\, L_2' = L_3'$ then $L_1' \subseteq L_3' \leftarrow\!\!\!\cdot\, L_2'$, but the reverse inclusions do not hold. Some particular cases in which the sequential insertion and deletion are inverse to each other are studied in Section 4.4.

Example 3.1 Let $L_1 = \{abababa, ab, ba^2, aba\}$, $L_2 = \{aba\}$. The sequential deletion of L_2 from L_1 is:

$$L_1 \rightarrow\!\!\!\cdot\, L_2 = \{baba, abba, abab, \lambda\},$$

which is the union of the sets:

$$\begin{aligned} \{abababa\} \rightarrow\!\!\!\cdot\, \{aba\} &= \{baba, abba, abab\}, \\ \{ab\} \rightarrow\!\!\!\cdot\, \{aba\} &= \emptyset, \\ \{ba^2\} \rightarrow\!\!\!\cdot\, \{aba\} &= \emptyset, \\ \{aba\} \rightarrow\!\!\!\cdot\, \{aba\} &= \{\lambda\}. \end{aligned}$$

□

The left and right quotient can be obtained using the sequential deletion and a marker which forces the position of the deletion. If L_1, L_2 are languages over Σ then,

$$L_2 \backslash L_1 = (\#L_1) \rightarrow\!\!\!\cdot\, (\#L_2),$$

$$L_1 / L_2 = (L_1\#) \rightarrow\!\!\!\cdot\, (L_2\#),$$

where $\#$ is a new symbol which does not belong to Σ.

A parallel variant of the deletion can be defined as follows. Given words u and v, the *parallel deletion* of v from u consists of the words obtained by simultaneously erasing from u all the non-overlapping occurrences of v. The definition is extended to languages in the natural way. Given a word u and a language L_2, the parallel deletion $u \Rightarrow L_2$ consists of the words obtained by erasing from u all the non-overlapping occurrences of words in L_2.

Definition 3.2 *Let L_1, L_2 be languages over the alphabet Σ. The parallel deletion (shortly, PD) of L_2 from L_1 is:*

$$L_1 \Longrightarrow L_2 = \bigcup_{u \in L_1} (u \Longrightarrow L_2)$$

where

$$\begin{aligned} u \Longrightarrow L_2 = \quad & \{u_1 u_2 \dots u_k u_{k+1} |\ k \geq 1, u_i \in \Sigma^*, 1 \leq i \leq k+1 \text{ and} \\ & \exists v_i \in L_2, 1 \leq i \leq k \text{ such that } u = u_1 v_1 \dots u_k v_k u_{k+1}, \\ & \text{where } \{u_i\} \cap [\Sigma^*(L_2 - \{\lambda\})\Sigma^*] = \emptyset, 1 \leq i \leq k+1\}. \end{aligned}$$

The parallel deletion $u \Longrightarrow L_2$ erases from u the non-overlapping occurrences of words from L_2. Moreover, a supplementary condition has to be fulfilled: between two occurrences of words of L_2 to be erased, no nonempty word from L_2 appears as a subword. This assures that *all* occurrences of words from L_2 have been erased from u, and is taken care of by the last line of the definition. The reason why λ had to be excluded from L_2 is obvious. If this wouldn't be the case and λ would belong to L_2, the condition $\{u_i\} \cap \Sigma^* L_2 \Sigma^* = \emptyset$ would imply $\{u_i\} \cap \Sigma^* = \emptyset$ – a contradiction. Note that words from L_2 can still appear as subwords in $u \Longrightarrow L_2$, as the result of catenating the remaining pieces of u.

Example 3.2 Let $L_1 = \{abababa, aababa, abaabaaba\}$, $L_2 = \{aba\}$. The parallel deletion of L_2 from L_1 is:

$$L_1 \Longrightarrow L_2 = \{b, abba, aba, aab, \lambda\},$$

being the union of the sets:

$$\begin{aligned} \{abababa\} \Longrightarrow \{aba\} = \quad & \{b, abba\}, \\ \{aababa\} \Longrightarrow \{aba\} = \quad & \{aba, aab\}, \\ \{abaabaaba\} \Longrightarrow \{aba\} = \quad & \{\lambda\}. \end{aligned}$$

□

The right and left quotient can be obtained by using the parallel deletion and a marker which forces the operation to become sequential and also fixes the position of the deletion. If L_1, L_2 are languages over the alphabet Σ,

$$L_1/L_2 = (L_1\#) \Longrightarrow (L_2\#),$$

$$L_2 \backslash L_1 = (\#L_1) \Longrightarrow (\#L_2),$$

where # is a new symbol which does not belong to Σ.

The sequential and parallel deletion are not commutative operations. Indeed, take $L_1 = \{ab\}$ and $L_2 = \{b\}$. Then we have:

$$(ab \longrightarrow b) = (ab \Longrightarrow b) = a \neq (b \longrightarrow ab) = (b \Longrightarrow ab) = \emptyset.$$

The sequential and parallel deletion are not associative operations. For example, take $L_1 = \{aab\}$, $L_2 = \{b\}$ and $L_3 = \{a\}$. Then we have:

$$\begin{array}{ll} (aab \longrightarrow b) \longrightarrow a = & aa \longrightarrow a = a, \text{ whereas} \\ aab \longrightarrow (b \longrightarrow a) = & aab \longrightarrow \emptyset = \emptyset, \\ \text{and} & \\ (aab \Longrightarrow b) \Longrightarrow a = & aa \Longrightarrow a = \lambda, \text{ whereas} \\ aab \Longrightarrow (b \Longrightarrow a) = & aab \Longrightarrow \emptyset = \emptyset. \end{array}$$

In general, the sets $L_1 \longrightarrow (L_2 \longrightarrow L_3)$ and $(L_1 \longrightarrow L_2) \longrightarrow L_3$ are not comparable and the same thing can be said about the sets $L_1 \Longrightarrow (L_2 \Longrightarrow L_3)$ and $(L_1 \Longrightarrow L_2) \Longrightarrow L_3$. This fact can be proved by using the preceding and the following examples:

$$ab \longrightarrow (bc \longrightarrow c) = ab \Longrightarrow (bc \Longrightarrow c) = a,$$
$$(ab \longrightarrow bc) \longrightarrow c = (ab \Longrightarrow bc) \Longrightarrow c = \emptyset.$$

The following result is known for the right and left quotient of regular languages (see, for example, [4], p.50):

Theorem 3.1 *If L_1 is a regular language and L_2 is an arbitrary one, then the left quotient of L_1 by L_2 is a regular language.*

Proof. Let L_1, L_2 be languages over an alphabet Σ and let $A = (S, \Sigma, s_0, F, P)$ be a finite automaton that accepts L_1. For every two states s, s' in S define:

$$L_{s,s'} = \{w \in \Sigma^* | \; sw \Longrightarrow^* s' \text{ in } A\}.$$

Consider now the finite automaton $A' = (S, \Sigma \cup \{\#\}, s_0, F, P')$ where:

$$P' = P \cup \{s_0 \# \longrightarrow s' | \; L_2 \cap L_{s_0,s'} \neq \emptyset\}$$

and $\#$ is a new symbol which does not occur in Σ.

If one now defines the morphism $h : (\Sigma \cup \{\#\})^* \longrightarrow \Sigma^*$, $h(\#) = \lambda$, $h(a) = a$, $\forall a \in \Sigma$, it is easy to show that:

$$L_2 \backslash L_1 = h(L(A') \cap \# \Sigma^*).$$

The theorem follows as the family of regular languages is closed under intersection and morphism. □

Corollary 3.1 *The language $L_2 \backslash L_1$ from the preceding theorem can be effectively constructed if L_2 is a regular or a context-free language.*

Proof. The family of context-free (regular) languages is closed under intersection with regular languages. Consequently, all languages $L_2 \cap L_{s_0,s'}$ from the previous theorem are context-free (regular) and can be effectively constructed. As the emptiness problem is decidable for context-free (regular) languages, the automaton A' can be effectively constructed. □

Corollary 3.2 *If L_1 is a regular language, there exist finitely many different languages that can be obtained from L_1 by left quotient.*

Proof. The claim follows from the preceding theorem, by the fact that the automaton A is finite. There exist finitely many different possibilities of constructing the automaton A'. The languages that can be obtained from L_1 by left quotient will be among the languages:

$$L_{S'} = h(L(A_{S'}) \cap \#\Sigma^*),$$

where h is defined as in the theorem, $S' \subseteq S$ and $A_{S'}$ is the finite automaton:

$$A_{S'} = (S, \Sigma \cup \{\#\}, s_0, F, P_{S'}),$$

$$P_{S'} = P \cup \{s_0 \# \longrightarrow s' |\ s' \in S'\}.$$

Consequently, there exist at most $2^{\mathrm{card}(S)}$ different languages that can be obtained from L_1 by left quotient. □

Results similar to Theorem 3.1, Corollary 3.1 and Corollary 3.2 can be proved also for the right quotient, as we have:

$$L_1 / L_2 = \mathrm{Mi}(\mathrm{Mi}(L_2) \backslash \mathrm{Mi}(L_1)).$$

In order to show the closure of REG under sequential deletion, we will prove a lemma which generalizes these results. It will be shown that a regular language results when an arbitrary language is sequentially deleted from a regular one.

Lemma 3.1 *If L_1, L_2 are languages over the alphabet Σ, L_1 a regular one, then $L_1 \longrightarrow L_2$ is a regular language.*

Proof. Let L_1, L_2 be languages over Σ and let $A = (S, \Sigma, s_0, F, P)$ be a finite automaton that accepts L_1. For two states s, s' in S denote:

$$L_{s,s'} = \{w \in \Sigma^* |\ sw \Longrightarrow^* s' \text{ in } A\}.$$

The language $L_{s,s'}$ is regular for each $s, s' \in S$. Consider the automaton:

$$\begin{aligned} A' = &\quad (S, \Sigma \cup \{\#\}, s_0, F, P') \\ P' = &\quad P \cup \{s\# \longrightarrow s' |\ s, s' \in S \text{ and } L_2 \cap L_{s,s'} \neq \emptyset\}, \end{aligned}$$

where $\#$ is a new symbol which does not occur in Σ.

Claim.

$$L_1 \rightarrow L_2 = h(L(A') \cap \Sigma^* \# \Sigma^*)$$

where h is the morphism $h : (\Sigma \cup \{\#\})^* \longrightarrow \Sigma^*$, $h(\#) = \lambda$, $h(a) = a$, $\forall a \in \Sigma$.

" $\subseteq$ " Let u be a word in $L_1 \rightarrow L_2$. There exist $w \in L_1$, $v \in L_2$ such that $w = u_1 v u_2$, $u = u_1 u_2$.

The following derivation exists in A:

$$s_0 w = s_0 u_1 v u_2 \Longrightarrow^* s_1 v u_2 \Longrightarrow^* s_1' u_2 \Longrightarrow^* s_f, s_f \in F.$$

As $v \in L_2 \cap L_{s_1,s_1'}$, the rule $s_1\# \longrightarrow s_1'$ exists in P' and one can construct in A' the derivation:

$$s_0 u_1 \# u_2 \Longrightarrow^* s_1 \# u_2 \Longrightarrow s_1' u_2 \Longrightarrow^* s_f, s_f \in F,$$

which proves that $u_1 \# u_2 \in L(A') \cap \Sigma^* \# \Sigma^*$.

As $u = h(u_1 \# u_2)$, one deduces that u belongs to $h(L(A') \cap \Sigma^* \# \Sigma^*)$.

" $\supseteq$ " Let w be a word in $h(L(A') \cap \Sigma^* \# \Sigma^*)$. There exists a word $w' \in L(A') \cap \Sigma^* \# \Sigma^*$ such that $h(w') = w$.

The word w' is of the form $u_1 \# u_2$, $u_1, u_2 \in \Sigma^*$, and there also exists a derivation

$$s_0 u_1 \# u_2 \Longrightarrow^* s_f, s_f \in F,$$

in the automaton A'.

The derivation has the form:

$$s_0 u_1 \# u_2 \Longrightarrow^* s_1 \# u_2 \Longrightarrow s_1' u_2 \Longrightarrow^* s_f, s_f \in F,$$

where the rule $s_1\# \longrightarrow s_1'$ has been applied that is, $L_2 \cap L_{s_1,s_1'} \neq \emptyset$. This further implies that there exists $v \in L_2$ such that a derivation $s_1 v \Longrightarrow^* s_1'$ exists in A.

Because w' contains only one marker, only one rule from $P' - P$ has been used in the derivation, all the other rules being from P. Therefore the following derivation exists in A:

$$s_0u_1vu_2 \Longrightarrow^* s_1vu_2 \Longrightarrow^* s_1'u_2 \Longrightarrow^* s_f, s_f \in F,$$

which shows that $u_1vu_2 \in L_1$. As $v \in L_2$ one concludes that $w = u_1u_2 = h(u_1\#u_2) \in (u_1vu_2 \rightarrow v) \subseteq (L_1 \rightarrow L_2)$, which proves the second inclusion. The theorem follows as REG is closed under morphism and intersection with regular languages. □

Corollary 3.3 *The family of regular languages is closed under sequential deletion.*

Corollary 3.4 *The language $L_1 \rightarrow L_2$ can be effectively constructed if L_1 is a regular language and L_2 a regular or context-free language.*

Proof. If L_2 is a regular or context-free language, the emptiness of the intersection $L_2 \cap L_{s,s'}$ is decidable. Consequently, the automaton A' can be effectively constructed.

Corollary 3.5 *For any regular language L_1 there exist finitely many languages that can be obtained from L_1 by sequential deletion.*

Proof. The claim follows from the preceding lemma by the fact that the automaton A is finite. This implies that there are finitely many different possibilities of constructing the automaton A'. The languages that can be obtained from L_1 by sequential deletion will be among the languages:

$$L_{S'} = h(L(A_{S'}) \cap \Sigma^*\#\Sigma^*),$$

where S' is an arbitrary subset of $S \times S$ and $A_{S'}$ is the automaton:

$$\begin{aligned} A_{S'} &= (S, \Sigma \cup \{\#\}, s_0, F, P_{S'}), \\ P_{S'} &= P \cup \{s\# \longrightarrow s' \mid (s, s') \in S'\}. \end{aligned}$$

Consequently, the number of distinct languages that can be obtained from L_1 by sequential deletion is at most $2^{\mathrm{card}(S\times S)}$. □

If the language to be deleted is a regular one, the sequential deletion can be simulated by a generalized sequential machine with erasing.

Theorem 3.2 *If L and R are languages over the alphabet Σ, R a regular one, there exists a gsm g such that*

$$L \rightarrow R = g(L) \cup \{\lambda |\ \lambda \in L \cap R\}.$$

Proof. Let $A = (S, \Sigma, s_0, F, P)$ be a finite automaton that recognizes the language R. Construct the gsm with erasing,

$$\begin{aligned} g = \quad & (\Sigma, \Sigma, S \cup \{s_0', s_f\}, s_0', \{s_f\}, P'), \\ P' = \quad & P \cup \{s_0'a \longrightarrow as_0' |\ a \in \Sigma\} \cup \{s_0'a \longrightarrow s |\ s_0a \longrightarrow s \in P\} \cup \\ & \{sa \longrightarrow s_f |\ sa \longrightarrow s' \in P, s' \in F\} \cup \{s_fa \longrightarrow as_f |\ a \in \Sigma\} \cup \\ & \{s_0'a \longrightarrow s_f |\ s_0a \longrightarrow s \in P, s \in F\} \cup \{s_0'a \longrightarrow as_f |\ a \in \Sigma, \lambda \in R\}, \end{aligned}$$

which satisfies the requested equality.

Given a word $v \in L$ as an input and a word $w \in R$, the gsm g works as follows: the rules of P erase the word w from v while the ones of the type $s_0'a \longrightarrow as_0'$ and $s_fa \longrightarrow as_f$ cross over the letters which will remain in $v \rightarrow w$.

Indeed, let $u \in L \rightarrow R$. There exist words $v \in L$, $w \in R$ such that $v = u_1wu_2$, $u = u_1u_2$. As w belongs to R, it is accepted by the automaton A and therefore there exists:

$$s_0w \Longrightarrow^* s', s' \in F, \text{ in } A.$$

It will be shown in the following that $u \in g(v) \cup \{\lambda |\ \lambda \in L \cap R\}$. Indeed,

(i) If $w \neq \lambda$, $u \in g(v)$ as

$$s_0'v = s_0'u_1wu_2 \Longrightarrow^* u_1s_0'wu_2 \Longrightarrow^* u_1s_fu_2 \Longrightarrow^* u_1u_2s_f$$

is a derivation according to g. In the scanning of u_1 and u_2 rules of the type $s_0'a \longrightarrow as_0'$ and respectively $s_fa \longrightarrow as_f$ have been used. If $\lg(w) = 1$ then, for scanning w, the rule $s_0'w \longrightarrow s_f$ alone is used. Else, while scanning w, the derivation $s_0w \Longrightarrow^* s'$ has been used, with the first rule $s_0a \longrightarrow s''$ replaced by $s_0'a \longrightarrow s''$ and the last rule $sa \longrightarrow s'$ replaced by $sa \longrightarrow s_f$. Note that if $u_1 = \lambda$ (respectively $u_2 = \lambda$) no rule of the type $s_0'a \longrightarrow as_0'$ (respectively $s_fa \longrightarrow as_f$) is needed.

(ii) If $w = \lambda$ and $u \neq \lambda$, $u \in g(v)$ as

$$s_0'v = s_0'u \Longrightarrow as_fu' \Longrightarrow^* au's_f, u = au',$$

is a derivation according to g. The first rule applied is $s_0'a \longrightarrow as_f$ and in the rest rules $s_fa \longrightarrow as_f$ have been used (if $u' = \lambda$, no rule $s_fa \longrightarrow as_f$ is needed).

(iii) If $w = \lambda$ and $u = \lambda$ no derivation in g can be constructed but, as $\lambda \in L \cap R$, λ belongs also to the second member of the equality.

In all cases it resulted that $u \in g(L) \cup \{\lambda |\ \lambda \in L \cap R\}$, therefore one of the inclusions is proved.

In order to prove the reverse inclusion, let u be a word in $g(L) \cup \{\lambda |\ \lambda \in L \cap R\}$. If u belongs to the second set of the union then $u \in L \longrightarrow R$ as $\lambda \in L \cap R$ and $u = \lambda$. Else, if $u \in g(L)$, there exists $v \in L$ such that

$$s_0' v \Longrightarrow^* u s_f$$

according to the rules of P'.

If during the derivation a rule of the type $s_0' a \longrightarrow a s_f$ has been applied, then $\lambda \in R$ and the derivation has the form:

$$\begin{aligned} {s_0}' v = \quad & s_0' u_1 a u_2 \Longrightarrow^* u_1 s_0' a u_2 \Longrightarrow u_1 a s_f u_2 \Longrightarrow^* \\ & u_1 a u_2 s_f = u s_f = v s_f. \end{aligned}$$

This implies that $u \in (v \longrightarrow \lambda) \subseteq (L \longrightarrow R)$.

If no rule $s_0' a \longrightarrow a s_f$ has been applied during the derivation, s_f can be reached only by using a rule $s_0' a \longrightarrow s_f$ (originating from $s_0 a \longrightarrow s' \in P, s' \in F$) or $sa \longrightarrow s_f$ (originating from $sa \longrightarrow s' \in P, s' \in F$). In the first case the derivation is:

$$\begin{aligned} s_0' v = \quad & s_0' u_1 a u_2 \Longrightarrow^* u_1 s_0' a u_2 \Longrightarrow u_1 s_f u_2 \Longrightarrow^* \\ & u_1 u_2 s_f = u s_f, \text{ where } a \in R. \end{aligned}$$

In the second case, we notice that a state from S can be introduced only by a rule $s_0' a \longrightarrow s''$ (originating from $s_0 a \longrightarrow s'' \in P$). Moreover, between the application of the rules $s_0' a \longrightarrow s''$ and $sa \longrightarrow s_f$ only rules from P can be used. Notice finally that before the application of $s_0' a \longrightarrow s''$ only rules $s_0' a \longrightarrow a s_0'$ are possible and, after $sa \longrightarrow s_f$, only rules $s_f a \longrightarrow a s_f$ can be applied.

From the previous observations one can conclude that the derivation has to be of the form:

$$\begin{aligned} s_0' v = \quad & s_0' u_1 w u_2 \Longrightarrow^* u_1 s_0' w u_2 \Longrightarrow^* u_1 s_f u_2 \Longrightarrow^* \\ & u_1 u_2 s_f = u s_f, \end{aligned}$$

where $s_0 w \Longrightarrow^* s'$, $s' \in F$ is a derivation in P.

Both cases led to the conclusion that $v = u_1 w u_2 \in L$, $w \in R$, $u = u_1 u_2$ that is, $u \in (v \longrightarrow w) \subseteq (L \longrightarrow R)$, and the proof of the theorem is thus complete. □

Corollary 3.6 *The family of context-free languages is closed under sequential deletion with regular languages.*

Proof. The claim follows from the preceding theorem and the fact that CF is closed under union and gsm mapping. □

If the language to be deleted is regular, the parallel deletion $L \Longrightarrow R$ can be expressed as a morphic image of the intersection between a regular language and the image of L under a rational transduction.

Theorem 3.3 *Let L, R be languages over the alphabet Σ, L a λ-free language and R a regular one. There exist a rational transducer g, a morphism h and a regular language R' such that:*

$$L \Longrightarrow R = h(g(L) \cap R').$$

Proof. Let $A = (S, \Sigma, s_0, F, P)$ be a finite automaton that accepts the language R. Let us consider the rational transducer:

$$g = (\Sigma, \Sigma \cup \{\#\}, S \cup \{s_0'\}, s_0', \{s_0'\}, P'),$$

$$\begin{aligned} P' = \; & P \cup \{s_0' a \longrightarrow a s_0' \mid a \in \Sigma\} \cup \\ & \{s_0' a \longrightarrow s \mid s_0 a \longrightarrow s \in P, a \in \Sigma\} \cup \\ & \{s_0' a \longrightarrow \# s_0' \mid s_0 a \longrightarrow s \in P, a \in \Sigma, s \in F\} \cup \\ & \{sa \longrightarrow \# s_0' \mid sa \longrightarrow s' \in P, a \in \Sigma, s' \in F\} \cup \\ & \{s_0' \longrightarrow \# s_0' \mid \lambda \in R\}. \end{aligned}$$

The rational transducer g performes the following task: given a word of L as an input, it replaces arbitrary many words of R from it with the marker #.

Claim.

$$\begin{aligned} g(L) = \; & L \cup \{u_1 \# u_2 \# \dots u_k \# u_{k+1} \mid k \geq 1, u_i \in \Sigma^*, 1 \leq i \leq k+1 \\ & \text{and } \exists u \in L, v_i \in R, 1 \leq i \leq k : u = u_1 v_1 \dots u_k v_k u_{k+1}\}. \end{aligned}$$

" $\supseteq$ " Let w be a word in the right member of the equality.
If $w \in L$, then one can construct the derivation according to g:

$$s_0' w \Longrightarrow^* w s_0',$$

where only rules of the type $s_0' a \longrightarrow a s_0'$ have been applied. This shows that $w \in g(L)$.

Else, $w = u_1\#\ldots\#u_k\#u_{k+1}$ and there exist $u \in L$, $v_1, \ldots v_k \in R$ such that $u = u_1v_1 \ldots u_kv_ku_{k+1}$. For every i, $1 \leq i \leq k$ there exists a derivation $s_0v_i \Longrightarrow^* s_i$, $s_i \in F$ according to A. The following derivation according to g can be constructed:

$$\begin{aligned} s_0{}'u = \quad & s_0'u_1v_1 \ldots u_kv_ku_{k+1} \Longrightarrow^* u_1s_0'v_1u_2v_2 \ldots u_kv_ku_{k+1} \Longrightarrow^* \\ & u_1\#s_0'u_2v_2 \ldots u_kv_ku_{k+1} \Longrightarrow^* u_1\#u_2s_0'v_2 \ldots u_kv_ku_{k+1} \Longrightarrow^* \\ & u_1\#u_2\#\ldots u_k\#s_0'u_{k+1} \Longrightarrow^* u_1\#u_2\#\ldots u_k\#u_{k+1}s_0' = ws_0'. \end{aligned}$$

Rules of the type $s_0'a \longrightarrow as_0'$ have been used to scan $u_i \neq \lambda$, $1 \leq i \leq k+1$. They are not needed if $u_i = \lambda$. When scanning $v_i, 1 \leq i \leq k$, if $v_i = \lambda$, the rule $s_0' \longrightarrow \#s_0'$ has been applied. If $\lg(v_i) = 1$ then, for scanning v_i, only the rule $s_0'a \longrightarrow \#s_0'$ has been used. If v_i contains more than one letter, the derivation $s_0v_i \Longrightarrow^* s_i, s_i \in F$ has been used, with the rule $s_0a \longrightarrow s'$ replaced by $s_0'a \longrightarrow s'$ and the rule $s''b \longrightarrow s_i$ by $s''b \longrightarrow s_0'$.

This shows that $w \in g(u) \subseteq g(L)$.

" $\subseteq$ " Let w be a word in $g(L)$. There exist $u \in L$ and a derivation $s_0'u \Longrightarrow^* ws_0'$ according to g.

If during the derivation only rules of the type $s_0'a \longrightarrow as_0'$ have been applied then $w \in L$ which is included in the right member of the equality.

Else, at least one subderivation which leads to a marker has been used. The word w has then the form $w = u_1\#u_2\#\ldots u_k\#u_{k+1}$, $u_i \in \Sigma^*$, $1 \leq i \leq k+1$. We have to show that there exist $u \in L$ and $v_i \in R$, $1 \leq i \leq k$ such that $u = u_1v_1u_2v_2 \ldots u_kv_ku_{k+1}$.

Analyzing the rules of g one notices that they do not produce anything except the marker #. Except the rules that produce #, the others just leave the input letters unchanged, or erase them. This means that the word u has the form:

$$u = u_1v_1u_2v_2 \ldots u_kv_ku_{k+1}, v_i \in \Sigma^*, 1 \leq i \leq k,$$

and its derivation is:

$$s_0'u_1v_1u_2v_2 \ldots u_kv_ku_{k+1} \Longrightarrow^* u_1\#u_2\#\ldots\#u_{k+1}s_0'.$$

Moreover, it follows from the previous observations that in scanning $u_i \neq \lambda$ only rules $s_0'a \longrightarrow as_0'$ have been applied. While parsing v_i, $1 \leq i \leq k$ no such rules have been used, as no letter appears between u_i's. Instead, while parsing every v_i, $1 \leq i \leq k$, a marker # has been produced.

All that remains to be proved is that $v_i \in R$, $\forall 1 \leq i \leq k$.

Let us examine the parsing of a word v_i, $1 \leq i \leq k$. It will start with the current state being s'_0 (the scanning of u begins with s'_0 and that of u_i ends also in s'_0). For a marker to appear, one of the rules $s'_0 \longrightarrow \#s'_0$, $s'_0 a \longrightarrow \#s'_0$ or $sa \longrightarrow \#s'_0$ has to be used.

– If the rule $s'_0 \longrightarrow \#s'_0$ has been applied, it is also the last rule of the subderivation as otherwise undesired letters appear between u_i and u_{i+1}. This means that $v_i = \lambda$ and it belongs to R because this is the condition under which the rule $s'_0 \longrightarrow \#s'_0$ was introduced in P'.

– If the rule $s'_0 a \longrightarrow \#s'_0$ has been applied, one can deduce as before that this is also the last rule of the subderivation, which has the form:

$$s'_0 v_i = s'_0 a \Longrightarrow \#s'_0, s_0 a \longrightarrow s \in P, s \in F.$$

The existence of the derivation in A proves that $v_i \in R$.

– If the rule $sa \longrightarrow \#s'_0$ (where $sa \longrightarrow s'' \in P$, $s'' \in F$) has been applied, we notice that a state $s \in S$ can be reached only if a rule $s'_0 a \longrightarrow s'$, $s' \in S$, followed by some rules from P have been used. As the production $s'_0 a \longrightarrow s'$ originates from the rule $s_0 a \longrightarrow s' \in P$, the derivation has the form:

$$s'_0 v_i = \quad s'_0 a_1 a_2 \dots a_p \Longrightarrow s' a_2 \dots a_p \Longrightarrow^* s a_p \Longrightarrow \#s'_0,$$

where

$$s_0 a_1 a_2 \dots a_p \Longrightarrow s' a_2 \dots a_p \Longrightarrow^* s a_p \Longrightarrow s'', s'' \in F,$$

is a derivation in A, that is, $v_i \in R$.

In all the considered cases we have reached the conclusion that v_i belongs to R, therefore the proof of the claim is complete.

Let us return now to the proof of the theorem. Let h be the morphism $h : (\Sigma \cup \{\#\})^* \longrightarrow \Sigma^*$ defined by $h(\#) = \lambda$, $h(a) = a$, $a \in \Sigma$ and R' the regular set:

$$R' = [(\Sigma \cup \{\#\})^* (R - \{\lambda\})(\Sigma \cup \{\#\})^*]^c \cap (\Sigma^* \# \Sigma^*)^+.$$

It will be proved in the following that h, g and R' satisfy the requested equality. The first set of the intersection takes care that between two erased words (marked with #) no other candidate for erasing occurs. The second set takes care of that the words from L which do not contain any candidate for erasing are not retained in the final result. This is done retaining only those words in which at least one erasing (marker) occurs.

"$\subseteq$" Let α be a word in $L \Longrightarrow R$. There exist $u \in L$, $v_i \in R$, $1 \leq i \leq k$ such that:

$$\begin{aligned} u = \quad & u_1 v_1 u_2 v_2 \dots u_k v_k u_{k+1}, \\ \alpha = \quad & u_1 u_2 \dots u_k u_{k+1}, \\ & u_i \in \Sigma^*, 1 \leq i \leq k+1, \\ & \{u_i\} \cap \Sigma^*(R - \{\lambda\})\Sigma^* = \emptyset, 1 \leq i \leq k+1. \end{aligned} \qquad (*)$$

According to the previous claim, $w = u_1 \# u_2 \# \dots u_k \# u_{k+1}$ is in $g(L)$. The word w belongs also to R'. If this wouldn't be the case, w would be a word in $(\Sigma \cup \{\#\})^*(R - \{\lambda\})(\Sigma \cup \{\#\})^*$, which would imply that w contains a nonempty word from R as a subword. This, in turn, would mean that for some $1 \leq i \leq k+1$, u_i contains a nonempty word from R as a subword– a contradiction with (*). Finally it is obvious that $h(w) = \alpha$, $w \in (g(L) \cap R')$ that is, $\alpha \in h(g(L) \cap R')$.

For showing the reverse inclusion, let α be a word in $h(g(L) \cap R'))$. There exists $w \in (g(L) \cap R')$ such that $h(w) = \alpha$. As w contains at least one marker, according to the previous claim,

$$\begin{aligned} w = \quad & u_1 \# u_2 \# \dots u_k \# u_{k+1}, \\ & \exists u \in L, v_i \in R, 1 \leq i \leq k : \\ & u = u_1 v_1 u_2 v_2 \dots u_k v_k u_{k+1}. \end{aligned}$$

Because $w \in [(\Sigma \cup \{\#\})^*(R - \{\lambda\})(\Sigma \cup \{\#\})^*]^c$, none of the u_i's contains any nonempty word from R as a subword that is,

$$\{u_i\} \cap [(\Sigma \cup \{\#\})^*(R - \{\lambda\})(\Sigma \cup \{\#\})^*] = \emptyset, 1 \leq i \leq k+1.$$

We can conclude that $\alpha = h(w) = u_1 u_2 \dots u_{k+1}$ belongs to $L \Longrightarrow R$, all the conditions of the definition being satisfied. □

Corollary 3.7 *The family of regular and the family of context-free languages are closed under parallel deletion with regular languages.*

Proof. If λ is not a subword of L, the claim follows from the preceding theorem as REG and CF are closed under intersection with regular languages, rational transductions and morphism.

If λ belongs to L but not to R, then $L \Longrightarrow R = ((L - \{\lambda\}) \Longrightarrow R)$ and we can apply the previous proof for $L - \{\lambda\}$ and R.

If λ belongs to $L \cap R$, then $L \Longrightarrow R = [(L - \{\lambda\}) \Longrightarrow R] \cup \{\lambda\}$. We can use the same proof to show that $(L - \{\lambda\}) \Longrightarrow R$ is regular, respectively context-free. □

The following results show that CF and CS are closed under neither sequential nor parallel deletion. Moreover, in the context-sensitive case, there exists a language from which the PD of a single word produces a non-context-sensitive language.

Theorem 3.4 *The family of context-free languages is closed under neither sequential nor parallel deletion.*

Proof. Let L_1, L_2 be the context- free languages:

$$L_1 = \#\{a^i b^{2i} |\ i > 0\}^*,$$
$$L_2 = \#a\{b^i a^i |\ i > 0\}^*.$$

(Similar languages have been used in [3], p.40, to show that CF is not closed under left quotient.)

The language $L_1 \longrightarrow L_2$ is not context-free. Indeed, if this wouldn't be the case, then also the language

$$(L_1 \longrightarrow L_2) \cap b^+ = \{b^{2^n} |\ n > 0\}$$

would be context-free, which is a contradiction.

The same example can be used to prove that CF is not closed under parallel deletion, because the presence of the marker assures us that $L_1 \longrightarrow L_2 = L_1 \Longrightarrow L_2$. □

Theorem 3.5 *The family of context-sensitive languages is not closed under sequential deletion with regular languages.*

Proof. If L_1, L_2 are languages over the alphabet Σ, we notice that :

$$\#L_1 \longrightarrow \#L_2 = L_2 \backslash L_1,$$

where # is a symbol which does not belong to Σ and "$\backslash$" denotes the left quotient.

As the family CS is not closed under left quotient with regular languages, it follows that it is not closed under SD with regular languages either. □

Corollary 3.8 *The family of context-sensitive languages is not closed under sequential deletion.*

Theorem 3.6 *The family of context-sensitive languages is closed under sequential deletion with singletons.*

Proof. Let L be a context-sensitive language and w be a word over the same alphabet Σ. If $w \in L$ then

$$L \longrightarrow \{w\} = [(L - \{w\}) \longrightarrow \{w\}] \cup \{\lambda\}.$$

If $w = \lambda$ then $L \longrightarrow \{\lambda\} = L$. Therefore the theorem will hold if we prove that $L \longrightarrow \{w\}$ is context-sensitive for w nonempty and not belonging to L.

Let $A = (S, \Sigma, s_0, F, P)$ be a finite automaton that accepts the word w.

We can modify the proof of Theorem 3.2 such that the constructed gsm is λ-free. Indeed, let $\#$ be a new symbol which does not occur in Σ and consider the gsm:

$$\begin{aligned} g = \quad & (\Sigma, \Sigma \cup \{\#\}, S \cup \{s_0', s_f\}, s_0', \{s_f\}, P'), \\ P' = \quad & \{sa \longrightarrow \#s' |\ s, s' \in S, a \in \Sigma, sa \longrightarrow s' \in P\} \cup \\ & \{s_0'a \longrightarrow as_0' |\ a \in \Sigma\} \cup \{s_0'a \longrightarrow \#s |\ s_0a \longrightarrow s \in P\} \cup \\ & \{sa \longrightarrow \#s_f |\ sa \longrightarrow s' \in P, s' \in F\} \cup \{s_fa \longrightarrow as_f |\ a \in \Sigma\} \cup \\ & \{s_0'a \longrightarrow \#s_f |\ s_0a \longrightarrow s \in P, s \in F\}. \end{aligned}$$

It is easy to see that if $h : (\Sigma \cup \{\#\})^* \longrightarrow \Sigma^*$ is the morphism defined by $h(\#) = \lambda$, $h(a) = a$, $\forall\ a \in \Sigma$ then:

$$h(g(L)) = L \longrightarrow \{w\}.$$

If $\lg(w) = n$ then, for every word $\alpha \in g(L)$ the following inequality holds:

$$\lg(\alpha) \leq (n+1)\lg(h(\alpha))$$

which proves that h is an $(n+1)$- linear erasing with respect to $g(L)$.

As CS is closed under λ-free gsm mapping and under linear erasing, it follows that it is closed under sequential deletion with singletons, too. □

Theorem 3.7 *There exist a context-sensitive language L_1 and a word w over an alphabet Σ such that $L_1 \Longrightarrow w$ is not a context-sensitive language.*

Proof. Let L be a recursively enumerable language (which is not context-sensitive) over an alphabet Σ and let a, b be two letters which do not belong to Σ. Then there exists a context-sensitive language L_1 such that (see [12], p.89):

(i) L_1 consists of words of the form $a^ib\alpha$ where $i \geq 0$ and $\alpha \in L$;

(ii) For every $\alpha \in L$, there is an $i \geq 0$ such that $a^ib\alpha \in L_1$.

It is easy to see that:

$$aL_1 \Longrightarrow \{a\} = bL$$

which is not a context-sensitive language. We have concatenated a to the left of L_1 in order to avoid the case $i = 0$, when the corresponding words from L would have been lost. □

Corollary 3.9 *The family of context-sensitive languages is not closed under parallel deletion.*

3.2 Iterated deletion

A natural step following the definition of the sequential and parallel deletion is to consider their iterated versions. The *iterated sequential* and *parallel deletion* have somewhat unexpected properties. While the result of iterated SD from a regular language is regular regardless of the complexity of the deleted language, the families CF and CS are not even closed under iterated SD with singletons. It is an open problem whether REG is closed under iterated PD or iterated PD with singletons.

Definition 3.3 *Let L_1, L_2 be languages over the alphabet Σ. The iterated sequential deletion of order n, $L_1 \longrightarrow^n L_2$, is defined inductively by the equations:*

$$\begin{aligned} L_1 \longrightarrow^0 L_2 &= L_1, \\ L_1 \longrightarrow^{i+1} L_2 &= (L_1 \longrightarrow^i L_2) \longrightarrow L_2,\ i \geq 0. \end{aligned}$$

The iterated sequential deletion (iterated SD) of L_2 from L_1 is then defined as:

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

The *iterated parallel deletion (iterated PD)* of L_2 from L_1 is defined by replacing in the preceding definition the sequential deletion "$\longrightarrow$" with the parallel deletion "$\Longrightarrow$".

Example 3.3 Let $L_1 = \{a^n b^n c^n |\ n \geq 0\}$ and $L_2 = \{ab\}$. Then,

$$L_1 \longrightarrow^* L_2 = \{a^m b^m c^n |\ n, m \geq 0, n \geq m\} = L_1 \Longrightarrow^* L_2.$$

□

However, in general, the results of the iterated SD and iterated PD do not coincide.

Example 3.4 Let $L_1 = \{w \in \{a,b\}^* |\ N_a(w) = N_b(w)\}$ and $L_2 = \{a,b\}$. Then,

$$\begin{aligned} L_1 \longrightarrow^* L_2 &= \{a,b\}^*, \\ L_1 \Longrightarrow^* L_2 &= L_1, \text{ whereas} \\ L_1 \Longrightarrow L_2 &= \{\lambda\}. \end{aligned}$$

□

Given two languages L_1 and L_2 over the alphabet Σ, the following inclusions hold:

$$L_1 \Longrightarrow L_2 \subseteq L_1 \Longrightarrow^* L_2 \subseteq L_1 \longrightarrow^* L_2.$$

Indeed, any parallel deletion can be simulated by a string of sequential deletions. On the other hand, the preceding example shows that the reverse inclusions do not hold.

The iterated sequential and parallel deletion are not commutative operations. For example,

$$ab \longrightarrow^* b = ab \Longrightarrow^* b = \{ab, a\} \neq b \longrightarrow^* ab = b \Longrightarrow^* ab = b.$$

The iterated sequential and parallel deletion are not associative. Take, for example, $L_1 = \{aab\}$, $L_2 = \{b\}$ and $L_3 = \{a\}$. We have:

$$\begin{aligned} (aab \longrightarrow^* b) \longrightarrow^* a &= \{aab, aa\} \longrightarrow^* a = \{aab, aa, ab, b, a, \lambda\}, \\ aab \longrightarrow^* (b \longrightarrow^* a) &= aab \longrightarrow^* b = \{aab, aa\}, \\ \text{and} & \\ (aab \Longrightarrow^* b) \Longrightarrow^* a &= \{aab, aa\} \Longrightarrow^* a = \{aab, aa, b, \lambda\}, \\ aab \Longrightarrow^* (b \Longrightarrow^* a) &= aab \Longrightarrow^* b = \{aab, aa\}. \end{aligned}$$

In general, the sets $L_1 \longrightarrow^* (L_2 \longrightarrow^* L_3)$ and $(L_1 \longrightarrow^* L_2) \longrightarrow^* L_3$ are incomparable and the same can be said about the sets $L_1 \Longrightarrow^* (L_2 \Longrightarrow^* L_3)$ and $(L_1 \Longrightarrow^* L_2) \Longrightarrow^* L_3$. This can be proved by using the preceding and the following examples:

$$ab \longrightarrow^* (bc \longrightarrow^* c) = ab \Longrightarrow^* (bc \Longrightarrow^* c) = \{ab, a\},$$

$$(ab \longrightarrow^* bc) \longrightarrow^* c = (ab \Longrightarrow^* bc) \Longrightarrow^* c = \{ab\}.$$

Let L_1, L_2 be languages over the alphabet Σ. The following lemma shows that the iterated deletion $L_1 \longrightarrow^* L_2$ amounts to the erasing from the words of L_1 of arbitrary numbers of non-overlapping words from $L_2 \longleftarrow^* L_2$.

Lemma 3.2 *Let L_1, L_2 be languages over an alphabet Σ. Then,*

$$L_1 \rightarrow^* L_2 = L_1 \cup \{u_1u_2\dots u_ku_{k+1} |\ k \geq 1, u_i \in \Sigma^*, 1 \leq i \leq k+1, \text{ and } \exists u \in L_1, v_i \in (L_2 \leftarrow^* L_2), 1 \leq i \leq k : u = u_1v_1u_2v_2\dots u_kv_ku_{k+1}\}.$$

Proof. Let us denote by B the right member of the equality.

" $\subseteq$ " We will show by induction on n that $L_1 \rightarrow^n L_2 \subseteq B$.

$n = 0$. $L_1 \rightarrow^0 L_2 \subseteq B$.

$n \mapsto (n+1)$. Assume the statement true for numbers up to n and let α be a word in $L_1 \rightarrow^{n+1} L_2$. There exist $w \in L_2$ and $xwy \in L_1 \rightarrow^n L_2$ such that α equals xy. According to the induction hypothesis, xwy belongs to B.

If xwy is a word in L_1 we are through as $\alpha = xy$ satisfies the properties required by B.

Else, xwy is of the form

$$xwy = u_1u_2\dots u_ku_{k+1}, k \geq 1, u_i \in \Sigma^*, 1 \leq i \leq k+1$$
and
$$\exists u \in L_1, v_i \in (L_2 \leftarrow^* L_2), 1 \leq i \leq k : u = u_1v_1\dots u_kv_ku_{k+1}.$$

Let us analyze the position from where w has been erased. There are two possibilities:

(i) There exists i, $1 \leq i \leq k+1$ such that w has been erased from u_i, $u_i = u_i'wu_i''$. Then α can be written as

$$\alpha = u_1u_2\dots u_{i-1}u_i'u_i''u_{i+1}\dots u_ku_{k+1}$$

where there exist

$$v_1, v_2, \dots, v_{i-1}, w, v_i, \dots v_k \in (L_2 \leftarrow^* L_2), u \in L_1,$$

such that

$$u = u_1v_1u_2v_2\dots u_{i-1}v_{i-1}u_i'wu_i''v_i\dots u_kv_ku_{k+1},$$

which implies $\alpha \in B$.

(ii) The word w is extracted from two or more u_i's. That implies that xwy can be rewritten as:

$$xwy = u_1u_2\dots u_i'u_i''\dots u_j'u_j''\dots u_ku_{k+1},\ i < j,$$
$$u_i = u_i'u_i'', u_j = u_j'u_j'', w = u_i''u_{i+1}\dots u_j',$$

Note that u_{i+p}, $p > 0$, does not occur in case j equals $i + 1$. The word α can be further expressed as:

$$\alpha = u_1 u_2 \dots u_i' u_j'' \dots u_k u_{k+1}.$$

The word $u_i'' v_i u_{i+1} v_{i+1} \dots v_{j-1} u_j' = w'$ is in $(w \leftarrow (L_2 \leftarrow^* L_2)) \subseteq L_2 \leftarrow^* L_2$. (We have used the results from Theorem 2.1.) Therefore we have found now the words $v_1, v_2, \dots v_{i-1}$, $w', v_j, \dots, v_k$ from $L_2 \leftarrow^* L_2$ and $u \in L_1$ such that

$$u = u_1 v_1 u_2 v_2 \dots u_i' w' u_j'' v_j \dots u_k v_k u_{k+1},$$

which means that $\alpha = u_1 u_2 \dots u_i' u_j'' \dots u_k u_{k+1}$ is in B.

" $\supseteq$ " We first prove that for all languages $L_1, L_2, L_3 \subseteq \Sigma^*$ the following relation holds:

$$L_1 \rightarrow (L_2 \leftarrow L_3) \subseteq (L_1 \rightarrow L_3) \rightarrow L_2. \tag{1}$$

Indeed, let α be a word in $L_1 \rightarrow (L_2 \leftarrow L_3)$. There exist words $u \in L_1$ and $v \in L_2 \leftarrow L_3$ such that $u = u_1 v u_2$, $\alpha = u_1 u_2$. As v belongs to $L_2 \leftarrow L_3$ there exists words $w \in L_3$ and $v_1 v_2 \in L_2$ such that $v = v_1 w v_2$. We conclude that u equals $u_1 v_1 w v_2 u_2$. This implies that $u_1 v_1 v_2 u_2 \in u \rightarrow w$ belongs to $L_1 \rightarrow L_3$ and, further, that $\alpha = u_1 u_2 \in u_1 v_1 v_2 u_2 \rightarrow v_1 v_2$ belongs to $(L_1 \rightarrow L_3) \rightarrow L_2$.

Using the relation (1) we can prove that

$$L_1 \rightarrow (L_2 \leftarrow^* L_3) \subseteq (L_1 \rightarrow^* L_3) \rightarrow L_2. \tag{2}$$

Indeed, one can show, by induction on n, that

$$L_1 \rightarrow (L_2 \leftarrow^n L_3) \subseteq (L_1 \rightarrow^n L_3) \rightarrow L_2.$$

For $n = 0$ the inclusion holds as both members equal $L_1 \rightarrow L_2$.

Assume that the inclusion holds for all languages $L_1, L_2, L_3 \subseteq \Sigma^*$ and all numbers up to n. Then,

$$\begin{aligned} L_1 \rightarrow (L_2 \leftarrow^{n+1} L_3) = \quad & L_1 \rightarrow [(L_2 \leftarrow^n L_3) \leftarrow L_3] \subseteq \\ & (L_1 \rightarrow L_3) \rightarrow (L_2 \leftarrow^n L_3) \subseteq \\ & [(L_1 \rightarrow L_3) \rightarrow^n L_3] \rightarrow L_2, \end{aligned}$$

where for the first inclusion we have used the relation (1), and for the second the induction hypothesis with $L_1 \rightarrow L_3$ in the role of L_1.

The relation to be proved holds as we have:

$$(L_1 \longrightarrow L_3) \longrightarrow^n L_3 = L_1 \longrightarrow^{n+1} L_3.$$

Return to the proof of the theorem. If we take $L_3 = L_2$ in (2), we obtain

$$L_1 \longrightarrow (L_2 \longleftarrow^* L_2) \subseteq (L_1 \longrightarrow^* L_2) \longrightarrow L_2 \subseteq L_1 \longrightarrow^* L_2. \tag{3}$$

Next we show that,

$$L_1 \longrightarrow^* (L_2 \longleftarrow^* L_2) \subseteq L_1 \longrightarrow^* L_2. \tag{4}$$

Using induction on k we prove that,

$$L_1 \longrightarrow^k (L_2 \longleftarrow^* L_2) \subseteq L_1 \longrightarrow^* L_2.$$

Indeed, for $k = 0$ the relation holds as $L_1 \subseteq L_1 \longrightarrow^* L_2$. Assume the relation true for k. We have:

$$\begin{aligned} L_1 \longrightarrow^{k+1} (L_2 \longleftarrow^* L_2) = \quad & [L_1 \longrightarrow^k (L_2 \longleftarrow^* L_2)] \longrightarrow (L_2 \longleftarrow^* L_2) \subseteq \\ & (L_1 \longrightarrow^* L_2) \longrightarrow (L_2 \longleftarrow^* L_2) \subseteq \\ & (L_1 \longrightarrow^* L_2) \longrightarrow^* L_2 \subseteq \\ & L_1 \longrightarrow^* L_2, \end{aligned}$$

where for the first inclusion we use the induction hypothesis and for the second one the relation (3) with $L_1 \longrightarrow^* L_2$ in the role of L_1.

From the relation (4) and as we obviously have

$$B \subseteq L_1 \longrightarrow^* (L_2 \longleftarrow^* L_2),$$

we conclude that $B \subseteq L_1 \longrightarrow^* L_2$. □

The lemma helps in proving that if L_1 is regular then $L_1 \longrightarrow^* L_2$ is regular no matter how complex the language L_2 is. Moreover, if L_2 is regular or context-free, the language $L_1 \longrightarrow^* L_2$ can be effectively constructed.

A *finite automaton with λ-transitions* is a finite automaton in which also rules of the type $s \longrightarrow s'$, where s, s' are states, are allowed. If a language L is accepted by a finite automaton with λ-transitions then L is accepted by a finite automaton without λ-transitions (see, for example, [5], pp.24-27).

Theorem 3.8 *Let L_1, L_2 be languages over the alphabet Σ, L_1 a regular one. Then the iterated deletion $L_1 \longrightarrow^* L_2$ is a regular language.*

Proof. Let L_1, L_2 be two languages over Σ and $A = (S, \Sigma, s_0, F, P)$ a finite automaton that recognizes L_1. Let us consider the finite automaton with λ-transitions:

$$\begin{aligned} A' = &\quad (S, \Sigma, s_0, F, P'), \\ P' = &\quad P \cup \{s \longrightarrow s' |\ \exists w \in (L_2 {\leftarrow\!\!\!-}^* L_2) :\ sw \Longrightarrow^* s' \text{ in } A\}. \end{aligned}$$

It will be proved in the following that $L(A') = B$, where B is the set defined in the preceding lemma.

" $\subseteq$ " Let α be a word in $L(A')$ and $s_0\alpha \Longrightarrow^* s_f$, $s_f \in F$, a derivation for α. If no rule from $P' - P$ has been applied during the derivation, then $\alpha \in L_1 \subseteq B$.

Else, assume that k rules from $P' - P$ have been applied. The derivation has the form:

$$\begin{aligned} s_0\alpha = &\quad s_0u_1 \ldots u_ku_{k+1} \Longrightarrow^* s_1u_2 \ldots u_ku_{k+1} \Longrightarrow s'_1u_2 \ldots u_ku_{k+1} \Longrightarrow^* \\ &\quad s_ku_{k+1} \Longrightarrow^* s'_ku_{k+1} \Longrightarrow^* s_f, s_f \in F, \end{aligned}$$

where $u_i \in \Sigma^*, 1 \leq i \leq k+1$.

According to the definition of $P' - P$ the following derivation exists in A, for some words $v_1, \ldots v_k$:

$$\begin{aligned} s_0u = &\quad s_0u_1v_1u_2v_2 \ldots u_kv_ku_{k+1} \Longrightarrow^* s_1v_1u_2v_2 \ldots u_kv_ku_{k+1} \Longrightarrow^* \\ &\quad s'_1u_2v_2 \ldots u_kv_ku_{k+1} \Longrightarrow^* s_kv_ku_{k+1} \Longrightarrow^* s'_ku_{k+1} \Longrightarrow^* s_f. \end{aligned}$$

The existence of this derivation shows that $u \in L_1$ and the definition of $P' - P$ that $v_i \in (L_2 {\leftarrow\!\!\!-}^* L_2)$, $1 \leq i \leq k$. The conditions required by B being satisfied, α belongs to B.

" $\supseteq$ " Let α be a word in B. If $\alpha \in L_1$ then obviously α belongs also to $L(A')$.

Else, assume that

$$\begin{aligned} &\alpha = u_1u_2 \ldots u_ku_{k+1}, k \geq 1, u_i \in \Sigma^*, 1 \leq i \leq k+1 \text{ and} \\ &\exists u \in L_1, v_i \in (L_2 {\leftarrow\!\!\!-}^* L_2), 1 \leq i \leq k : \\ &u = u_1v_1u_2v_2 \ldots u_kv_ku_{k+1}. \end{aligned}$$

As $u \in L_1$, the following derivation exists in A:

$$\begin{aligned} s_0u = &\quad s_0u_1v_1u_2v_2 \ldots u_kv_ku_{k+1} \Longrightarrow^* s_1v_1u_2v_2 \ldots u_kv_ku_{k+1} \Longrightarrow^* \\ &\quad s'_1u_2v_2 \ldots u_kv_ku_{k+1} \Longrightarrow^* s_kv_ku_{k+1} \Longrightarrow^* s'_ku_{k+1} \Longrightarrow^* s_f, s_f \in F. \end{aligned}$$

One can construct now the following derivation in A':

$$\begin{aligned} s_0\alpha = \quad & s_0u_1u_2\dots u_ku_{k+1} \Longrightarrow^* s_1u_2\dots u_ku_{k+1} \Longrightarrow s'_1u_2\dots u_ku_{k+1} \Longrightarrow^* \\ & s_ku_{k+1} \Longrightarrow s'_ku_{k+1} \Longrightarrow^* s_f, s_f \in F. \end{aligned}$$

All the subderivations $s_iv_i \Longrightarrow^* s'_i$ have been replaced with the rules $s_i \longrightarrow s'_i$, as $v_i \in (L_2 \xleftarrow{}{}^* L_2)$, $1 \leq i \leq k$. The existence of this derivation shows that $\alpha \in L(A')$ and the proof of the inclusion is complete.

The theorem now follows because, according to the preceding lemma, the equality $L_1 \rightarrow^* L_2 = B$ holds. □

Corollary 3.10 *The family of regular languages is closed under iterated SD.*

Corollary 3.11 *For any regular language L_1 there exist finitely many languages that can be obtained from L_1 by iterated SD.*

Proof. The claim follows from the preceding theorem by the fact that the automaton A is finite similarily as, for instance, in Corollary 3.5. □

Corollary 3.12 *The proof of the preceding theorem is constructive if L_2 is a regular or a context-free language.*

Proof. The emptiness problem is decidable for REG and CF. Therefore one can decide whether or not the intersection

$$(L_2 \xleftarrow{}{}^* L_2) \cap \{w |\ sw \Longrightarrow^* s'\}$$

is empty, where $s, s' \in S$. Recall that CF is closed under iterated SIN, by Theorem 2.6, and also under intersection with regular languages.

One can construct now the set of rules of the automaton A' as follows. For every two states s, s', the above intersection is formed. If the intersection is not empty, the transition $s \longrightarrow s'$ is added to P', else nothing happens. As S is a finite set, the process terminates. □

Open problem. Is the family of regular languages closed under iterated parallel deletion? What about the case where the language to be deleted is a singleton?

The answer to both of the previous questions is negative in the context-free and context-sensitive case.

Theorem 3.9 *There exist a context-free language L over $\{a, b, \#\}$ and a word w over $\{a, b\}$ such that*

$$L \longrightarrow^* \{w\} \text{ and } L \Longrightarrow^* \{w\}$$

are not context-free languages.

Proof. Let L be the language

$$L = \{a^i \# b^{2i} \mid i > 0\}^*,$$

and $w = ba$.

The following equalities hold:

$$(L \longrightarrow^* ba) \cap a\#^+b^+ = \{a\#^n b^{2^n} \mid n > 0\},$$

$$(L \Longrightarrow^* ba) \cap a\#^+b^+ = \{a\#^n b^{2^n} \mid n > 0\}.$$

Let us prove, for example, the first equality.

" $\subseteq$ " Let u be a word in $(L \longrightarrow^* ba) \cap a\#^+b^+$. There exists a word $v \in L$ such that $u \in v \longrightarrow^* ba$, where

$$v = a^{i_1} \# b^{2i_1} a^{i_2} \# b^{2i_2} \dots a^{i_k} \# b^{2i_k}.$$

As $u \in a\#^+b^+$, everything from between the markers has been erased from v and $i_1 = 1$. This implies

$$2i_1 = i_2, 2i_2 = i_3, \dots, 2i_{k-1} = i_k.$$

Taking into account that $i_1 = 1$, one deduces that $u = a\#^k b^{2^k}$, $k > 0$.

" $\supseteq$ " Let $u = a\#^k b^{2^k}$, $k > 0$. There exists $v \in L$,

$$v = a\#b^2a^2\#b^4 \dots a^{2^{k-1}} \# b^{2^k},$$

such that $u \in (v \longrightarrow^* ba) \subseteq (L \longrightarrow^* ba)$. As u belongs also to $a\#^+b^+$, the proof of the second inclusion is completed.

The theorem follows because the language $\{a\#^n b^{2^n} \mid n > 0\}$ is not a context- free language. □

Corollary 3.13 *The family of context-free languages is closed under neither iterated SD nor iterated PD.*

Theorem 3.10 *Let Σ be an alphabet and a, b letters which do not occur in Σ. There exist a context-sensitive language L_1 over $\Sigma \cup \{a, b\}$ and a word w over $\{a, b\}$ such that*

$$L_1 \longrightarrow^* w \text{ and } L_1 \Longrightarrow^* w$$

are not context-sensitive languages.

Proof. Let L be the recursively enumerable (which is not context- sensitive) language and L_1 the context-sensitive language defined in Theorem 3.7.

It is obvious that

$$(L_1 \longrightarrow^* \{a\}) \cap b\Sigma^* = bL,$$

$$(L_1 \Longrightarrow^* \{a\}) \cap b\Sigma^* = bL,$$

and bL is not a context-sensitive language. □

Corollary 3.14 *The family of context-sensitive languages is closed under neither iterated SD nor iterated PD.*

3.3 Permuted deletion

In this section the permuted variants of the sequential and parallel deletion will be investigated. The *permuted SD* of the word v from the word u, $(u \rightsquigarrow v)$, is the set obtained by erasing from u arbitrary occurrences (but one at a time in the sequential case) of words which are letter-equivalent to v. The *permuted PD*, $(u \Rightsquigarrow v)$, is the set obtained by erasing from u all the non-overlapping occurrences of words which are letter-equivalent to v. If none of the words letter-equivalent to v is a subword of u, the result of the permuted SD, as well as of the permuted PD is the empty set.

Definition 3.4 *Let L_1, L_2 be two languages over the alphabet Σ. The permuted sequential deletion (shortly, permuted SD) of L_2 from L_1 is defined as:*

$$L_1 \rightsquigarrow L_2 = \bigcup_{u \in L_1, v \in L_2} (u \rightsquigarrow v)$$

where $u \rightsquigarrow v = u \longrightarrow com(v)$.

Example 3.5 Let L_1, L_2 be the languages:

$$\begin{aligned} L_1 = &\quad \{a^3b^3a^3, ba^2, b^2a\}, \\ L_2 = &\quad \{aba\}. \end{aligned}$$

The permuted sequential deletion of L_2 from L_1 is:

$$L_1 \rightsquigarrow L_2 = L_1 \rightsquigarrow \{aab\} = L_1 \rightsquigarrow \{baa\} = \{ab^2a^3, a^3b^2a, \lambda\},$$

being the union of the sets:

$$\begin{aligned} a^3b^3a^3 \rightsquigarrow aba &= \{ab^2a^3, a^3b^2a\}, \\ ba^2 \rightsquigarrow aba &= \{\lambda\}, \\ b^2a \rightsquigarrow aba &= \emptyset. \end{aligned}$$

□

Replacing in the previous definition the sequential deletion $\longrightarrow$ with the parallel deletion, $\Longrightarrow$, one obtains the definition for the *permuted parallel deletion*.

Example 3.6 Let L_1, L_2 be the languages:

$$\begin{aligned} L_1 = &\quad \{abababa, ba^3b, ba^3baba\}, \\ L_2 = &\quad \{aba\}. \end{aligned}$$

The permuted parallel deletion of L_2 from L_1 is:

$$L_1 \approx\!\!\rightsquigarrow L_2 = \{b, abba, ab, ba\},$$

being the union of the sets:

$$\begin{aligned} abababa \approx\!\!\rightsquigarrow aba &= \{b, abba\}, \\ ba^3b \approx\!\!\rightsquigarrow aba &= \{ab, ba\}, \\ ba^3baba \approx\!\!\rightsquigarrow aba &= \{ab, ba\}. \end{aligned}$$

□

It becomes obvious from the definition that, if one replaces the language to be deleted by a letter-equivalent one, the result of the permuted SD, as well as of the permuted PD does not change.

The permuted sequential deletion is not commutative. For example, $ab \rightsquigarrow b = a$, whereas $b \rightsquigarrow ab = \emptyset$. The same languages can be used to prove that the permuted parallel deletion is not commutative.

The permuted sequential and parallel deletion are not associative. In general, the sets $L_1 \rightsquigarrow (L_2 \rightsquigarrow L_3)$ and $(L_1 \rightsquigarrow L_2) \rightsquigarrow L_3$ are incomparable and a similar statement holds in the parallel case. Indeed, this is proved by the following examples:

$$ab \rightsquigarrow (bc \rightsquigarrow c) = ab \approx (bc \approx c) = a \text{ while}$$
$$(ab \rightsquigarrow bc) \rightsquigarrow c = (ab \approx bc) \approx c = \emptyset,$$

$$(ab \rightsquigarrow b) \rightsquigarrow a = (ab \approx b) \approx a = \lambda \text{ while}$$
$$ab \rightsquigarrow (b \rightsquigarrow a) = ab \approx (b \approx a) = \emptyset.$$

In the sequel, the closure properties of the families in the Chomsky hierarchy under permuted SD and PD will be investigated.

Theorem 3.11 *The family of regular languages is closed under permuted sequential deletion.*

Proof. The claim follows from Definition 3.4 and from Lemma 3.1. □

Note that, unlike all other closure results presented in Chapters 2, 3, the above one is not effective. Indeed, Lemma 3.1 states that the result of the sequential deletion from a regular language is always regular. However, Corollary 3.4 emphasises that the result of the sequential deletion from a regular language can be effectivelly constructed only in case the language to be sequentially deleted is regular or context-free. As we have seen in Example 2.10, there exist regular languages whose commutative closure is not context-free. Consequently, Corollary 3.4 cannot be applied in the case of permuted SD. In the particular case where the language to be (permuted sequentially) deleted is a singleton, Corollary 3.4 is applicable. Therefore the proof of the closure of REG under permuted sequential deletion with singletons is effective.

In the parallel case one obtains the following non-closure result.

Theorem 3.12 *The family of regular languages is not closed under permuted parallel deletion.*

Proof. Let L_1, L_2 be the regular languages:

$$\begin{aligned} L_1 = &\quad \$a^*b^*\#\#a^*b^*\$, \\ L_2 = &\quad \#\$(ab)^*. \end{aligned}$$

Then the permuted PD of L_2 from L_1 is:

$$L_1 \approx L_2 = \{\$a^n b^m \#|\ m,n \geq 0, m \neq n\} \cup \{\#a^n b^m \$|\ m,n \geq 0, m \neq n\} \cup \{\lambda\}.$$

Indeed, let $u = \$a^n b^m \# \# a^p b^q \$$ be a word in L_1 and w a word in $com(L_2)$.

Because of the presence of the markers, the result of the permuted PD

$$\$a^n b^m \# \# a^p b^q \$ \approx w,\ w \in com(L_2)$$

is not empty iff

$$w = \$a^r b^r \#, r \geq 0 \text{ and } m = n = r,$$

or

$$w = \#a^s b^s \$, s \geq 0 \text{ and } p = q = s.$$

The situations being similar, let us assume that the first case holds.

Then, if $p = q$, the result of the operation will be $\{\lambda\}$, as the word $\#a^p b^p \$ \in com(L_2)$ will be deleted in parallel with w from $u \in L_1$. In order to obtain a nonempty word, the condition $p \neq q$ must be satisfied.

In the second case, reasoning similarily, one deduces that the condition $m \neq n$ must be fulfilled in order to get a nonempty word in the result of the deletion.

It has therefore been shown that the words v in the language $L_1 \approx L_2$ have one of the following forms:

$$\begin{aligned} v &= \$a^n b^m \#, n,m \geq 0, n \neq m, \\ v &= \#a^p b^q \$, p,q \geq 0, p \neq q, \\ v &= \lambda. \end{aligned}$$

As words of this form can be obtained in $L_1 \approx L_2$ for any numbers $n,m,p,q \geq 0$, the equality is proved.

The theorem now follows because the language

$$\{\$a^n b^m \#|\ n,m \geq 0, n \neq m\} \cup \{\#a^n b^m \$|\ n,m \geq 0, n \neq m\} \cup \{\lambda\}$$

is not a regular one. □

The family of context-free languages is not closed even under permuted SD with regular languages as shown below.

Theorem 3.13 *The family of context-free languages is closed under neither permuted sequential nor permuted parallel deletion with regular languages.*

Proof. Let L_1 be the context-free language:

$$L_1 = \{a_1^n b_1^m c_1^l \# c_2^l b_2^m a_2^n \# |\ n, m, l \geq 0\}$$

and L_2 the regular language:

$$L_2 = \#\#(a_2 b_2 c_2)^*.$$

Then the permuted SD of L_2 from L_1 is:

$$L_1 \rightsquigarrow L_2 = L_1 \longrightarrow com(L_2) = \{a_1^n b_1^n c_1^n |\ n \geq 0\}.$$

Indeed, let $u = a_1^n b_1^m c_1^l \# c_2^l b_2^m a_2^n \#$ and $w \in com(L_2)$. The set

$$a_1^n b_1^m c_1^l \# c_2^l b_2^m a_2^n \# \longrightarrow w$$

is not empty iff $w = \#c_2^r b_2^r a_2^r \#$ and $m = n = l = r$. This, in turn, implies that the only word in $u \rightsquigarrow w$ is $a_1^r b_1^r c_1^r$.

As such a word can be obtained in $L_1 \longrightarrow com(L_2)$ for every $r \geq 0$, the requested equality follows.

Because of the presence of the markers, the permuted SD and PD coincide,

$$L_1 \rightsquigarrow L_2 = L_1 \rightsquigarrow\!\!\!\!\!\!\sim L_2.$$

The theorem now follows as the language $\{a_1^n b_1^n c_1^n |\ n \geq 0\}$ is not a context-free one. □

Corollary 3.15 *The family of context-free languages is closed under neither permuted SD nor permuted PD.*

In the particular case when the language to be deleted is a singleton, the permuted SD and PD preserve the families of regular and context-free languages.

Theorem 3.14 *The family of regular and the family of context-free languages are closed under permuted SD and permuted PD with singletons.*

Proof. Let L be a regular (respectively context-free) language and w a word over the same alphabet Σ. Then,

$$L \rightsquigarrow \{w\} = L \longrightarrow com(w),$$

$$L \approx\!\!\rightsquigarrow \{w\} = L \Longrightarrow com(w).$$

As the set $com(w)$ is finite and the families of regular and context-free languages are closed under SD and PD with regular languages (see Corollaries 3.3, 3.6, 3.7), the theorem is proved. □

The family of context-sensitive languages will not be closed under permuted SD and permuted PD as it is not closed under permuted SD with regular languages and under permuted PD with singletons.

Theorem 3.15 *There exist a context-sensitive language L_1 over the alphabet $\Sigma \cup \{a, b\}$ and a regular language R over $\{a, b\}$ such that $L_1 \rightsquigarrow R$ is not context-sensitive.*

Proof. Let L be a recursively enumerable (which is not context-sensitive) language over Σ and L_1 the context-sensitive language over $\Sigma \cup \{a, b\}$, defined in Theorem 3.7.

It is easy to see that

$$(L_1 \rightsquigarrow a^*b) \cap \Sigma^* = L,$$

which implies that $L_1 \rightsquigarrow a^*b$ is not context-sensitive. □

Corollary 3.16 *The family of context-sensitive languages is not closed under permuted SD.*

Theorem 3.16 *The family of context-sensitive languages is closed under permuted SD with singletons.*

Proof. Let L be a language and w be a word over the same alphabet Σ. Then,

$$L \rightsquigarrow \{w\} = \bigcup_{u \in com(w)} (L \longrightarrow \{u\}).$$

As the family of context-sensitive languages is closed under SD with singletons (see Theorem 3.6) and under finite union, it is closed under permuted SD with singletons too. □

Theorem 3.17 *There exists a context-sensitive language L_1 over $\Sigma \cup \{a, b\}$ and a word $w = a$ such that $L_1 \approx w$ is not a context-sensitive language.*

Proof. As $w = a$, the permuted PD amounts to ordinary PD and the same proof as for Theorem 3.7 holds. □

Corollary 3.17 *The family of context-sensitive languages is not closed under permuted PD.*

3.4 Controlled deletion

In all the previous variants of deletion no restriction was made concerning the position where the deletion was performed. One can apply also for deletion the notion of control introduced in Section 2.4 for insertion: every letter determines what can be deleted after it.

Definition 3.5 *Let L be a language over the alphabet Σ. For each letter a of the alphabet, let $\Delta(a)$ be a language over Σ. The Δ-controlled sequential deletion from L (shortly, controlled SD) is defined as:*

$$L \rightarrowtail \Delta = \bigcup_{u \in L} (u \rightarrowtail \Delta),$$

where

$$u \rightarrowtail \Delta = \{u_1 a u_2 \in \Sigma^* \mid u = u_1 a v u_2 \text{ for some } u_1, u_2 \in \Sigma^*, a \in \Sigma \text{ and } v \in \Delta(a)\}.$$

The function $\Delta : \Sigma \longrightarrow 2^{\Sigma^*}$ is called a control function.

As a language operation, the Δ-controlled SD has the arity $\mathrm{card}(\Sigma) + 1$.

If one imposes the restriction that for a distinguished $b \in \Sigma$, $\Delta(b) = L_2$, and $\Delta(a) = \emptyset$ for any letter $a \neq b$, a special case of controlled SD is obtained: the *sequential deletion next to the letter b*, denoted by $L \overset{b}{\rightarrowtail} L_2$. The SD next to a letter is a binary operation. The words in $L \overset{b}{\rightarrowtail} L_2$ are obtained by erasing from words in L one occurrence of a word of L_2 which appears immediately next to a letter b. The words from L which do not contain the letter b followed by a word from L_2 do not contribute to the result.

Example 3.7 Let L be the language $L = \{abba, aab, bba, aabb\}$ and Δ the control function $\Delta(a) = b$, $\Delta(b) = a$. Then we have:

$$\begin{aligned} L \rightarrowtail \Delta &= \{aba, abb, aa, bb, aab\}, \\ L \overset{a}{\rightarrowtail} \{b\} &= \{aba, aa, aab\}, \\ L \overset{b}{\rightarrowtail} \{a\} &= \{abb, bb\}. \end{aligned}$$

□

In general, if L is a language over Σ and $\Delta : \Sigma \longrightarrow 2^{\Sigma^*}$ a control function,

$$L \rightarrowtail \Delta = \bigcup_{a \in \Sigma} (L \overset{a}{\rightarrowtail} \Delta(a)) = \bigcup_{u \in L} \bigcup_{a \in \Sigma} (u \overset{a}{\rightarrowtail} \Delta(a)).$$

The sequential deletion $L_1 \rightarrow L_2$ can be expressed in terms of controlled SD by using a control function which has the value L_2 for all letters in Σ and a marker. Indeed,

$$L_1 \rightarrow L_2 = h(\#L_1 \rightarrowtail \Delta),$$

where $\Delta(\#) = \Delta(a) = L_2, \forall a \in \Sigma$ and h is the morphism that erases the marker $\#$.

The left quotient can be obtained from the SD next to a letter by using a marker and the morphism h which erases the marker:

$$L_2 \backslash L_1 = h(\#L_1 \overset{\#}{\rightarrowtail} L_2).$$

Notice that if the letter a does not occur in the word u then $u \overset{a}{\rightarrowtail} \Delta(a) = \emptyset$. This happens also if a occurs in u but no word of the form av, $v \in \Delta(a)$ exists in u. In particular, if λ belongs to L, λ does not contribute to the result of the controlled SD:

$$L \rightarrowtail \Delta = (L - \{\lambda\}) \rightarrowtail \Delta, \forall\ L \subseteq \Sigma^*, \Delta : \Sigma \longrightarrow 2^{\Sigma^*}.$$

The kind of control that has been defined above is a *right* control: a letter determines what may be deleted from its right. A *left* Δ*-controlled* SD from L, denoted $L \rightarrowtail\!\!\!\!\!\rightarrow \Delta$, can be analogously defined by replacing in Definition 3.5 "u_1avu_2" by "u_1vau_2". The *left-SD next to a letter* can be defined in a similar way.

Example 3.8 If we consider the language and the function from Example 3.7 then:

$$\begin{aligned} L \rightarrowtail \Delta &= \{aba, bba, ab, ba, abb\}, \\ L \overset{a}{\rightarrowtail} \{b\} &= \{aba, ba\}, \\ L \overset{b}{\rightarrowtail} \{a\} &= \{bba, ab, abb\}. \end{aligned}$$

□

The right quotient can be obtained from the left-SD next to a letter in the following way.

$$L_1/L_2 = h(L_1\# \overset{\#}{\rightarrowtail} L_2),$$

where h is the morphism erasing the marker $\#$.

The left controlled SD is similar to the right controlled SD as we have:

Theorem 3.18 *Let L be a language over Σ and $\Delta : \Sigma \longrightarrow 2^{\Sigma^*}$ be a control function. Then*

$$L \rightarrowtail \Delta = Mi(L' \rightarrowtail \Delta'),$$

where $L' = Mi(L)$ and, for every $a \in \Sigma$, $\Delta'(a) = Mi(\Delta(a))$.

Proof. " $\subseteq$ " Let w be a word in $L \rightarrowtail \Delta$. Then $w = u_1 a u_2$ where $u = u_1 a v u_2 \in L$, $v \in \Delta(a)$.

As $\mathrm{Mi}(v)$ belongs to $\Delta'(a)$ we have:

$$\begin{aligned} \mathrm{Mi}(w) = \; & \mathrm{Mi}(u_2)a\mathrm{Mi}(u_1) \in \mathrm{Mi}(u_2)\mathrm{Mi}(v)a\mathrm{Mi}(u_1) \rightarrowtail \Delta' = \\ & \mathrm{Mi}(u_1 a v u_2) \rightarrowtail \Delta' = \mathrm{Mi}(u) \rightarrowtail \Delta' \subseteq L' \rightarrowtail \Delta', \end{aligned}$$

which implies that $w \in \mathrm{Mi}(L' \rightarrowtail \Delta')$.

" $\supseteq$ " Conversely, if $w \in \mathrm{Mi}(L' \rightarrowtail \Delta')$ then $\mathrm{Mi}(w) \in L' \rightarrowtail \Delta'$, that is:

$$\begin{aligned} & \mathrm{Mi}(w) = u_1 a u_2, \\ & v \in \Delta'(a) = \mathrm{Mi}(\Delta(a)), \\ & u_1 v a u_2 \in \mathrm{Mi}(L). \end{aligned}$$

As $\mathrm{Mi}(v)$ belongs to $\Delta(a)$ we have:

$$\begin{aligned} w = \; & \mathrm{Mi}(u_1 a u_2) \in \mathrm{Mi}(u_2)a\mathrm{Mi}(v)\mathrm{Mi}(u_1) \rightarrowtail \Delta \; = \\ & \mathrm{Mi}(u_1 v a u_2) \rightarrowtail \Delta \; \subseteq L \rightarrowtail \Delta \; , \end{aligned}$$

and the second inclusion is proved. □

A parallel variant of the controlled deletion will be defined in the sequel. Let $u \in \Sigma^*$ be a word and $\Delta : \Sigma \longrightarrow 2^{\Sigma^*}$ be a control function which does not have $\emptyset$ as its value. The set $u \models\!\!\!\Rightarrow \Delta$ is obtained by finding all the nonoverlapping occurrences of av_a, $v_a \in \Delta(a)$, in u, and by deleting v_a from them. Between any two occurrences of words of the type av_a, $v_a \in \Delta(a)$, in u, no other words of this type may remain.

Definition 3.6 *Let L be a language over an alphabet Σ and $\Delta : \Sigma \longrightarrow 2^{\Sigma^*}$ be a control function such that $\Delta(a) \neq \emptyset$, $\forall a \in \Sigma$. The Δ- controlled parallel deletion from L (shortly, controlled PD) is defined as:*

$$L \models\!\!\!\Rightarrow \Delta = \bigcup\nolimits_{u \in L} (u \models\!\!\!\Rightarrow \Delta),$$

where

$$\begin{aligned} u \models\!\!\!\Rightarrow \Delta = \quad & \{u_1 a_1 u_2 a_2 \dots u_k a_k u_{k+1} |\ k \geq 1, a_j \in \Sigma, 1 \leq j \leq k, \\ & u_i \in \Sigma^*, 1 \leq i \leq k+1, \text{ and there exist } v_i \in \Delta(a_i), 1 \leq i \leq k, \\ & \text{such that } u = u_1 a_1 v_1 \dots u_k a_k v_k u_{k+1}, \text{where} \\ & \{u_i\} \cap \Sigma^* (\cup_{a \in \Sigma} a \Delta(a)) \Sigma^* = \emptyset,\ 1 \leq i \leq k+1.\} \end{aligned}$$

The last line is a formalization of the condition that no word av_a, $v_a \in \Delta(a)$, may occur in u between $a_i v_i$, $1 \leq i \leq k$, $v_i \in \Delta(a_i)$.

The arity of the Δ-controlled parallel deletion is $\mathrm{card}(\Sigma) + 1$. A binary variant of it is defined in the sequel.

If one imposes the restriction that for a distinguished letter $b \in \Sigma$ we have $\Delta(b) = L_2$, and $\Delta(a) = \lambda$ for any letter $a \neq b$, a special case of controlled PD is obtained: *parallel deletion next to the letter b.* The parallel deletion next to b is denoted by $\overset{b}{\models\!\!\!\Rightarrow}$. Let us examine the set $u \overset{b}{\models\!\!\!\Rightarrow} L_2$, where u is a nonempty word and L_2 is a language over an alphabet Σ. If $u = b^k$, $k > 0$, and no word of the form bv, $v \in L_2$ occurs as a subword in u, the set $u \overset{b}{\models\!\!\!\Rightarrow} L_2$ equals the empty set. If u contains at least one letter different from b, u is retained in the result as we can erase λ near that letter. The other words in $u \overset{b}{\models\!\!\!\Rightarrow} L_2$ are obtained by finding all the nonoverlapping occurrences of words of the type bv_i, $v_i \in L_2$, in u, and deleting v_i from them. There may exist more than one possibility of finding such a decomposition of u into subwords.

Example 3.9 Let $L = \{abababa, a^3b^3, abab\}$ and $\Delta(a) = b$, $\Delta(b) = a$. Then:

$$L \models\!\!\!\Rightarrow \Delta = \{a^4, ab^3, a^2b^2, ab^2a^2, a^3b, a^3b^2, a^2, ab^2\},$$

being the union of the sets:

$$\begin{aligned} abababa \models\!\!\Rightarrow \Delta &= \{a^4, ab^3, a^2b^2, ab^2a^2, a^3b\}, \\ a^3b^3 \models\!\!\Rightarrow \Delta &= \{a^3b^2\}, \\ abab \models\!\!\Rightarrow \Delta &= \{a^2, ab^2\} \end{aligned}$$

whereas

$$\begin{aligned} L \overset{a}{\models\!\!\Rightarrow} \{b\} &= \{a^4, a^3ba, a^2ba^2, aba^3, a^2baba, \\ &\quad aba^2ba, ababa^2, abababa, a^3b^2, \\ &\quad a^3b^3, a^2, a^2b, aba, abab\}, \\ L \overset{b}{\models\!\!\Rightarrow} \{a\} &= \{ab^3, ab^3a, ab^2ab, abab^2, ab^2aba, abab^2a, \\ &\quad ababab, abababa, a^3b^3, ab^2, abab\}. \end{aligned}$$

□

As in the sequential case, if the empty word belongs to L, this does not influence the result of the controlled PD:

$$L \models\!\!\Rightarrow \Delta = (L - \{\lambda\}) \models\!\!\Rightarrow \Delta, \ \forall L \subseteq \Sigma^*, \Delta : \Sigma \longrightarrow 2^{\Sigma^*}, \Delta(a) \neq \emptyset, \forall a \in \Sigma.$$

The *left* Δ*-controlled PD* from L, denoted $L \Rightarrow\!\!\!\!\succ \Delta$, can be defined by replacing in Definition 3.6

$$\text{"}u = u_1a_1v_1 \ldots u_ka_kv_ku_{k+1}\text{"} \text{ with } \text{"}u = u_1v_1a_1 \ldots u_kv_ka_ku_{k+1}\text{"}$$

and

$$\text{"}\{u_i\} \cap \Sigma^*(\cup_{a\in\Sigma} a\Delta(a))\Sigma^* = \emptyset\text{"} \text{ with } \text{"}\{u_i\} \cap \Sigma^*(\cup_{a\in\Sigma} \Delta(a)a)\Sigma^* = \emptyset\text{"}.$$

The *left PD next to a letter* can be defined in a similar way.

Example 3.10 Let L be the language and Δ the control function defined in Example 3.9. Then we have:

$$L \Rightarrow\!\!\!\!\succ \Delta = \{a^4, b^2a^2, ba^3, b^3a, a^2b^2a, a^2b^3, b^2, a^2b\},$$

being the union of the sets

$$\begin{aligned} abababa \Rightarrow\!\!\!\!\succ \Delta &= \{a^4, b^2a^2, b^3a, ba^3, a^2b^2a\}, \\ a^3b^3 \Rightarrow\!\!\!\!\succ \Delta &= \{a^2b^3\}, \\ abab \Rightarrow\!\!\!\!\succ \Delta &= \{b^2, a^2b\}, \end{aligned}$$

whereas

$$\begin{aligned} L \overset{a}{\rightarrowtail} \{b\} &= \{a^4, a^2baba, aba^2ba, ababa^2, a^3ba, aba^3, \\ &\quad a^2ba^2, abababa, a^3b^3, a^2b, abab\}, \\ L \overset{b}{\rightarrowtail} \{a\} &= \{b^3a, b^2aba, bab^2a, ab^3a, bababa, \\ &\quad ab^2aba, abab^2a, abababa, a^2b^3, \\ &\quad a^3b^3, b^2, bab, ab^2, abab\}. \end{aligned}$$

□

The left controlled PD proves to be similar to the right controlled PD as shown below.

Theorem 3.19 *Let L be a language over Σ and $\Delta : \Sigma \longrightarrow 2^{\Sigma^*}$ be a control function, $\Delta(a) \neq \emptyset$, $\forall a \in \Sigma$. Then,*

$$L \rightarrowtail \Delta = Mi(L' \rightarrowtail \Delta'),$$

where $L' = Mi(L)$ and, for every $a \in \Sigma$, $\Delta'(a) = Mi(\Delta(a))$.

Proof. Similar to the sequential case proved already in this section. □

In the general case of controlled SD and controlled PD, we cannot speak about commutativity and associativity. However, in the case of SD next to a letter and PD next to a letter, these notions can be defined and studied.

Obviously, the SD next to a letter and PD next to a letter are not commutative operations. For example, $(ab \overset{a}{\mapsto} b) = a$, $(b \overset{a}{\mapsto} ab) = \emptyset$ and $(ab \overset{a}{\Mapsto} b) = \{a, ab\}$, while $(b \overset{a}{\Mapsto} ab) = b$.

The SD next to a letter and PD next to a letter are not associative operations either. In general, the sets $L_1 \overset{a}{\mapsto} (L_2 \overset{b}{\mapsto} L_3)$ and $(L_1 \overset{a}{\mapsto} L_2) \overset{b}{\mapsto} L_3$ are incomparable, and the same can be said about the sets $L_1 \overset{a}{\Mapsto} (L_2 \overset{b}{\Mapsto} L_3)$ and $(L_1 \overset{a}{\Mapsto} L_2) \overset{b}{\Mapsto} L_3$. This is proved by the following examples:

$$\begin{array}{rlrl} bb \overset{b}{\mapsto} (ba \overset{b}{\mapsto} a) = b & \text{whereas} & & (bb \overset{b}{\mapsto} ba) \overset{b}{\mapsto} a = \emptyset, \\ (aab \overset{a}{\mapsto} b) \overset{a}{\mapsto} a = a & \text{whereas} & & aab \overset{a}{\mapsto} (b \overset{a}{\mapsto} a) = \emptyset, \\ & & \text{and} & \\ (aab \overset{a}{\Mapsto} b) \overset{a}{\Mapsto} a = \{a, ab\} & \text{whereas} & aab \overset{a}{\Mapsto} (b \overset{a}{\Mapsto} a) = \{aa, aab\}. & \end{array}$$

If the control function has as values regular languages for every letter of the alphabet, the controlled SD can be simulated by a gsm with erasing.

Theorem 3.20 *Let L be a language over Σ and $\Delta : \Sigma \longrightarrow 2^{\Sigma^*}$ a control function whose values are regular languages. There exists a gsm g such that:*

$$L \longmapsto \Delta \;=\; g(L).$$

Proof. According to a previous remark, one can assume that L is λ-free. For every $a \in \Sigma$, let $A_a = (S_a, \Sigma, s_a, F_a, P_a)$ be a finite automaton that accepts the language $\Delta(a)$. Assume further that all the state sets S_a, $a \in \Sigma$, are pairwise disjoint.

Consider the gsm with erasing:

$$\begin{array}{ll} g = & (\Sigma, \Sigma, S, s_0, \{s_f\}, P), \\ S = & (\cup_{a\in\Sigma} S_a) \cup \{s_0, s_f\}, \\ P = & (\cup_{a\in\Sigma} P_a) \cup \{s_0 a \longrightarrow a s_0 |\ a \in \Sigma\} \cup \\ & \{s_0 a \longrightarrow a s_a |\ a \in \Sigma\} \cup \{s_f a \longrightarrow a s_f |\ a \in \Sigma\} \cup \\ & \{sb \longrightarrow s_f |\ b \in \Sigma \text{ and } \exists a \in \Sigma, s' \in F_a : sb \longrightarrow s' \in P_a\} \cup \\ & \{s_0 a \longrightarrow a s_f |\ a \in \Sigma, \lambda \in \Delta(a)\}, \end{array}$$

where s_0, s_f are new symbols of states. The construction of g and the proof that $L \longmapsto \Delta \;=\; g(L)$ are similar to that of Theorem 3.2. The only difference is that, in the controlled case, words from $\Delta(a)$ are erased only if they occur after the letter a. □

Corollary 3.18 *The family of regular and the family of context-free languages are closed under controlled SD with regular languages.*

Proof. The claim follows from the preceding theorem as REG and CF are closed under gsm mapping. □

The family of context-free languages is closed under neither controlled SD nor controlled PD as it is not closed under SD and PD next to one letter.

Theorem 3.21 *There exist two context-free languages L_1, L_2 over an alphabet Σ and a letter $\#$ in Σ such that $L_1 \overset{\#}{\longmapsto} L_2$ and $L_1 \overset{\#}{\Longmapsto} L_2$ are not context-free languages.*

Proof. Let $\Sigma = \{a, b, \#\}$ and L_1, L_2 be the context-free languages:

$$\begin{array}{ll} L_1 = & \#\{a^i b^{2i} |\ i > 0\}^*, \\ L_2 = & a\{b^i a^i |\ i > 0\}^*. \end{array}$$

Then,

$$(L_1 \overset{\#}{\longmapsto} L_2) \cap \#b^+ = \{\#b^{2^n} \mid n > 0\},$$

which is not a context-free language.

Because of the presence of the marker, in this case

$$(L_1 \overset{\#}{\longmapsto} L_2) \cap \#b^+ = (L_1 \overset{\#}{\models} L_2) \cap \#b^+.$$

□

Corollary 3.19 *The family of context-free languages is closed under neither controlled sequential nor controlled parallel deletion.*

If the control function has as values only nonempty regular languages then the controlled PD, $L \models \Delta$, can be expressed as a morphic image of an intersection between a regular language and the image of L through a gsm with erasing.

Theorem 3.22 *Let L be a language over Σ and $\Delta : \Sigma \longrightarrow 2^{\Sigma^*}$ a control function whose values are nonempty regular languages. There exist a gsm g, a morphism h and a regular language R' such that:*

$$L \models \Delta = h(g(L) \cap R').$$

Proof. We can assume, without loss of generality, that L is a λ-free language. For every letter $a \in \Sigma$, let $A_a = (S_a, \Sigma, s_a, F_a, P_a)$ be a finite automaton that accepts the language $\Delta(a)$. Assume that all the state sets S_a, $a \in \Sigma$, are pairwise disjoint.

Consider the gsm with erasing:

$$\begin{array}{rl} g = & (\Sigma, \Sigma \cup \{\#\}, S, s_0, F, P), \\ \text{where} & \\ S = & (\cup_{a\in\Sigma} S_a) \cup \{s_0\}, \\ F = & \{s_0\}, \\ P = & (\cup_{a\in\Sigma} P_a) \cup \{s_0 a \longrightarrow a s_0 \mid a \in \Sigma\} \cup \\ & \{s_0 a \longrightarrow a s_a \mid a \in \Sigma\} \cup \{s_0 a \longrightarrow a \# s_0 \mid a \in \Sigma, \lambda \in \Delta(a)\} \cup \\ & \{sb \longrightarrow \# s_0 \mid b \in \Sigma, s \in S_a, s' \in F_a \text{ and } sb \longrightarrow s' \in P_a\}, \end{array}$$

where $s_0, \#$ are symbols which do not occur in any of the given sets. The construction is similar to that of Theorem 3.3. The only difference is that

the input letters determine to which automaton the derivation is switched, that is, words of which language are to be erased.

One can prove as in Theorem 3.3 that:

$$\begin{aligned} g(L) = \quad & L \cup \{u_1a_1\#u_2a_2\#\dots u_ka_k\#u_{k+1} |\ k \geq 1, \\ & a_j \in \Sigma, 1 \leq j \leq k, u_i \in \Sigma^*, 1 \leq i \leq k+1 \\ & \text{ and } \exists u \in L, v_i \in \Delta(a_i), 1 \leq i \leq k : \\ & u = u_1a_1v_1u_2a_2v_2 \dots u_ka_kv_ku_{k+1}\}. \end{aligned}$$

Considering now the morphism $h : (\Sigma \cup \{\#\})^* \longrightarrow \Sigma^*$, $h(\#) = \lambda$, $h(a) = a$, $\forall a \in \Sigma$ and the regular language

$$R' = [(\Sigma \cup \Sigma\#)^*(\cup_{a\in\Sigma} a\Delta(a))(\Sigma \cup \Sigma\#)^*]^c \cap (\Sigma^*\#\Sigma^*)^+,$$

the equality

$$L \models\!\!\Rightarrow \Delta \ = \ h(g(L) \cap R')$$

holds. Indeed, the first set of the intersection takes care that between two markers (representing erased words), no other candidate for erasing occurs. The second set assures that words from L which contain no candidate for erasing are excluded. This is done by retaining only the words in which at least one marker (erasing) occurrs.

Corollary 3.20 *The family of regular and the family of context-free languages are closed under controlled parallel deletion with regular languages.*

Proof. The claim follows from the preceding theorem as REG and CF are closed under intersection with regular languages, gsm mapping and morphism. □

The family of context-sensitive languages is not closed under controlled SD and controlled PD. However, in the particular case when the control function has as values only singletons, CS is closed under these operations.

Theorem 3.23 *Let Σ be an alphabet and $a, b, \#$ symbols which do not belong to Σ. There exists a context-sensitive language L_1' over the alphabet $\Sigma \cup \{a, b, \#\}$ and a regular language R over $\{a, b\}^*$ such that $L_1' \overset{\#}{\longmapsto} R$ and $L_1' \overset{\#}{\models\!\!\Rightarrow} R$ are not context-sensitive languages.*

Proof. Let L be the recursively enumerable language (which is not context-sensitive) over Σ and L_1 the context-sensitive language over $\Sigma \cup \{a, b\}$, defined in Theorem 3.7.

Because # is a symbol which does not belong to $\Sigma \cup \{a, b\}$ then

$$(\#L_1) \overset{\#}{\mapsto} (a^*b) = [(\#L_1) \overset{\#}{\Longrightarrow} (a^*b)] \cap \#\Sigma^* = \#L.$$

□

Corollary 3.21 *The family of context-sensitive languages is not closed under controlled SD and controlled PD.*

Theorem 3.24 *The family of context-sensitive languages is closed under controlled sequential and controlled parallel deletion with singletons.*

Proof. We can assume, without loss of generality, that L is a λ-free language over the alphabet Σ. Let $\Delta : \Sigma \longrightarrow 2^{\Sigma^*}$ be a control function and, for every $a \in \Sigma$, let $A_a = (S_a, \Sigma, s_a, F_a, P_a)$ be a finite automaton that accepts the word $\Delta(a)$. Assume that the state sets S_a, $a \in \Sigma$, are pairwise disjoint. One can modify the construction found in Theorem 3.3 such that g is not a rational transducer (CS is not closed under rational transductions) but a λ-free gsm (CS is closed under λ-free gsm's).

Indeed, let g be the following gsm:

$$\begin{array}{rl} g = & (\Sigma, \Sigma \cup \{\#\}, S, s_0', \{s_0'\}, P), \\ \text{where} & \\ S = & (\cup_{a \in \Sigma} S_a) \cup \{{s_0}'\}, \\ P = & \{s_0' a \longrightarrow a s_0' |\ a \in \Sigma\} \cup \{s_0' a \longrightarrow a s_a |\ a \in \Sigma, \Delta(a) \neq \lambda\} \cup \\ & \{s_0' a \longrightarrow a \# s_0' |\ a \in \Sigma, \Delta(a) = \lambda\} \cup \\ & \{sb \longrightarrow \# s' |\ \exists a \in \Sigma : sb \longrightarrow s' \in P_a\} \cup \\ & \{sb \longrightarrow \# s_0' |\ \exists a \in \Sigma, s' \in F_a : sb \longrightarrow s' \in P_a\}. \end{array}$$

The construction differs from that of Theorem 3.3 in the following way:

- the marker # has been introduced in order to transform every erasing rule into a non- erasing one;
- during a derivation, a switch to the rules of the automaton A_a can be made only immediately after scanning the letter a in the input.

One can prove, analogously to the claim in Theorem 3.3 that:

$$\begin{aligned} g(L) = \quad & L \cup \{u_1a_1\#^{l_1}u_2a_2\#^{l_2}\dots u_ka_k\#^{l_k}u_{k+1} |\ k \geq 1, \\ & a_j \in \Sigma, 1 \leq j \leq k, u_i \in \Sigma^*, 1 \leq i \leq k+1 \text{ and } \exists u \in L : \\ & u = u_1a_1\Delta(a_1)u_2a_2\Delta(a_2)\dots u_ka_k\Delta(a_k)u_{k+1}\}, \end{aligned}$$

where l_i is the length of $\Delta(a_i)$, $1 \leq i \leq k$, with the following exception. If $\Delta(a_i) = \lambda$ then $l_i = 1$. If one considers now the morphism

$$h : (\Sigma \cup \{\#\})^* \longrightarrow \Sigma^*, h(\#) = \lambda, h(a) = a, \forall a \in \Sigma,$$

and the regular languages

$$\begin{aligned} R_1 = \quad & \Sigma^*\#^+\Sigma^*, \\ R_2 = \quad & [(\Sigma \cup \Sigma\#^+)^*(\cup_{a\in\Sigma}a\Delta(a))(\Sigma \cup \Sigma\#^+)^*]^c \cap (\Sigma^*\#\Sigma^*)^+, \end{aligned}$$

then:

$$\begin{aligned} L \mapsto \Delta \quad &= \quad h(g(L) \cap R_1), \\ L \Mapsto \Delta \quad &= \quad h(g(L) \cap R_2). \end{aligned}$$

The intersection with the language R_1 imposes that only words in which exactly one erasing is performed are retained in the result. The intersection with the language R_2 takes care of that between two sequences of markers (erased words) no other candidate for erasing appears, and that at least one erasing is performed. As the proof of the above equalities is similar to that of Theorem 3.3, the only thing that remains to be shown is that h is a q- linear erasing with respect to $g(L)$, for some integer q.

Denote, for every $a \in \Sigma$, $l_a = \lg(\Delta(a))$ and $p = \max\{l_a |\ a \in \Sigma\}$.

Let w be a word in $g(L)$. If $w \in L$ then $h(w) = w$.

Else,

$$\begin{aligned} w = \quad & u_1a_1\#^{l_1}u_2a_2\#^{l_2}\dots u_ka_k\#^{l_k}u_{k+1}, k \geq 1, \\ & a_j \in \Sigma, 1 \leq j \leq k, u_i \in \Sigma^*, q1 \leq i \leq k+1 \text{ and } \exists u \in L : \\ & u = u_1a_1\Delta(a_1)u_2a_2\Delta(a_2)\dots u_ka_k\Delta(a_k)u_{k+1}. \end{aligned}$$

The length of the word w is :

$$\lg(w) = \sum_{i=1}^{k+1} \lg(u_i) + k + \sum_{i=1}^{k} l_i$$

and, as $h(w) = u_1a_1u_2a_2 \dots u_ka_ku_{k+1}$, its length is:

$$\lg(h(w)) = \sum_{i=1}^{k+1} \lg(u_i) + k.$$

For any word $w \in g(L)$, the following inequalities hold:

$$\lg(w) = \sum_{i=1}^{k+1} \lg(u_i) + k + \sum_{i=1}^{k} l_i \leq \sum_{i=1}^{k+1} \lg(u_i) + k + k \cdot p \leq$$

$$(p+1)(\sum_{i=1}^{k} \lg(u_i) + k) = (p+1)\lg(h(w)).$$

Taking $q = p + 1$, this proves that h is a q- linear erasing with respect to $g(L)$.

As CS is closed under linear erasing, λ-free gsm mapping and intersection with regular languages, it follows that it is closed also under controlled sequential and controlled parallel deletion with singletons. □

3.5 Scattered sequential deletion

The various variants of deletion dealt with so far have been considered only from the *compact* point of view. A *scattered* variant of the sequential deletion has been defined in [13]. Given two words u and v, if the letters of v can also be found in u, in the same order, the *scattered sequential deletion* erases them from u without taking into account their places; else, the result of the scattered sequential deletion of v from u is the empty set.

Definition 3.7 *Let L_1, L_2 be languages over the alphabet Σ. The scattered sequential deletion of L_2 from L_1 (shortly, scattered SD) is defined as:*

$$L_1 \dashrightarrow L_2 = \bigcup\nolimits_{u \in L_1, v \in L_2} (u \dashrightarrow v),$$

where

$$\begin{aligned} u \dashrightarrow v = \; & \{u_1u_2 \dots u_{k+1} \in \Sigma^* \mid k \geq 1, u = u_1v_1u_2v_2 \dots u_kv_ku_{k+1}, \\ & v = v_1v_2 \dots v_k,\ u_i \in \Sigma^*, 1 \leq i \leq k+1, v_i \in \Sigma^*, 1 \leq i \leq k\}. \end{aligned}$$

The parallel variants of deletion do not have their natural scattered counterparts. Therefore we shall often use in the sequel the term *scattered deletion* instead of scattered sequential deletion.

Example 3.11 Let L_1, L_2 be the languages:

$$\begin{array}{ll} L_1 = & \{a^n b^n c^n |\ n \geq 1\}, \\ L_2 = & \{ab^2c^3\}. \end{array}$$

The scattered deletion of L_2 from L_1 is:

$$L_1 \dashrightarrow L_2 = \{a^{n+2} b^{n+1} c^n |\ n \geq 0\},$$

whereas the ordinary sequential deletion is $L_1 \longrightarrow L_2 = \emptyset$. □

Indeed, we notice that the necessary condition for a set $u \dashrightarrow v$ to be nonempty is much weaker than in the case of sequential deletion. The word v does not need to be a subword of u but u has to contain the letters of v, in the same order.

In general,

$$L_1 \longrightarrow L_2 \ \subseteq L_1 \dashrightarrow L_2$$

for every two languages L_1, L_2 over an alphabet Σ.

Obviously, the scattered deletion is not a commutative operation. Indeed, for example, $ab \dashrightarrow b = a$ whereas $b \dashrightarrow ab = \emptyset$. The scattered SD is not an associative operation either. In general, the sets $L_1 \dashrightarrow (L_2 \dashrightarrow L_3)$ and $(L_1 \dashrightarrow L_2) \dashrightarrow L_3$ are incomparable. For example,

$$ab \dashrightarrow (bc \dashrightarrow c) = a \text{ while } (ab \dashrightarrow bc) \dashrightarrow c = \emptyset,$$

$$(aab \dashrightarrow b) \dashrightarrow a = a \text{ while } aab \dashrightarrow (b \dashrightarrow a) = \emptyset.$$

As expected, the families of regular and context-free languages are closed under scattered deletion with regular languages because we have:

Theorem 3.25 *If L, R are languages over the alphabet Σ, R a regular one, the scattered deletion $L \dashrightarrow R$ is the image of L through a gsm mapping.*

Proof. Let $A = (S, \Sigma, s_0, F, P)$ be a finite automaton that recognizes the language R. We construct the gsm with erasing:

$$\begin{array}{ll} g = & (\Sigma, \Sigma, S, s_0, F, P') \\ \text{where} & \\ P' = & P \cup \{sa \longrightarrow as |\ s \in S, a \in \Sigma\}. \end{array}$$

Given $u \in L$ as an input and $v \in R$, the gsm works as follows: the rules of P erase the symbols which come from v, in the correct order, whereas those of the form $sa \longrightarrow as$ cross the symbols that will remain in $u \dashrightarrow v$.

We claim that g satisfies the equality requested by the theorem that is, $L \dashrightarrow R = g(L)$.

" $\subseteq$ " Let α be a word in $L \dashrightarrow R$,

$\alpha = u_1 u_2 \dots u_{k+1}, k \geq 1, u_i \in \Sigma^*, 1 \leq i \leq k+1,$
and $\exists u \in L, v \in R$ such that $v = v_1 v_2 \dots v_k, v_i \in \Sigma^*, 1 \leq i \leq k,$
$u = u_1 v_1 u_2 v_2 \dots u_k v_k u_{k+1}$.

The following derivation according to A exists:

$$s_0 v_1 v_2 \dots v_k \Longrightarrow^* s_1 v_2 \dots v_k \Longrightarrow^* s_{k-1} v_k \Longrightarrow^* s_k, s_k \in F.$$

Consequently, one can construct the following derivation according to g:

$$\begin{aligned} s_0 u = \quad & s_0 u_1 v_1 u_2 v_2 \dots u_k v_k u_{k+1} \Longrightarrow^* u_1 s_0 v_1 u_2 v_2 \dots u_k v_k u_{k+1} \Longrightarrow^* \\ & u_1 s_1 u_2 v_2 \dots u_k v_k u_{k+1} \Longrightarrow^* u_1 u_2 s_1 v_2 \dots u_k v_k u_{k+1} \Longrightarrow^* \\ & u_1 u_2 \dots s_{k-1} v_k u_{k+1} \Longrightarrow^* u_1 u_2 \dots s_k u_{k+1} \Longrightarrow^* u_1 u_2 \dots u_{k+1} s_k = \\ & \alpha s_k. \end{aligned}$$

We have used the rules of the derivation $s_0 v \Longrightarrow^* s_k$ to scan v_i, $1 \leq i \leq k$, and rules of the type $sa \longrightarrow as$ to cross over u_i, $1 \leq i \leq k+1$.

This proves that $\alpha \in g(u) \subseteq g(L)$.

" $\supseteq$ " Let α be a word in $g(L)$. There exists $u \in L$ and a derivation $s_0 u \Longrightarrow^* \alpha s_k$, $s_k \in F$ according to g. If we separate the rules of P from the ones in $P' - P$, the derivation has the form:

$$\begin{aligned} s_0 u = \quad & s_0 u_1 v_1 u_2 v_2 \dots u_k v_k u_{k+1} \Longrightarrow^* u_1 s_0 v_1 u_2 v_2 \dots u_k v_k u_{k+1} \Longrightarrow^* \\ & u_1 s_1 u_2 v_2 \dots u_k v_k u_{k+1} \Longrightarrow^* u_1 u_2 s_1 v_2 \dots u_k v_k u_{k+1} \Longrightarrow^* \\ & u_1 u_2 \dots s_{k-1} v_k u_{k+1} \Longrightarrow^* u_1 u_2 \dots s_k u_{k+1} \Longrightarrow^* u_1 u_2 \dots u_{k+1} s_k = \\ & \alpha s_k. \end{aligned}$$

In scanning u_i, $1 \leq i \leq k+1$, rules of $P' - P$ have been used and in scanning v_j, $1 \leq j \leq k$, rules of P have been applied.

If no rule of P has been applied then $s_0 \in F$. This implies that $\lambda \in R$ and therefore $u = u_1 u_{k+1} \in (u \dashrightarrow \lambda) \subseteq (L \dashrightarrow R)$.

Else, gathering together the rules of P which have been used, the following derivation according to A can be constructed:

$$s_0 v_1 v_2 \dots v_k \Longrightarrow^* s_1 v_2 \dots v_k \Longrightarrow^* s_{k-1} v_k \Longrightarrow^* s_k, s_k \in F,$$

which implies that the word $v = v_1 v_2 \dots v_k$ belongs to R.

This further implies that

$$\alpha = u_1 u_2 \dots u_{k+1} \in (u \dashrightarrow v) \subseteq L \dashrightarrow R$$

and the theorem is proved. □

Corollary 3.22 *The family of regular and the family of context-free languages are closed under scattered deletion with regular languages.*

Proof. The claim follows from the preceding theorem, as REG and CF are closed under gsm mapping. □

However, in general, the family of context-free languages is not closed under scattered deletion. In fact a stronger result holds.

A *linear grammar* is a grammar $G = (N, \Sigma, S, P)$ whose productions are of one of the two forms $A \longrightarrow w,\ A \longrightarrow uBv,\ A, B \in N,\ u, v, w \in \Sigma^*$. The family of linear languages strictly includes REG and is strictly included in CF.

Theorem 3.26 *There exist two linear languages L_1 and L_2 such that the scattered deletion of L_2 from L_1 is not a context-free language.*

Proof. Let L_1, L_2 be the linear languages

$$\begin{aligned} L_1 = &\ \{a^n (bc)^n (df)^m \mid n, m \geq 1\}, \\ L_2 = &\ \{c^n d^n \mid n \geq 1\}. \end{aligned}$$

One can easily see that:

$$(L_1 \dashrightarrow L_2) \cap a^* b^* f^* = \{a^n b^n f^n \mid n \geq 1\}.$$

As CF is closed under intersection with regular sets but $\{a^n b^n f^n | n \geq 1\}$ is not a context-free language, it follows that the language $L_1 \dashrightarrow L_2$ is not context-free. □

Corollary 3.23 *The family of context-free languages is not closed under scattered deletion.*

As it is not closed under right and left quotient with regular languages, CS is not closed under scattered deletion either.

Theorem 3.27 *The family of context-sensitive languages is not closed under scattered deletion with regular languages.*

Proof. Let Σ be an alphabet and denote $\Sigma' = \{a' |\ a \in \Sigma\}$. To every word $w \in \Sigma^*$ corresponds a word $w' \in \Sigma'^*$, obtained from w by changing every letter a into a'.

A λ-free gsm g can be easily constructed to satisfy:

$$g(L) = \{w_1 w_2' |\ w_1, w_2 \in \Sigma^*, w_1 w_2 \in L\}, \qquad (*)$$

where L is an arbitrary λ-free language over Σ.

If we define now the λ-free morphism $h : \Sigma^* \longrightarrow \Sigma'^*$, $h(a) = a'$, $\forall a \in \Sigma$, the following equality holds:

$$L_1/L_2 = \{[g(L_1) \dashrightarrow h(L_2)] \cap \Sigma^*\} \cup \{\lambda |\ \lambda \in L_1 \cap L_2\}$$

for every two languages L_1, L_2 over Σ.

" $\subseteq$ " Let u be a word in L_1/L_2. There exist $\alpha \in L_1$ and $v \in L_2$ such that $\alpha = uv$.

If $\alpha = u = v = \lambda$ then u belongs to the right member of the equality.

Else, according to (*), $uv' \in g(\alpha) \subseteq g(L_1)$ and as $v' \in h(L_2)$ we have:

$$u \in \begin{array}{l}(g(uv) \dashrightarrow h(v)) \cap \Sigma^* \subseteq \\ [g(L_1) \dashrightarrow h(L_2)] \cap \Sigma^*.\end{array}$$

" $\supseteq$ " If $u \in \Sigma^*$ is a word in $g(L_1) \dashrightarrow h(L_2)$, there exist $\alpha \in L_1$, $v \in L_2$ such that $u \in (g(\alpha) \dashrightarrow v')$. This further implies that there exists a decomposition of α in $\alpha = w_1 w_2$ such that $u \in w_1 w_2' \dashrightarrow v'$. As u does not contain marked letters we deduce that $w_1 = u$, $w_2 = v$ and, consequently, $\alpha = uv \in L_1$, $v \in L_2$ that is, $u \in L_1/L_2$.

If $u = \lambda$ and $\lambda \in L_1 \cap L_2$ then $u \in L_1/L_2$ too, and the proof of the equality is finished.

As CS is closed under λ-free gsm mapping, λ-free morphism, union and intersection but it is not closed under right and left quotient with regular languages, it follows that it is not closed under scattered deletion with regular languages either. □

Corollary 3.24 *The family of context-sensitive languages is not closed under scattered deletion.*

In the particular case when the language to be deleted is a singleton, CS is closed under scattered deletion. This follows because the amount of erasing is limited to the letters of a single word.

Theorem 3.28 *The family of context-sensitive languages is closed under scattered deletion with singletons.*

Proof. Let L be a context-sensitive language and w a word over the same alphabet Σ.

If w belongs to L then

$$L \dashrightarrow \{w\} = [(L - \{w\}) \dashrightarrow \{w\}] \cup \{\lambda\}.$$

If $w = \lambda$, then $L \dashrightarrow \{w\} = L$.

Consequently, the theorem will follow if we prove that $L \dashrightarrow \{w\}$ is context-sensitive for a context-sensitive L and a nonempty w not in L.

Let $A = (S, \Sigma, s_0, F, P)$, $s_0 \notin F$, be a finite automaton that accepts the word $w = a_1 \dots a_k$, $a_i \in \Sigma$, $1 \leq i \leq k$. One can easily modify the construction of Theorem 3.25 such that the gsm considered is a λ-free gsm. Indeed, let g be the gsm:

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

Given a word $y \in L$ as an input, the gsm g works as follows: if the letters of w can be found in u, in the same order, they are replaced with the marker #, the rest of the word remaining unchanged; else, a final state cannot be reached. It can easily be proved that:

$$\begin{array}{ll} g(L) = & \{u_1 \# u_2 \# \dots \# u_{k+1} |\ k \geq 1, \text{ and } \exists\, u \in L : \\ & u = u_1 a_1 u_2 a_2 \dots u_k a_k u_{k+1}, u_i \in \Sigma^*, 1 \leq i \leq k+1, \}. \end{array}$$

Note that, as we assumed that w does not belong to L, $u_1 u_2 \dots u_{k+1}$ is a nonempty word.

If one considers now the morphism $h : (\Sigma \cup \{\#\})^* \longrightarrow \Sigma^*$, $h(\#) = \lambda$, $h(a) = a$, $\forall a \in \Sigma$ it is obvious that

$$h(g(L)) = L \dashrightarrow \{w\}.$$

As, for every $u \in g(L)$ we have:

$$\begin{array}{ll} \lg(u) = & \lg(u_1 u_2 \dots u_{k+1}) + k \leq (k+1) \lg(u_1 u_2 \dots u_{k+1}) = \\ & (k+1)\ \lg(h(u)), \end{array}$$

h is a $(k+1)$-linear erasing with respect to $g(L)$.

The family of context-sensitive languages is closed under linear erasing and under λ-free gsm mapping and, consequently, under scattered deletion with singletons. □

The controlled variant of deletion does not have its natural scattered counterpart. However, a scattered variant of the permuted sequential deletion (see Section 3.3) has been defined in [13]. Given two words u and v, if the letters of v can also be found in u, they are erased without taking into account their places or their order; else, the result of the permuted scattered SD of v from u is the empty set.

Definition 3.8 *Let L_1, L_2 be languages over the alphabet Σ. The permuted scattered sequential deletion of L_2 from L_1 (shortly, permuted scattered SD) is defined by:*

$$L_1 \rightsquigarrow L_2 = \bigcup_{u \in L_1, v \in L_2} (u \rightsquigarrow v),$$

where $u \rightsquigarrow v = u \dashrightarrow com(v)$.

As we refer all the time to the sequential case, the term *permuted scattered deletion* will be used in the sequel instead of permuted scatterred sequential deletion.

The permuted scattered deletion is a generalization of SD and scattered SD. In general,

$$L_1 \longrightarrow L_2 \subseteq L_1 \dashrightarrow L_2 \subseteq L_1 \rightsquigarrow L_2,$$

for all languages L_1, L_2 over an alphabet Σ.

As $L_1 \rightsquigarrow L_2 = L_1 \dashrightarrow com(L_2)$, if one replaces the language to be deleted with a letter-equivalent language, the result of the permuted scattered SD remains unchanged.

Example 3.12 Let L_1, L_2 be the languages :

$$\begin{aligned} L_1 = &\quad \{a^3b, b^2a^2, b^3a^3, ab^4\} \\ L_2 = &\quad \{ab^2\}. \end{aligned}$$

The result of the permuted scattered deletion is:

$$L_1 \rightsquigarrow L_2 = \{a, ba^2, b^2\},$$

whereas the results of the scattered deletion and deletion are:

$$L_1 \dashrightarrow L_2 = L_1 \longrightarrow L_2 = \{b^2\}.$$

□

The permuted scattered deletion is neither a commutative nor an associative operation. In order to prove this one can use the languages chosen to show that the scattered deletion is neither commutative nor associative.

None of the families REG, CF, CS is closed under permuted scattered SD. The operation is still family preserving if the language to be erased is a singleton.

Theorem 3.29 *The family of regular languages is not closed under permuted scattered deletion.*

Proof. Let L_1, L_2 be the regular languages:

$$\begin{aligned} L_1 = &\quad \{(bc)^m(df)^p |\ m, p \geq 1\}, \\ L_2 = &\quad \{(cd)^n |\ n \geq 0\}. \end{aligned}$$

One can prove that

$$(L_1 \rightsquigarrow L_2) \cap b^*f^* = \{b^m f^m |\ m \geq 1\}.$$

□

Theorem 3.30 *The family of context-free languages is not closed under permuted scattered deletion with regular languages.*

Proof. Let L_1, L_2 be the context-free respectively regular languages:

$$\begin{aligned} L_1 = &\quad \{a^n(bc)^n(df)^m |\ n, m \geq 1\}, \\ L_2 = &\quad \{(cd)^n |\ n \geq 1\}. \end{aligned}$$

The relation

$$(L_1 \rightsquigarrow L_2) \cap a^*b^*f^* = \{a^n b^n f^n |\ n \geq 1\}$$

is obvious. □

Corollary 3.25 *The family of context-free languages is not closed under permuted scattered deletion.*

Theorem 3.31 *The family of context-sensitive languages is not closed under permuted scattered deletion with regular languages.*

Proof. Let L be the recursively enumerable language (which is not context-sensitive) over an alphabet Σ and L_1 the context-sensitive language over $\Sigma \cup \{a, b\}$ defined in Theorem 3.7. Then,

$$(L_1 \rightsquigarrow a^*b) \cap \Sigma^* = L.$$

□

Corollary 3.26 *The family of context-sensitive languages is not closed under permuted scattered deletion.*

Theorem 3.32 *The families of regular context-free and context-sensitive languages are closed under permuted scattered deletion with singletons.*

Proof. Let L be a language and w be a word over an alphabet Σ. Then,

$$L \rightsquigarrow \{w\} = \bigcup_{u \in com(w)} (L \dashrightarrow u).$$

As REG, CF, CS are closed under scattered deletion with singletons and under finite union, it follows that they are closed under permuted scattered deletion, too. □

Chapter 4

Operations: power and restrictions

4.1 Power of operations

This section is devoted to the study of classes of languages which contain simple ones and are closed under some of the operations previously defined. Each of the studied classes is closed under an insertion operation, a deletion operation and an iterative insertion one. The operations are controlled and have been chosen as stated in order to allow an increase as well as a decrease of the length of the words in the operands. The iterative operation has been included in each class to provide an infinite growth of the strings. Finally, the mirror image and the union with lambda have been introduced for technical reasons.

The *iterated controlled SIN*, needed in the sequel, can be defined starting from the controlled SIN. The formal definition of the iterated controlled SIN can be obtained from Definition 2.3 by replacing SIN with controlled SIN. If the control function is $\Delta : \Sigma \longrightarrow 2^{\Sigma'^*}$ and $\Sigma' - \Sigma \neq \emptyset$, we put $\Delta(a) = \emptyset$ for $a \in \Sigma' - \Sigma$.

Lemma 4.1 *The family of context-free languages is closed under iterated controlled SIN.*

Proof. Let L be a language generated by the context-free grammar $G = (N, \Sigma, S, P)$ and $\Delta : \Sigma \longrightarrow 2^{\Sigma'^*}$ be a control function. The fact whether or

not $\lambda \in L$ is irrelevant to the result of the controlled SIN into L. Therefore, if $\lambda \in L$ then $L{\leftarrow\!\!\dashv}^*\Delta = [(L - \{\lambda\}){\leftarrow\!\!\dashv}^*\Delta] \cup \{\lambda\}$. Consequently we can assume, without loss of generality, that L is a λ-free language. Assume that, for every $a \in \Sigma$, the language $\Delta(a)$ is generated by the context-free grammar $G_a = (N_a, \Sigma_a, S_a, P_a)$, that the nonterminal sets N, N_a, $a \in \Sigma$, are pairwise disjoint and $\Sigma' = \cup_{a\in\Sigma}\Sigma_a$. Assume further that the grammars G, G_a, $a \in \Sigma$, satisfy the requirements of Theorem 2.6.

Construct the context-free grammar:

$$\begin{aligned} G' = &\quad (N', \Sigma \cup \Sigma', S, P'), \\ N' = &\quad N \cup (\cup_{a\in\Sigma} N_a), \\ P' = &\quad P \cup (\cup_{a\in\Sigma} P_a) \cup \\ &\quad \{A \longrightarrow aS_a |\ A \in N', a \in \Sigma, A \longrightarrow a \in P \cup (\cup_{a\in\Sigma} P_a)\}. \end{aligned}$$

The construction and the proof that

$$L(G') = L{\leftarrow\!\!\dashv}^*\Delta$$

are similar to that of Theorem 2.6. The only differences are that:

- We need not consider the case where λ appears in L;
- Every letter a determines the language whose words can be inserted next to it;
- The insertions are made only to the right of the control letters. □

Lemma 4.2 *The family of regular languages is not closed under iterated controlled SIN.*

Proof. Take $L = \{ab\}$ and the control function Δ defined by:

$$\Delta : \{a, b\} \longrightarrow 2^{\{a,b\}^*}, \Delta(a) = \Delta(b) = \{ab\}.$$

Then $L = \{ab\}{\leftarrow\!\!\dashv}^*\Delta$ equals the Dyck language of order one, which is not a regular language. □

Let $\mathcal{S}$ be the smallest class of languages which contains $\emptyset$, the language $\{\lambda\}$, the singleton letters and is closed under union with the empty word, mirror image, controlled SIN, iterated controlled SIN and controlled SD with singletons. The union with lambda has been added because λ cannot occur in the result of controlled SIN and SD. If this operation wouldn't have been used, the class $\mathcal{S}$ would not contain any language L with $\lambda \in L$, except $\{\lambda\}$.

Theorem 4.1 *$\mathcal{S}$ is contained in the family of context-free languages and properly contains the family of regular languages.*

Proof. In order to show that REG $\subseteq \mathcal{S}$ we will prove the closure of $\mathcal{S}$ under catenation, union and catenation closure.

Catenation. Let L_1, L_2 be two languages in $\mathcal{S}$, over the alphabet Σ. If $\#$ is a new symbol which does not belong to Σ, let Δ_1, Δ_2 be the control functions:

$$\begin{array}{ll} \Delta_1 : \{\#\} \longrightarrow 2^{\Sigma^*}, & \Delta_2 : \Sigma \cup \{\#\} \longrightarrow 2^{\Sigma^*}, \\ \Delta_1(\#) = L_2, & \Delta_2(\#) = L_1,\ \Delta_2(a) = \emptyset, \forall a \in \Sigma. \end{array}$$

The following equality holds:

$$\#L_1L_2 = (\{\#\} \leftarrowtail \Delta_1) \leftarrowtail \Delta_2.$$

The Δ_1-controlled SIN performs the task of catenating the symbol $\#$ and the language L_2. The Δ_2-controlled SIN inserts the language L_1 in the language $\#L_2$, at the right of $\#$, realizing thus the catenation $\#L_1L_2$.

If we define now the control function:

$$\Delta_3 : \Sigma \cup \{\#\} \longrightarrow 2^{(\Sigma \cup \{\#\})^*}, \Delta_3(\#) = \emptyset, \Delta_3(a) = \#, \forall a \in \Sigma,$$

we have that:

$$\begin{array}{lll} L_1L_2 = & \mathrm{Mi}(\mathrm{Mi}(\#L_1L_2) \mapsto \Delta_3), & \text{if } \lambda \notin L_1 \cap L_2, \\ L_1L_2 = & \mathrm{Mi}(\mathrm{Mi}(\#L_1L_2) \mapsto \Delta_3) \cup \{\lambda\}, & \text{if } \lambda \in L_1 \cap L_2. \end{array}$$

The role of the Δ_3-controlled SD is to delete the symbol $\#$ in every word of $\mathrm{Mi}(\#L_1L_2)$. This operation could be performed only after Mi transferred the symbol $\#$ to the right extremity of the words. This transfer was needed because the first letter of a word cannot be erased by controlled deletion. Finally, Mi was used again, in order to obtain the desired language from its mirror image.

The catenation L_1L_2 has been obtained from $L_1, L_2, \{\#\}, \emptyset \in \mathcal{S}$ by using the operations union with lambda, mirror image, controlled SIN and controlled SD with singletons. Therefore, the class $\mathcal{S}$ is closed under catenation.

Union. We will show first that the union of two letters is a language belonging to $\mathcal{S}$. Indeed, let $\{a\}, \{b\}$ be two singleton letters. Let $\#$ be a letter different from a and b and define the control function Δ_4 by:

$$\Delta_4 : \{a, b, \#\} \longrightarrow 2^{\{a,b,\#\}^*}, \Delta_4(\#) = a, \Delta_4(a) = b, \Delta_4(b) = \emptyset.$$

The following relation holds:

$$\{\#a, \#b\} = \{\#ab\} \mapsto \Delta_4.$$

The Δ_4- controlled SD was used to obtain a set of two elements from a singleton. The additional symbol # was needed in order to make possible the deletion at the left extremity of the word ab.

If we define now the control function:

$$\Delta_5 : \{a, b, \#\} \longrightarrow 2^{\{a,b,\#\}^*}, \Delta_5(a) = \Delta_5(b) = \#, \Delta_5(\#) = \emptyset,$$

we obtain the requested set:

$$\{a, b\} = \mathrm{Mi}(\{\#a, \#b\}) \mapsto \Delta_5.$$

The role of the Δ_5-controlled deletion was to delete the symbol # and the mirror image transferred it to the right extremity of every word, to allow its deletion. Observe that another application of Mi is not needed.

As we have obtained the set $\{a, b\}$ starting from the sets $\emptyset$, #, a, b and $\{\#ab\}$ (which belongs to $\mathcal{S}$ as $\mathcal{S}$ is closed under catenation) and applying only controlled SIN, controlled SD with singletons and mirror image, we conclude that it belongs to $\mathcal{S}$.

Returning now to the general case, let L_1, L_2 be two languages in $\mathcal{S}$, over the alphabet Σ. Let $\#_1, \#_2$ be two symbols which do not occur in Σ and Δ_6, Δ_7 be the control functions:

$$\begin{array}{ll} \Delta_6 : \{\#_1, \#_2\} \longrightarrow 2^{\Sigma^*}, & \Delta_7 : \Sigma \cup \{\#_1, \#_2\} \longrightarrow 2^{\{\#_1, \#_2\}^*}, \\ \Delta_6(\#_1) = L_1, & \Delta_7(\#_1) = \#_2, \\ \Delta_6(\#_2) = L_2, & \Delta_7(\#_2) = \#_1, \Delta_7(a) = \emptyset, \forall a \in \Sigma. \end{array}$$

The following equality is now obvious:

$$\#_1\#_2L_1 \cup \#_2\#_1L_2 = (\{\#_1, \#_2\} \longleftarrow \Delta_6) \longleftarrow \Delta_7.$$

Indeed, the Δ_6-controlled SIN inserts L_1 after $\#_1$ and L_2 after $\#_2$, yielding thus $\#_1L_1 \cup \#_2L_2$. The Δ_7-controlled SIN inserts then $\#_2$ after $\#_1$ and $\#_1$ after $\#_2$.

If we further define the control functions :

$$\Delta_8 : \Sigma \cup \{\#_1, \#_2\} \longrightarrow 2^{(\Sigma \cup \{\#_1, \#_2\})^*}, \Delta_9 : \Sigma \cup \{\#_2\} \longrightarrow 2^{(\Sigma \cup \{\#_2\})^*},$$

$$\begin{array}{ll}\Delta_8(a) = \#_1, \forall a \in \Sigma \cup \{\#_2\}, & \Delta_9(a) = \#_2, \forall a \in \Sigma, \\ \Delta_8(\#_1) = \emptyset, & \Delta_9(\#_2) = \emptyset,\end{array}$$

then

$$\begin{array}{ll}L_1 \cup L_2 = & \mathrm{Mi}((\mathrm{Mi}(\#_1\#_2L_1 \cup \#_2\#_1L_2) \longmapsto \Delta_8) \longmapsto \Delta_9), \\ & \text{if } \lambda \notin L_1 \cup L_2, \\ L_1 \cup L_2 = & \mathrm{Mi}((\mathrm{Mi}(\#_1\#_2L_1 \cup \#_2\#_1L_2) \longmapsto \Delta_8) \longmapsto \Delta_9) \cup \{\lambda\}, \\ & \text{if } \lambda \in L_1 \cup L_2.\end{array}$$

The role of the Δ_8-controlled SD was to erase the symbol $\#_1$ and that of the Δ_9-controlled SD to erase $\#_2$. We needed two controlled SD's because only controlled SD with singletons had to be used. The role of the mirror image operator has been similar as in the previous cases.

We have obtained $L_1 \cup L_2$ starting with the languages $L_1, L_2, \#_1, \#_2, \emptyset$ in $\mathcal{S}$ and with the set $\{\#_1, \#_2\}$ (which consists of two letters and therefore belongs to $\mathcal{S}$) by applying only controlled SIN, mirror image, union with λ and controlled SD with singletons. Therefore $L_1 \cup L_2$ is in $\mathcal{S}$, that is, $\mathcal{S}$ is closed under union.

Catenation closure. Let L be a language in $\mathcal{S}$, over the alphabet Σ, and let $\#$ be a letter which does not belong to Σ. If Δ_{10} is the control function defined as:

$$\Delta_{10} : \Sigma \cup \{\#\} \longrightarrow 2^{\Sigma^*}, \Delta_{10}(\#) = L, \Delta_{10}(a) = \emptyset, \forall a \in \Sigma,$$

then

$$\#L^* = \{\#\} \longmapsto\!\!\!\!\!\!\leftarrow {}^*\Delta_{10}.$$

Indeed, the Δ_{10}-controlled SIN inserts words from L only to the right of $\#$, assuring that the controlled insertion amounts to catenation. Defining finally the control function

$$\Delta_{11} : \Sigma \cup \{\#\} \longrightarrow 2^{(\Sigma \cup \{\#\})^*}, \Delta_{11}(a) = \#, \forall a \in \Sigma, \Delta_{11}(\#) = \emptyset,$$

the catenation closure of L will be

$$L^* = \mathrm{Mi}(\mathrm{Mi}(\#L^*) \longmapsto \Delta_{11}) \cup \{\lambda\}.$$

With the help of the mirror image operator, which puts $\#$ to the end of words, the Δ_{11}-controlled deletion erases the letter $\#$ from all the words in $\#L^*$. Finally, Mi restores the form of the words from L.

We have obtained L^* starting from $L, \emptyset$ and $\{\#\}$ in $\mathcal{S}$ and using iterated controlled SIN, mirror image, union with λ and controlled SD with singletons. Therefore, $\mathcal{S}$ is closed under catenation closure.

As $\mathcal{S}$ contains the singleton letters and is closed under catenation, union and catenation closure, it follows that it contains all the regular languages. According to Lemma 4.2 the inclusion is proper.

The inclusion $\mathcal{S} \subseteq$ CF follows from the fact that CF is closed under mirror image, controlled SIN (see Theorem 2.18), iterated controlled SIN (see Lemma 4.1) and under controlled sequential deletion with singletons (see Corollary 3.18). □

Theorem 4.2 *The family $\mathcal{S}$ is not closed under intersection.*

Proof. We will prove that there exist two languages $L_1, L_2 \in \mathcal{S}$ whose intersection is not a context-free language. As, according to Theorem 4.1, $\mathcal{S} \subseteq$CF, this will imply that $\mathcal{S}$ is not closed under intersection.

Let L_1 be the language defined by:

$$L_1 = \{\#\} \leftarrowtail^* \Delta_1,$$

where Δ_1 is the control function $\Delta_1 : \{\#, a, b, d\} \longrightarrow 2^{\{\#,a,b,d\}^*}$ defined by:

$$\Delta_1(\#) = \{a\#b\#, b\#a\#, d\#\}, \Delta_1(a) = \Delta_1(b) = \Delta_1(d) = \emptyset.$$

Claim. $L_1 = \{w \in \#(\Sigma\#)^* |\ N_a(w) = N_b(w)\},$ where $\Sigma = \{a, b, d\}$.

" $\subseteq$ " This inclusion is obvious, as we insert an equal number of a's and b's at every iteration step.

" $\supseteq$ " We will show by induction on n that if w is a word in the right member of the equality satisfying $N_a(w) = N_b(w) = n$, then $w \in L_1$.

$n = 0$. Let $w = \#(d\#)^p$, where $p \geq 0$. Then we have:

$$w \in \{\#\} \leftarrowtail^p \Delta_1 \ \subseteq \{\#\} \leftarrowtail^* \Delta_1,$$

as w is obtained by p insertions of $d\#$ next to the first symbol $\#$.

$n \mapsto n + 1$. Assume the statement true for numbers up to n and let w be a word in $\#(\Sigma\#)^*$, containing $n + 1$ letters a and $n + 1$ letters b. The word w is of one of the forms:

$$\begin{aligned} w = &\quad \#\alpha a\#(d\#)^m b\#\beta, m \geq 0, \alpha, \beta \in (\Sigma\#)^*, \\ w = &\quad \#\alpha b\#(d\#)^m a\#\beta, m \geq 0, \alpha, \beta \in (\Sigma\#)^*. \end{aligned}$$

Assume that the first case holds, the other one being similar. Consider the word $w' = \#\alpha\beta$. According to the induction hypothesis, w' is a word in L_1. Therefore we have:

$$w \in \{w'\}\leftarrow\!\!\!\!\mid^{m+1}\Delta_1 \ \subseteq\ \{\#\}\leftarrow\!\!\!\!\mid^{*}\Delta_1.$$

Indeed, w is obtained from w' by inserting first $a\#b\#$ at the right extremity of α and then inserting m times $d\#$ at the right extremity of $a\#$. We conclude that $w \in L_1$ and the claim is proved.

If we define now the control function $\Delta_2 : \{\#, a, b, d\} \longrightarrow 2^{\{\#,a,b,d\}^*}$ by:

$$\Delta_2(\#) = \{d\#b\#, b\#d\#, a\#\}, \Delta_2(a) = \Delta_2(b) = \Delta_2(d) = \emptyset,$$

one can prove, as before, that:

$$L_2 = \{\#\}\leftarrow\!\!\!\!\mid^{*}\Delta_2 = \{w \in \#(\Sigma\#)^* |\ N_b(w) = N_d(w)\}.$$

It is easy to show that:

$$L_1 \cap L_2 = \{w \in \#(\Sigma\#)^* |\ N_a(w) = N_b(w) = N_d(w)\},$$

which is not a context-free language. As L_1 and L_2 belong to $\mathcal{S} \subseteq$CF, it follows that $\mathcal{S}$ is not closed under intersection. □

For the sake of completeness we investigate also the closure of CS under the iterated controlled SIN.

Theorem 4.3 *The family of context-sensitive languages is closed under iterated controlled SIN.*

Proof. Let L be a language generated by the context-sensitive grammar $G = (N, \Sigma, S, P)$ and $\Delta : \Sigma \longrightarrow 2^{\Sigma'^*}$ be a control function. The fact whether or not $\lambda \in L$ is irrelevant to the result of the controlled SIN into L. Therefore, if $\lambda \in L$ then $L\leftarrow\!\!\!\!\mid^{*}\Delta = [(L - \{\lambda\})\leftarrow\!\!\!\!\mid^{*}\Delta] \cup \{\lambda\}$. Consequently we can assume, without loss of generality, that L is a λ-free language. Assume that, for every $a \in \Sigma$, the language $\Delta(a)$ is generated by the context-sensitive grammar $G_a = (N_a, \Sigma_a, S_a, P_a)$, that the nonterminal sets N, N_a, $a \in \Sigma$, are pairwise disjoint and $\Sigma' = \cup_{a\in\Sigma}\Sigma_a$. Assume further that the grammars G, G_a, $a \in \Sigma$, satisfy the requirements of Theorem 2.7.

Construct the context-sensitive grammar:

$$\begin{aligned} G' = &\quad (N', \Sigma \cup \Sigma', S, P'), \\ N' = &\quad N \cup (\cup_{a\in\Sigma} N_a) \cup \{\#\}, \\ P' = &\quad P \cup (\cup_{a\in\Sigma}(P_a - \{S_a \longrightarrow \lambda\})) \cup \\ &\quad \{A \longrightarrow a\#S_a\#|\ A \in N', a \in \Sigma, A \longrightarrow a \in P \cup (\cup_{a\in\Sigma} P_a)\}. \end{aligned}$$

Define now the morphism $h : \Sigma'^* \longrightarrow \Sigma'^*$ by $h(\#) = \lambda$, $h(a) = a$ for all $a \neq \#$. The construction, the proof that

$$h(L(G')) = L \leftarrow\!\!\!\shortmid\ ^*\Delta,$$

and that h is a 3-linear erasing with respect to $L(G')$ are similar to that of Theorems 2.7, 2.6. We conclude that CS is closed under iterated controlled SIN. □

The *iterated controlled PIN* can be defined starting from the controlled PIN. The formal definition can be obtained from Definition 2.3 by replacing SIN by controlled PIN. Observe, however, that the iterated controlled PIN can be defined only when the control function Δ, defined on Σ, has as values languages over the same alphabet Σ.

In order to prove the closure of CS under iterated controlled PIN, the *workspace theorem* (see, for example, [12], pp.93-97) will be invoked.

Assume that

$$D : S = u_0 \Longrightarrow u_1 \Longrightarrow \ldots \Longrightarrow u_n = u$$

is a derivation according to a grammar G. Then the *workspace of u by the derivation D* is defined by

$$\mathrm{WS}_G(u, D) = \max\{\lg(u_i)|\ 0 \leq i \leq n\}.$$

The *workspace* of $u \in L(G)$ is defined by

$$\mathrm{WS}_G(u) = \min\{\mathrm{WS}_G(u, D)|\ D \text{ is a derivation of } u\}.$$

Note that $\mathrm{WS}_G(u) \geq \lg(u)$ for any G and u.

The *Workspace Theorem* claims that if $G = (N, T, S, P)$ is a type-0 grammar and there is a natural number p such that

$$\mathrm{WS}(u) \leq p \times \lg(u),$$

for all nonempty words $u \in L(G)$, then $L(G)$ is a context-sensitive language.

Lemma 4.3 *The family of context-sensitive languages is closed under iterated controlled PIN.*

Proof. Let L be a language generated by the context-sensitive grammar $G = (N, \Sigma, S, P)$, and let $\Delta : \Sigma \longrightarrow 2^{\Sigma^*}$ be a control function, $\Delta(a) \neq \emptyset$, $\forall a \in \Sigma$. Assume that, for every $a \in \Sigma$, the language $\Delta(a)$ is generated by the context- sensitive grammar $G_a = (N_a, \Sigma_a, S_a, P_a)$, that the nonterminal sets N, N_a, $a \in \Sigma$ are pairwise disjoint and that $\cup_{a\in\Sigma}\Sigma_a \subseteq \Sigma$. Assume further that all the above grammars satisfy the requirements of Theorem 2.7. The fact whether or not $\lambda \in L$ does not affect the result of controlled PIN into L. If λ belongs to L then $L \leftarrowtail^* \Delta = [(L - \{\lambda\}) \leftarrowtail^* \Delta] \cup \{\lambda\}$.Consequently we will assume, without loss of generality, that L is λ-free.

We can construct then the following grammar:

$$
\begin{aligned}
G' = \;& (N', \Sigma', S', P'),\\
\Sigma' = \;& \Sigma \cup \{\$, \#\},\\
N' = \;& N \cup (\cup_{a\in\Sigma} N_a) \cup \{S', X\},\\
P' = \;& P \cup (\cup_{a\in\Sigma}(P_a - \{S_a \longrightarrow \lambda\})) \cup\\
& \{S' \longrightarrow XS', S' \longrightarrow \$S\#, X\$ \longrightarrow \$X\} \cup\\
& \{Xa \longrightarrow aX \mid a \in \Sigma, \lambda \in \Delta(a)\} \cup\\
& \{Xa \longrightarrow aS_aX \mid a \in \Sigma\} \cup\\
& \{Xa\# \longrightarrow aS_a\# \mid a \in \Sigma\} \cup\\
& \{Xa\# \longrightarrow a\# \mid a \in \Sigma, \lambda \in \Delta(a)\},
\end{aligned}
$$

where $S', X, \$, \#$ are new symbols which do not occur in any of the given alphabets.

Intuitively, the grammar G' works as follows. First, a sentential form of the type $X^n\$w\#$, $w \in L$ is generated, where n represents the number of parallel iterations that will be made into w. X starts to move to the right. When crossing a letter a, it generates at its left a start symbol of $\Delta(a)$. The rules $S_a \longrightarrow \lambda$ are never needed. When X reaches the right extremity of the sentential form, it disapears.

The language $L(G')$ is context-sensitive. Indeed, all rules of G' except the ones of the form $Xa\# \longrightarrow a\#$ are length-increasing. However, the application of such a rule during a minimal derivation of a word $\alpha \in L(G)$ is always preceded by the application of a rule $Xa \longrightarrow aS_aX$. If this wouldn't be the case, our X would represent a dummy iteration step, in which all the inserted words are empty. This would further imply that the derivation for

α is not minimal, as α could be obtained with a shorter derivation where the dummy iteration step is omitted.

The rule $Xa \longrightarrow aS_aX$ increases the length of the sentential form by one and the rule $Xa\# \longrightarrow a\#$ decreases its length by one. Combining these observations we conclude that the longest sentential form in a terminal derivation of α has the length smaller than or equal to $\lg(\alpha) + 1$.

Consequently, for all words $\alpha \in L(G')$ we have:

$$\mathrm{WS}_{G'}(\alpha) \leq 2 \lg(\alpha),$$

and, according to the workspace theorem, $L(G')$ is a context-sensitive language.

If we consider now the morphism $h : (\Sigma \cup \{\$, \#\})^* \longrightarrow \Sigma^*$, defined by $h(\$) = h(\#) = \lambda$ and $h(a) = a$ for $a \in \Sigma$ it can be proved that $h(L(G')) = L \Leftarrow\!\!| \, ^*\Delta$. The construction and the proof are analogous to that of Theorem 2.9. The only difference is that here, every letter determines which words can be inserted after it. Clearly, h is 3-linear erasing with respect to the language $L(G')$. □

Lemma 4.4 *The family of regular and the family of context-free languages are not closed under iterated controlled PIN.*

Proof. Take $L = \{a\}$ and the control function Δ defined by $\Delta(a) = a$. Then,

$$L \Leftarrow\!\!| \, ^*\Delta = \{a^{2^n} \mid n \geq 0\},$$

which is not a context-free language. □

Let $\mathcal{P}$ be the smallest class of languages which contains the empty set, the language $\{\lambda\}$, the singleton letters and is closed under mirror image, union with λ, controlled PIN, iterated controlled PIN and controlled PD with singletons.

Theorem 4.4 *$\mathcal{P}$ is contained in the family of context-sensitive languages and properly contains the family of regular languages.*

Proof. The fact that $\mathcal{P}$ contains REG can be shown using the proof of Theorem 4.1. The control functions that appear in the proof have the value $\emptyset$ for some arguments. However, in the case of controlled PIN (controlled PD), the control function cannot have the empty set as its value. We will modify the functions as follows. Let \$ be a new symbol which does not occur

in any of the alphabets used in Theorem 4.1. For every controlled SIN and iterated controlled SIN (resp. controlled SD) the control function receives the value λ (resp. the value \$) for all those letters for which it had previously the value $\emptyset$. After this change, one notices that if we replace everywhere in the proof the controlled SIN, iterated controlled SIN, controlled SD with singletons with controlled PIN, iterated controlled PIN, controlled PD with singletons respectively, the same relations hold. This happens because, in all the cases occurring in the proof of Theorem 4.1, the parallel insertion or deletion will amount in fact to sequential insertion or deletion.

According to Lemma 4.4, the inclusion REG$\subseteq \mathcal{P}$ is proper.

The inclusion of $\mathcal{P}$ in CS follows from the fact that CS is closed under mirror image, controlled PIN (see Theorem 2.19), iterated controlled PIN (see the preceding Lemma) and controlled PD with singletons (see Theorem 3.24). □

Let $\mathcal{P}'$ be the smallest class of languages containing the empty set, $\{\lambda\}$, the singleton letters and closed under mirror image, union with λ, controlled PIN, iterated controlled PIN and controlled PD. The difference between $\mathcal{P}$ and $\mathcal{P}'$ is that, in the case of $\mathcal{P}$, the controlled PD is restricted to the case where only singletons are erased.

Theorem 4.5 *$\mathcal{P}'$ is a Boolean algebra properly containing the family of regular languages.*

Proof. The family $\mathcal{P}$ is included in $\mathcal{P}'$, therefore $\mathcal{P}'$ properly contains the family of regular languages.

It will be showed in the following that $\mathcal{P}'$ is closed under complementation. Let L be a language in $\mathcal{P}'$, over the alphabet Σ, and let $\#, \$$ be letters which do not occur in Σ. Then,

$$\{\#\$\} \cup \#L^c\$\$ = \#\Sigma^*\$\$\# \models \Delta_1,$$

where Δ_1 is the control function:

$$\Delta_1 : \Sigma \cup \{\#, \$\} \longrightarrow 2^{(\Sigma \cup \{\#, \$\})^*}, \Delta_1(\#) = L\$, \Delta_1(a) = \Delta_1(\$) = \#, \forall a \in \Sigma.$$

Indeed, given a word $w = \#u\$\$\# \in \#\Sigma^*\$\$\#$, the Δ_1-controlled PD:

- if $u \in L$, erases both $u\$$ and the last $\#$, yielding $\#\$$;
- if $u \in \Sigma^* - L$, erases only the last $\#$, yielding $\#u\$\$$.

One can use the control function Δ_2 to erase the marker \$, where

$$\Delta_2 : \Sigma \cup \{\#, \$\} \longrightarrow 2^{(\Sigma \cup \{\$, \#\})^*},$$

$$\Delta_2(\#) = \Delta_2(\$) = \Delta_2(a) = \$\$, \forall a \in \Sigma.$$

Consequently we have:

$$\#L^c = (\{\#\$\} \cup \#L^c\$\$) \models \Delta_2.$$

Using now the control function Δ_3 to erase the marker #:

$$\Delta_3 : \Sigma \cup \{\#\} \longrightarrow 2^{(\Sigma \cup \{\#\})^*}, \Delta_3(\#) = \Delta_3(a) = \#, \forall a \in \Sigma,$$

we obtain,

$$\begin{array}{lll} L^c = & \mathrm{Mi}(\mathrm{Mi}(\#L^c) \models \Delta_3), & \text{if } \lambda \in L, \\ L^c = & \mathrm{Mi}(\mathrm{Mi}(\#L^c) \models \Delta_3) \cup \{\lambda\}, & \text{if } \lambda \notin L. \end{array}$$

The language $\#\Sigma^*\$\$\#$, can be obtained from the singleton letters by using catenation and catenation closure. As, in order to obtain L^c, we have started from Σ, \$, # and L, and we have used only the operations of $\mathcal{P}'$, we deduce that $L^c \in \mathcal{P}'$. Being closed under complementation and union, $\mathcal{P}'$ is closed also under intersection. Consequently $\mathcal{P}'$ is a Boolean algebra. □

4.2 Modifying the operands

In this section a particular case of sequential insertion will be considered, namely the case when the result of the sequential insertion is a regular language. The main theorems of the section state that, if the result of sequential insertion $L_1 \leftarrow L_2$ is regular, the same result can be obtained by replacing L_2 with a regular language R such that $L_2 \subseteq R$.

Before proving these results for the sequential insertion, the simpler case of catenation will be considered.

Theorem 4.6 *Let L_1, R be languages over the alphabet Σ, R a regular one. If there exists $L_2 \subseteq \Sigma^*$ with the property $L_1 L_2 = R$ then there exists a regular language R', $L_2 \subseteq R' \subseteq \Sigma^*$, with the same property.*

Proof. Let R' be the language defined by:

$$R' = (L_1 \backslash R^c)^c.$$

(i) R' is a regular language. Indeed, the left quotient of a regular language by an arbitrary language is regular (see Theorem 3.1).

(ii) $L_1R' \subseteq R$. Assume, for the sake of contradiction, that L_1R' is not included in R. There exist then words $u \in L_1$, $v \in R'$, such that $uv \in R^c$. This implies that $v = (u\backslash uv) \subseteq (L_1\backslash R^c)$ - a contradiction with the fact that v was a word in R'.

(iii) Any language L_2 with the property $L_1L_2 \subseteq R$ is included in R'. Indeed, assume that there exists a language L_2 as before such that $L_2 - R' \neq \emptyset$. Let v be a word in $L_2 - R'$. As v belongs to $L_1\backslash R^c$, there exist words $w \in R^c$, $u \in L_1$, such that $uv = w$. This implies $w \in L_1L_2 \subseteq R$ - a contradiction with the fact that w was a word in R^c.

If there exists a language L_2 such that $L_1L_2 = R$, according to (iii), $L_2 \subseteq R'$ and therefore $R = L_1L_2 \subseteq L_1R'$. As, according to (ii), we have that $L_1R' \subseteq R$, we deduce that $L_1R' = R$. It has been showed in (i) that R' is a regular language, therefore the proof of the theorem is complete. □

Corollary 4.1 *The regular language R' from the preceding theorem can be effectively constructed if L_1 is a regular or context-free language.*

Proof. It follows from the preceding theorem and from Corollary 3.1. □

Corollary 4.2 *Let R be a regular language over an alphabet Σ. There exists a finite number $n \geq 1$ of distinct regular languages R'_i, $1 \leq i \leq n$, such that for any $L_1 \subseteq \Sigma^*$ the following statements are equivalent:*

(i) There exists a language $L_2 \subseteq \Sigma^*$ with the property $L_1L_2 = R$.

(ii) There exists i, $1 \leq i \leq n$, such that $L_1R'_i = R$.

Moreover, the regular languages R'_i, $1 \leq i \leq n$, can be effectively constructed.

Proof. It follows from the preceding theorem and from Corollary 3.2. The languages R'_i, $1 \leq i \leq n$, are constructed by forming the complements of all the possible (finitely many) languages that can be obtained from R^c by left quotient. Since the equivalence problem is decidable for regular languages, duplicates can be removed from the list R'_i. □

Observe that the list obtained in the preceding corollary may contain languages R'_j for which the equality $L_1 R'_j = R$ does not hold for any language L_1. However, these languages can be removed from the list in the following way. Note that, by using the mirror image operator, results similar to Theorem 4.6, Corollary 4.1 and Corollary 4.2 can be obtained also for the left operand.

Theorem 4.7 *Let L_2, R be languages over the alphabet Σ, R a regular one. If there exists $L_1 \subseteq \Sigma^*$ with the property $L_1 L_2 = R$ then there exists a regular language R'', $L_1 \subseteq R'' \subseteq \Sigma^*$, with the same property.*

Corollary 4.3 *The regular language R'' from the preceding theorem can be effectively constructed if L_2 is a regular or context-free language.*

Corollary 4.4 *Let R be a regular language over an alphabet Σ. There exists a finite number $m \geq 1$ of distinct regular languages R''_i, $1 \leq i \leq m$ such that for any $L_2 \subseteq \Sigma^*$ the following statements are equivalent:*

(i) There exists a language $L_1 \subseteq \Sigma^*$ with the property $L_1 L_2 = R$.

(ii) There exists i, $1 \leq i \leq m$, such that ${R''}_i L_2 = R$.

Moreover, the regular languages ${R''}_i$, $1 \leq i \leq m$, can be effectively constructed.

We are now in position to effectively exclude from the list of Corollary 4.2 the languages R'_j for which the equality $L_1 R'_j = R$ does not hold for any L_1. According to the preceding corollary, if such an L_1 would exist then we would also have $R''_i R'_j = R$ for some index i, $1 \leq i \leq m$.

For each j, $1 \leq j \leq n$, check, for all i, $1 \leq i \leq m$, whether or not ${R''}_i R'_j = R$. If the equality holds for at least one index i, the language R'_j is retained in the list, otherwise it is eliminated.

In a similar way, we can effectively exclude from the list in Corollary 4.4 the languages ${R''}_j$ for which the equality ${R''}_j L_2 = R$ does not hold for any L_2.

In order to prove similar results for the more general case of sequential insertion, a new operation will be introduced: *the dipolar deletion*, denoted by $\rightleftharpoons$. The dipolar deletion of the word v from the word u is the set consisting of the words obtained from u by erasing a prefix and a suffix whose catenation equals v. The dipolar deletion is needed in this section to perform a task "inverse" to SIN: if $v \leftarrow w = u$ then w can be recovered

from u by dipolar deletion of v. However, the sequential insertion and the dipolar deletion are not inverse operations. In general, if L_1, L_2, L_3 are languages over Σ such that $L_1 \leftarrow L_2 = L_3$ then $L_2 \subseteq L_3 \rightleftharpoons L_1$, but the other inclusion does not hold.

Definition 4.1 *Let L_1, L_2 be languages over the alphabet Σ. The dipolar deletion of L_2 from L_1 is:*

$$L_1 \rightleftharpoons L_2 = \bigcup_{u \in L_1, v \in L_2} (u \rightleftharpoons v),$$

where

$$u \rightleftharpoons v = \quad \{w \in \Sigma^* | \; \exists \, v_1, v_2 \in \Sigma^* : u = v_1 w v_2, v = v_1 v_2\}.$$

Example 4.1 Let $L_1 = \{ab\}$, $L_2 = \{aba\}$ and $L_3 = \{abaab, aabab, ababa\} = L_1 \leftarrow L_2$. The dipolar deletion of L_1 from L_3 is:

$$L_3 \rightleftharpoons L_1 = \{aba, baa, aab\},$$

a set which strictly includes L_2. □

Theorem 4.8 *The family of context-sensitive languages is not closed under dipolar deletion with regular languages.*

Proof. Let L_1, L_2 be languages over an alphabet Σ and $\#, \$$ be letters which do not occur in Σ. The theorem follows from the fact that we have

$$\#L_1\$ \rightleftharpoons \#L_2 = (L_2 \backslash L_1)\$,$$

and the family of context-sensitive languages is not closed under left quotient with regular languages. □

Theorem 4.9 *The family of context-sensitive languages is closed under dipolar deletion with singletons.*

Proof. Let L be a language and w be a word over the same alphabet Σ. The theorem follows as we have

$$L \rightleftharpoons \{w\} = \bigcup_{w_1, w_2}^{w_1 w_2 = w} (w_1 \backslash L)/w_2,$$

and CS is closed under left and right quotient with singletons and under finite union. □

Theorem 4.10 *The family of context-free languages is not closed under dipolar deletion.*

Proof. The proof is similar to that of Theorem 3.4. Let L_1, L_2 be the languages defined by:

$$\begin{aligned} L_1 = &\quad \#\{a^i b^{2i} |\ i > 0\}^*\$, \\ L_2 = &\quad \#a\{b^i a^i |\ i > 0\}^*. \end{aligned}$$

Then we have

$$(L_1 \rightleftharpoons L_2) \cap b^+\$ = \{b^{2^n}\$ |\ n > 0\},$$

which is not a context-free language. □

The following result is analogous to Lemma 3.1: the result of the dipolar deletion from a regular language is regular regardless the complexity of the deleted language.

Lemma 4.5 *Let L, R be two languages over the alphabet Σ. If R is a regular language then the language $R \rightleftharpoons L$ is regular.*

Proof. Let $A = (S, \Sigma, s_0, F, P)$ be a finite automaton that accepts the language R, in which all the states are useful. A state is called useful if there exists a path containing it which starts from the initial state and ends in a final state. For every two states s_1, s_2 in S define:

$$L_{s_1,s_2} = \{w \in \Sigma^* |\ s_1 w \Longrightarrow^* s_2 \text{ in } A\}.$$

It will be showed in the following that:

$$R \rightleftharpoons L = \bigcup\nolimits_{(s_1,s_2)\in S'} L_{s_1,s_2}, \qquad (*)$$

where

$$S' = \{(s_1, s_2) \in S \times S |\ \exists s_f \in F :\ L_{s_0,s_1} L_{s_2,s_f} \cap L \neq \emptyset\}.$$

Indeed, let w be a word in $R \rightleftharpoons L$. According to the definition of the dipolar deletion, there exist words $u \in R$, $v \in L$, $v_1, v_2 \in \Sigma^*$ such that $u = v_1 w v_2$, $v = v_1 v_2$.

As u belongs to $R = L(A)$ there exists a derivation

$$s_0 u = s_0 v_1 w v_2 \Longrightarrow^* s_1 w v_2 \Longrightarrow^* s_2 v_2 \Longrightarrow^* s_f, s_f \in F,$$

according to the rules of P. The existence of the subderivations :

$$\begin{aligned} &s_0v_1 \Longrightarrow^* s_1, \\ &s_1w \Longrightarrow^* s_2, \\ &s_2v_2 \Longrightarrow^* s_f, \quad s_f \in F, \end{aligned}$$

implies that $v_1 \in L_{s_0,s_1}$, $w \in L_{s_1,s_2}$ and $v_2 \in L_{s_2,s_f}$. As v_1v_2 belongs to the set $L_{s_0,s_1}L_{s_2,s_f}$ and also to L, it follows that $L_{s_0,s_1}L_{s_2,s_f} \cap L \neq \emptyset$, and $(s_1, s_2) \in S'$. Consequently, w is a word in the right member of (*).

For the reverse inclusion let w be a word in the right member of (*). There exist states $s_1, s_2 \in S$ and $s_f \in F$ such that $w \in L_{s_1,s_2}$ and $L_{s_0,s_1}L_{s_2,s_f} \cap L \neq \emptyset$. Let v be a word in $L_{s_0,s_1}L_{s_2,s_f} \cap L$, being therefore of the form $v = v_1v_2$ where $v_1 \in L_{s_0,s_1}$ and $v_2 \in L_{s_2,s_f}$.

According to the definition of L_{s_1,s_2} the following derivations exist in A:

$$\begin{aligned} &s_0v_1 \Longrightarrow^* s_1, \\ &s_1w \Longrightarrow^* s_2, \\ &s_2v_2 \Longrightarrow^* s_f. \end{aligned}$$

We can construct then the following accepting derivation in A:

$$s_0v_1wv_2 \Longrightarrow^* s_1wv_2 \Longrightarrow^* s_2v_2 \Longrightarrow^* s_f, s_f \in F,$$

which shows that v_1wv_2 is a word in $R = L(A)$. As $v = v_1v_2$ is a word in L, according to the definition of the dipolar deletion, we deduce that w is a word in the set $(v_1wv_2 \rightleftharpoons v) \subseteq R \rightleftharpoons L$. The equality (*) is therefore proved.

The lemma now follows as $R \rightleftharpoons L$ is a regular language, being equal to a finite union of regular languages. □

Corollary 4.5 *The language $R \rightleftharpoons L$ can be effectively constructed if R is a regular language and L is a regular or context-free language.*

Proof. The claim follows from the proof of the preceding lemma. Indeed, if R is a regular language and L is regular (context-free) then the language $L_{s_0,s_1}L_{s_2,s_f} \cap L$ is regular (context-free) for any states s_1, s_2, s_f. As the emptiness problem is decidable for regular (context-free) languages, the set S' and therefore the language $R \rightleftharpoons L$, can be effectively constructed.

Corollary 4.6 *For any regular language R there exist finitely many languages that can be obtained from R by dipolar deletion.*

Proof. It follows from the preceding lemma by the fact that the automaton A is finite. This implies that there are finitely many different possibilities of constructing the union from (*).

The languages that can be obtained from R by dipolar deletion will be among the languages:

$$L_{S'} = \bigcup_{(s_1,s_2)\in S'} L_{s_1,s_2},$$

where S' is an arbitrary subset of $S \times S$. There exists at most $2^{\mathrm{card}(S\times S)}$ such different languages. □

The lemma is used to prove the main result of this section: if the result of SIN between two languages is regular, the same result can be obtained by replacing the inserted language with a regular one. This language can be effectively constructed if the language in which the insertion was made is regular or context-free.

Theorem 4.11 *Let L_1, R be languages over the alphabet Σ, R a regular one. If there exists $L_2 \subseteq \Sigma^*$ with the property $L_1 \leftarrow L_2 = R$ then there exists a regular language R', $L_2 \subseteq R' \subseteq \Sigma^*$, with the same property.*

Proof. Let R' be the language defined by $R' = (R^c \rightleftharpoons L_1)^c$.

(i) From Lemma 4.5 and from the fact that REG is closed under complementation it follows that R' is a regular language.

(ii) $L_1 \leftarrow R' \subseteq R$. Indeed, assume that $L_1 \leftarrow R'$ is not included in R. There exist then words $u \in L_1$, $v \in R'$ and a decomposition of u, $u = u_1u_2$ such that u_1vu_2 does not belong to R. According to the definition of the dipolar deletion, the word v belongs to $R^c \rightleftharpoons L_1$. This contradicts our assumption that $v \in R' = (R^c \rightleftharpoons L_1)^c$. We conclude that $L_1 \leftarrow R' \subseteq R$.

(iii) Any language $L_2 \subseteq \Sigma^*$ with the property $L_1 \leftarrow L_2 \subseteq R$ is included in R'. Indeed, assume that there exists a language $L_2 \subseteq \Sigma^*$, $L_1 \leftarrow L_2 \subseteq R$ such that L_2 is not included in R'. There exists then a word v in $L_2 - R'$. As v belongs to $(R')^c = R^c \rightleftharpoons L_1$, there exist words $w \in R^c$, $u \in L_1$ and u_1, $u_2 \in \Sigma^*$ such that $w = u_1vu_2$, and $u = u_1u_2$. The word $w = u_1vu_2$ is an element of the set $(u \leftarrow v) \subseteq L_1 \leftarrow L_2 \subseteq R$. We arrived at a contradiction as w was a word in R^c. Consequently, our assumption was false and L_2 is included in R'.

If there exists $L_2 \subseteq \Sigma^*$ with the property $L_1 \leftarrow L_2 = R$, according to (iii), $L_2 \subseteq R'$ and therefore $R = L_1 \leftarrow L_2 \subseteq L_1 \leftarrow R'$. As, according to

(ii) we have that $L_1 \leftarrow R' \subseteq R$, we deduce that $L_1 \leftarrow R' = R$. It has been showed in (i) that R' is a regular language, therefore the proof of the theorem is complete. □

Corollary 4.7 *The regular language R' from Theorem 4.11 can be effectively constructed if L_1 is a regular or context-free language.*

Proof. It follows from the preceding theorem and from Corollary 4.5. □

Corollary 4.8 *Let R be a regular language over an alphabet Σ. There exists a finite number $n \geq 1$ of distinct regular languages R'_i, $1 \leq i \leq n$, such that for any $L_1 \subseteq \Sigma^*$ the following statements are equivalent:*

(i) There exists a language $L_2 \subseteq \Sigma^*$ with the property $L_1 \leftarrow L_2 = R$.

(ii) There exists i, $1 \leq i \leq n$, such that $L_1 \leftarrow R'_i = R$.

Moreover, the regular languages R'_i, $1 \leq i \leq n$, can be effectively constructed.

Proof. It follows from Corollary 4.6 and from Theorem 4.11. The languages R'_i, $1 \leq i \leq n$, are constructed by forming the complements of all the possible (finitely many) languages that can be obtained from R^c by dipolar deletion. Since the equivalence problem is decidable for regular languages, duplicates can be removed from the list R'_i. □

Observe that the list obtained in the preceding corollary may contain languages R'_i for which the equality $L_1 \leftarrow R'_i = R$ does not hold for any language L_1. However, these languages can be eliminated from the list as shown in the end of this section.

Theorem 4.11 was concerned with the replacement of the right operand of the insertion with a regular one which produced the same result. An analogous theorem can be proved for the left operand. Here, instead of the dipolar deletion, the sequential deletion is used as an operation which performs a task "inverse" to SIN. If L_1, L_2, L_3 are languages over the alphabet Σ such that $L_1 \leftarrow L_2 = L_3$, the language L_1 can be recovered from L_3 by sequentially deleting L_2. However, the two operations are not inverse to each other. In general, if $L_1 \leftarrow L_2 = L_3$ we have that $L_1 \subseteq L_3 \rightarrow L_2$, but the reverse inclusion does not hold.

Theorem 4.12 *Let L_2, R be two languages over the alphabet Σ, R a regular one. If there exists $L_1 \subseteq \Sigma^*$ with the property $L_1 \leftarrow L_2 = R$ then there exists a regular language R', $L_1 \subseteq R' \subseteq \Sigma^*$ with the same property.*

Proof. Let $R' = (R^c \longrightarrow L_2)^c$.

(i) According to Lemma 3.1, $R^c \longrightarrow L_2$ is a regular language and, therefore, R' is also regular.

(ii) The language $R' \longleftarrow L_2$ is included in R. Indeed, let us assume that $R' \longleftarrow L_2$ is not included in R. There exist then $u = u_1u_2 \in R'$ where $u_1, u_2 \in \Sigma^*$, and $v \in L_2$ such that u_1vu_2 is an element of the set $(R' \longleftarrow L_2) - R$. As the word u_1vu_2 belongs to R^c, $u = u_1u_2$ is a word in the set $(u_1vu_2 \longrightarrow v) \subseteq R^c \longrightarrow L_2$. This contradicts the fact that $u \in R' = (R^c \longrightarrow L_2)^c$. Our assumption was false and, therefore, $R' \longleftarrow L_2 \subseteq R$.

(iii) Any language $L_1 \subseteq \Sigma^*$ with the property $L_1 \longleftarrow L_2 \subseteq R$ is included in R'. Indeed, assume that this is not the case and let $L_1 \subseteq \Sigma^*$ be a language such that $L_1 \longleftarrow L_2 \subseteq R$ and L_1 is not included in R'. Let u be a word in $L_1 - R'$. The word u belongs to $R'^c = R^c \longrightarrow L_2$ and therefore there exist $w \in R^c$, $v \in L_2$ and $u_1, u_2 \in \Sigma^*$ such that $w = u_1vu_2$, $u = u_1u_2$. According to the definition of the sequential insertion, $w = u_1vu_2$ is a word in the set $(u_1u_2 \longleftarrow v) \subseteq L_1 \longleftarrow L_2 \subseteq R$. This contradicts the fact that $w \in R^c$. Our assumption was false, therefore we conclude that $L_1 \subseteq R'$.

If there exists a language $L_1 \subseteq \Sigma^*$ with the property $L_1 \longleftarrow L_2 \subseteq R$ then, according to (iii), $L_1 \subseteq R'$. This implies that $R = L_1 \longleftarrow L_2 \subseteq R' \longleftarrow L_2$. As, according to (ii), $R' \longleftarrow L_2 \subseteq R$, we conclude that $R' \longleftarrow L_2 = R$. The theorem now follows as it has been showed in (i) that R' is a regular language. □

Corollary 4.9 *The language R' from Theorem 4.12 can be effectively constructed if L_2 is a regular or a context-free language.*

Proof. It follows from Theorem 4.12 and Corollary 3.4. □

Corollary 4.10 *Let R be a regular language over the alphabet Σ. There exists a finite number $n \geq 1$ of distinct regular languages R'_i, $1 \leq i \leq n$, such that, for every $L_2 \subseteq \Sigma^*$, the following statements are equivalent:*

(i) There exists $L_1 \subseteq \Sigma^*$ such that $L_1 \longleftarrow L_2 = R$.

(ii) There exists i, $1 \leq i \leq n$, such that $R'_i \longleftarrow L_2 = R$.

Moreover, the regular languages R'_i, $1 \leq i \leq n$, can be effectively constructed.

Proof. It follows from Corollary 3.5 and Theorem 4.12. The languages R'_i, $1 \leq i \leq n$, are constructed by forming the complements of all the possible (finitely many) languages that can be obtained from R^c by sequential

deletion. The duplicates can be eliminated as the equivalence problem is decidable for regular languages. □

Observe that the above list may contain languages R'_i for which the equality $R'_i \leftarrow L_2 = R$ does not hold for any language L_2. These languages can be eliminated from the list in a similar way as done in the remarks following Corollary 4.2, and using the twin list obtained in Corollary 4.8.

4.3 Derivatives

In studying the left and right quotient as operations on languages, of special interest is the case where the language to be deleted is a singleton.

The *left derivative* of a language L over Σ with respect to a word $w \in \Sigma^*$ is obtained as a particular case of left quotient:

$$\partial^l_w L = \{u \in \Sigma^* | \; wu \in L\}.$$

The *right derivative* of the language L with respect to the word w is defined similarily as:

$$\partial^r_w L = \{u \in \Sigma^* | \; uw \in L\}.$$

A natural generalization of the right and left derivative is the operation where the word w is extracted not from the left or right extremity of a word in L but from an arbitrary place in it.

Definition 4.2 *Let L be a language and w be a word over the alphabet Σ. The derivative of L with respect to w is defined as:*

$$\partial_w L = \{uv \in \Sigma^* | \; uwv \in L\}.$$

Example 4.2 Let $L = \{abbbab, aaabbb, abab\}$ and $w = ab$. The derivative of L with respect to w is:

$$\partial_w L = \{bbab, abbb, aabb, ab\}$$

whereas the left and right derivatives are respectively:

$$\partial^l_w L = \{bbab, ab\}, \; \partial^r_w L = \{abbb, ab\}.$$

□

The derivative is a particular case of sequential deletion where the language to be deleted consists of a single word. One can define, in a similar way, *the iterated, permuted, controlled, scattered* and *permuted scattered derivative* as particular cases of iterated, permuted, controlled, scattered and permuted scattered sequential deletion respectively. The closure properties of REG, CF, CS under all these types of derivatives have been studied in Chapter 3.

In this section, some properties of the derivatives of regular languages are dealt with. Before that, some supplementary notions will be introduced. Let L be a regular language and $A = (S, \Sigma, s_0, F, P)$ a finite deterministic automaton that accepts it. For every word $w \in \Sigma^*$ define the function $f_w^A : S \longrightarrow S$ as follows:

$$f_w^A(s) = s' \text{ iff } sw \Longrightarrow^* s' \text{ in } A.$$

The function is total. If the automaton is clear from the context, we will denote the function simply by f_w.

Let L be a language over an alphabet Σ. The relations $\mathcal{E}_L$ and $\equiv_L$ over Σ^*, referred to as the *equivalence* and the *congruence* relation *induced by* L are defined as follows. $u\mathcal{E}_L w$ iff, for all $y \in \Sigma^*$, uy is in L exactly when wy is in L. $u \equiv_L w$ iff, for all $x, y \in \Sigma^*$, $xuy \in L$ exactly when $xwy \in L$. Details about the equivalence and congruence relations induced by languages can be found for example in [11], pp.27-31, [5], pp.65-67.

It is known that $u\mathcal{E}_L w$ iff the left derivatives of L with respect to u and w coincide, that is, $\partial_u^l L = \partial_w^l L$. As regards the congruence relation, $u \equiv_L w$ iff $f_u = f_w$ in a minimal finite deterministic automaton that accepts L. (Here minimal refers to the number of states.) Obviously, if $u \equiv_L w$ then $\partial_u L = \partial_w L$. The reverse implication does not hold, as proved by the following example.

Example 4.3 Let $L = (ababa)^*$ and $w_1 = babaa$, $w_2 = baaba$, $w_3 = abaab$, $w_4 = aabab$. Then we have:

$$\partial_{w_i} L = (ababa)^+, \text{ for } i = 1, 2, 3, 4,$$

but $w_i \not\equiv_L w_j$ for $i \neq j$. For instance, $aw_1baba \in L$ but $aw_ibaba \notin L$ for $i \neq 1$. □

The derivatives define an equivalence relation $\mathcal{D}_L$ over Σ^* by $u\mathcal{D}_L v$ iff $\partial_u L = \partial_v L$. $\mathcal{D}_L$ is an equivalence relation with a "coarser" class division

than $\equiv_L$: each equivalence class of $\mathcal{D}_L$ consists of one or more classes of $\equiv_L$.

In the following a sufficient condition under which a language gives rise to the same derivative with respect to two different words will be given.

Theorem 4.13 *Let L be a regular language accepted by the deterministic automaton $A = (S, \Sigma, s_0, F, P)$ and u, v words over Σ^*. If $f_u = f_v$ then $\partial_u L = \partial_v L$.*

Proof. Let α be a word in $\partial_u L$. There exist $\alpha_1, \alpha_2 \in \Sigma^*$ and $w \in L$ such that $\alpha = \alpha_1\alpha_2$ and $w = \alpha_1 u \alpha_2$. Consequently, the derivation:

$$s_0\alpha_1 u\alpha_2 \Longrightarrow^* s_1 u\alpha_2 \Longrightarrow^* s_2\alpha_2 \Longrightarrow^* s_f,\ s_f \in F,$$

exists in the automaton A.

According to the definition of $f_u : S \longrightarrow S$, we have $s_2 = f_u(s_1)$. As $f_u = f_v$ it follows that $s_2 = f_v(s_1)$ and consequently the derivation $s_1 v \Longrightarrow^* s_2$ exists in A. Therefore the following derivation can be constructed in A:

$$s_0\alpha_1 v\alpha_2 \Longrightarrow^* s_1 v\alpha_2 \Longrightarrow^* s_2\alpha_2 \Longrightarrow^* s_f,\ s_f \in F.$$

We have used the derivation $s_0\alpha_1 u\alpha_2 \Longrightarrow^* s_f$ in which the subderivation $s_1 u \Longrightarrow^* s_2$ has been replaced by $s_1 v \Longrightarrow^* s_2$. This proves that the word $\alpha_1 v \alpha_2$ belongs to L which implies that $\alpha_1\alpha_2$ belongs to $\partial_v L$.

The reverse inclusion can be proved similarily. □

The converse of the theorem does not hold, as shown by the following example.

Example 4.4 Let $L = \{ab, ca\}$ and let $A = (S, \Sigma, s_0, F, P)$ be a finite deterministic automaton that accepts it, where:

$$\begin{aligned}
S = &\ \{s_0, s_1, s_2, s_3, s_4, s'\},\\
F = &\ \{s_2, s_4\},\\
\Sigma = &\ \{a, b, c\},\\
P = &\ \{s_0 a \longrightarrow s_1, s_0 b \longrightarrow s', s_0 c \longrightarrow s_3\} \cup\\
&\ \{s_1 a \longrightarrow s', s_1 b \longrightarrow s_2, s_1 c \longrightarrow s'\} \cup\\
&\ \{s_2 a \longrightarrow s', s_2 b \longrightarrow s', s_2 c \longrightarrow s'\} \cup\\
&\ \{s_3 a \longrightarrow s_4, s_3 b \longrightarrow s', s_3 c \longrightarrow s'\} \cup\\
&\ \{s_4 a \longrightarrow s', s_4 b \longrightarrow s', s_4 c \longrightarrow s'\} \cup\\
&\ \{s' a \longrightarrow s', s' b \longrightarrow s', s' c \longrightarrow s'\}.
\end{aligned}$$

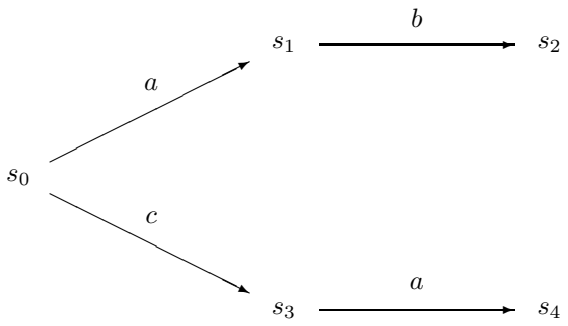

Figure 1: The automaton from Example 4.4.

The automaton A is represented in Figure 1. The state s' is a "garbage" state, introduced only to make the automaton deterministic. It has been omitted from the figure, for reasons of clarity.

The derivative of L with respect to both b and c equals $\{a\}$ but the functions f_b and f_c are not equal: $f_b(s_1) = s_2$ whereas $f_c(s_1) = s'$. □

Corollary 4.11 *A regular language L has at most n^n different derivatives, where n is number of states in a minimal finite deterministic automaton accepting L.*

Proof. Let L be a regular language accepted by a minimal finite deterministic automaton $A = (S, \Sigma, s_0, F, P)$ with n states. The number of different total functions $f : S \longrightarrow S$ is $k = n^n$. Using the previous theorem we deduce that there exist at most k different derivatives of L. □

Example 4.5 Let us consider the minimal finite deterministic automaton $A = (S, \Sigma, s_0, F, P)$ where

$$\begin{aligned} S = &\ \{s_0, s_1\}, \\ \Sigma = &\ \{a_1, a_2, a_3, a_4\}, \\ F = &\ \{s_0\}, \\ P = &\ \{s_0 a_1 \longrightarrow s_0, s_1 a_1 \longrightarrow s_1\} \cup \\ &\ \{s_0 a_2 \longrightarrow s_0, s_1 a_2 \longrightarrow s_0\} \cup \\ &\ \{s_0 a_3 \longrightarrow s_1, s_1 a_3 \longrightarrow s_1\} \cup \\ &\ \{s_0 a_4 \longrightarrow s_1, s_1 a_4 \longrightarrow s_0\}. \end{aligned}$$

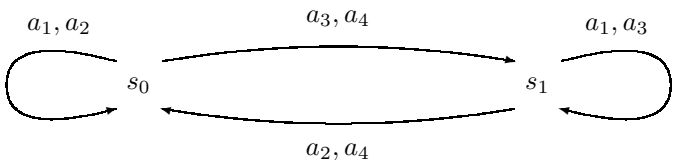

Figure 2: The automaton from Example 4.5.

The automaton A is represented in Figure 2.
The words a_1, a_2, a_3, a_4 determine respectively the functions:

$$\begin{array}{ll} a_1: & f_1(s_0) = s_0, \quad f_1(s_1) = s_1, \\ a_2: & f_2(s_0) = s_0, \quad f_2(s_1) = s_0, \\ a_3: & f_3(s_0) = s_1, \quad f_3(s_1) = s_1, \\ a_4: & f_4(s_0) = s_1, \quad f_4(s_1) = s_0. \end{array}$$

According to the preceding corollary, the maximum number of different derivatives that $L(A) = L$ can have is $\mathrm{card}(S)^{\mathrm{card}(S)} = 4$. In order to show that L has 4 different derivatives we will prove that $\partial_{a_1}L$, $\partial_{a_2}L$, $\partial_{a_3}L$, $\partial_{a_4}L$ are all different.

The word $a_3a_2a_1$ is in L therefore $a_3a_1 \in \partial_{a_2}L$. But a_3a_1 is not in $\partial_{a_1}L$ because neither $a_1a_3a_1$ nor $a_3a_1a_1$ belongs to L. Consequently, $\partial_{a_2}L \neq \partial_{a_1}L$.

The word a_1a_1 belongs to L therefore $a_1 \in \partial_{a_1}L$. However, a_1 is neither in $\partial_{a_3}L$ nor in $\partial_{a_4}L$ as none of the words a_1a_3, a_3a_1, a_1a_4, a_4a_1 is in L. Consequently, $\partial_{a_3}L \neq \partial_{a_1}L$ and $\partial_{a_4}L \neq \partial_{a_1}L$.

The word a_2a_1 belongs to L, therefore $a_1 \in \partial_{a_2}L$. But, as none of the words a_3a_1, a_1a_3, a_4a_1, a_1a_4 is in L, a_1 belongs neither to $\partial_{a_3}L$ nor to $\partial_{a_4}L$. Consequently, $\partial_{a_3}L \neq \partial_{a_2}L$ and $\partial_{a_4}L \neq \partial_{a_2}L$.

The word $a_3a_4a_1$ belongs to L, therefore $a_3a_1 \in \partial_{a_4}L$. But a_3a_1 does not belong to $\partial_{a_3}L$ because neither $a_3a_3a_1$ nor $a_3a_1a_3$ is in L. Consequently, $\partial_{a_3}L \neq \partial_{a_4}L$.

The derivatives $\partial_{a_1}L$, $\partial_{a_2}L$, $\partial_{a_3}L$, $\partial_{a_4}L$ are pairwise distinct. Consequently, the language L has four different derivatives which is the maximal number of derivatives it can have. □

Let $A = (S, \Sigma, s_0, F, P)$ be a finite deterministic automaton. Two states $s, s' \in S$ are called *distinguishable* if there exists a word $u \in \Sigma^*$ such

that $su \Longrightarrow^* s_1$, $s'u \Longrightarrow^* s'_1$ and $s_1 \in F$, $s'_1 \notin F$, or viceversa. A finite deterministic automaton in which all states are distinguishable is minimal for its language (see [5], pp.68-71).

Using the method developed in the previous example we can prove a more general result.

Theorem 4.14 *Let n be a natural number, $n \geq 1$. There exists a minimal finite deterministic automaton A_n, with n states, such that the language accepted by A_n has n^n different derivatives. Moreover, no other language accepted by a minimal finite deterministic automaton with n states has more different derivatives.*

Proof. Let $n \geq 1$ be a natural number and A_n be the automaton:

$$\begin{aligned} A_n = &\quad (S, \Sigma, s_1, F, P), \\ S = &\quad \{s_1, s_2, \ldots, s_n\}, \\ \Sigma = &\quad \{f |\ f : S \longrightarrow S\}, \\ F = &\quad \{s_1\}, \\ P = &\quad \{s_i f \longrightarrow s_j |\ s_i, s_j \in S, f \in \Sigma, \text{ and } f(s_i) = s_j\}. \end{aligned}$$

Clearly, A_n is a finite deterministic automaton. A is also minimal. This follows because any two distinct states s_i and s_j are 1-distinguishable, i.e., a letter distinguishes them. Such a letter is f which, viewed as a function, maps s_i into s_1 and s_j into s_2.

We shall show in the following that the language $L = L(A_n)$ has n^n different derivatives.

If $n = 1$ then $\text{card}(\Sigma) = n^n = 1$. The language accepted by the automaton A_1 is $L = \{f^p |\ p \geq 0\}$ and has one derivative, $\partial_f L = L$.

If $n > 1$, as $\text{card}(\Sigma) = n^n$, the proof is complete if we show that for every $a, b \in \Sigma$, $a \neq b$ we have $\partial_a L \neq \partial_b L$. Let a, b be two distinct letters in Σ. One of the following cases holds:

(i) $a(s_1) \neq b(s_1)$.

If this is the case, then either $a(s_1) \neq s_1$ or $b(s_1) \neq s_1$. Assume that the first alternative holds, the other one being similar. Choose $f \in \Sigma$ with the following properties:

$$f(s_1) = s_1, f(a(s_1)) = a(s_1), f(b(s_1)) = s_1.$$

The word bf belongs to L as we can construct the derivation:

$$s_1 bf \Longrightarrow b(s_1)f \Longrightarrow f(b(s_1)) = s_1,$$

according to A_n. Consequently, f is a word in $\partial_b L$. However, f does not belong to $\partial_a L$ as neither af nor fa belong to L:

$$s_1 af \Longrightarrow a(s_1)f \Longrightarrow f(a(s_1)) = a(s_1) \neq s_1,$$
$$s_1 fa \Longrightarrow f(s_1)a = s_1 a \Longrightarrow a(s_1) \neq s_1.$$

Consequently, if (i) holds then $\partial_a L \neq \partial_b L$.

(ii) $a(s_1) = b(s_1)$ and $a(s_i) \neq b(s_i)$ for some $1 < i \leq n$.

If this is the case then either $a(s_i) \neq s_1$ or $b(s_i) \neq s_1$. Assume that $a(s_i) \neq s_1$, the other alternative being similar. We now consider two subcases.

$(ii)'$ $b(s_i) \neq s_i$.

Choose $f, g \in \Sigma$ with the properties:

$$f(x) = s_i,\ \forall x \in S,$$
$$g(x) = s_i,\ \forall x \in S, x \neq b(s_i) \text{ and } g(b(s_i)) = s_1.$$

The word fbg belongs to L as we can construct the derivation:

$$s_1 fbg \Longrightarrow f(s_1)bg = s_i bg \Longrightarrow b(s_i)g \Longrightarrow g(b(s_i)) = s_1,$$

according to A_n. This implies that fg belongs to $\partial_b L$. However, fg is not a word in $\partial_a L$ as none of the words afg, fag, fga is in L:

$$s_1 afg \Longrightarrow a(s_1)fg \Longrightarrow f(a(s_1))g = s_i g \Longrightarrow g(s_i) = s_i \neq s_1,$$

(we have used the fact that $b(s_i) \neq s_i$)

$$s_1 fag \Longrightarrow f(s_1)ag = s_i ag \Longrightarrow a(s_i)g \Longrightarrow g(a(s_i)) = s_i \neq s_1,$$

(we have used the fact that $a(s_i) \neq b(s_i)$)

$$s_1 fga \Longrightarrow f(s_1)ga = s_i ga \Longrightarrow g(s_i)a = s_i a \Longrightarrow a(s_i) \neq s_1,$$

(we have used the facts that $b(s_i) \neq s_i$ and $a(s_i) \neq s_1$).

Consequently, if $(ii)'$ holds then $\partial_a L \neq \partial_b L$.

$(ii)''$ $b(s_i) = s_i$.

As $s_i \neq s_1$, the above equality implies $b(s_i) \neq s_1$. As $a(s_i) \neq b(s_i)$ and $b(s_i) = s_i$ we deduce that $a(s_i) \neq s_i$. Therefore we are now in the case $b(s_i) \neq s_1$ and $a(s_i) \neq s_i$, which is symmetric to $(ii)'$ (with a and b switching their roles). Consequently, also if this case holds we obtain $\partial_a L \neq \partial_b L$.

As, in all the possible cases we found that $\partial_a L \neq \partial_b L$, we deduce that the two derivatives are distinct. The two letters a, b were arbitrarily chosen from Σ, therefore we conclude that all the n^n elements of Σ produce derivatives which are pairwise distinct. Consequently, $L = L(A_n)$ has n^n diffferent derivatives. The second claim of the theorem follows from Corollary 4.11. □

The following theorem shows that the language consisting of the words with respect to which a given regular language has the same derivative is regular.

Theorem 4.15 *Let L be a regular language over the alphabet Σ. For any word $w \in \Sigma^*$ the language:*

$$L_w = \{v \in \Sigma^* | \ \partial_w L = \partial_v L\}$$

is regular and can be effectively constructed.

Proof. Let $A = (S, \Sigma, s_0, F, P)$ be a finite deterministic automaton that accepts the language L.

For every state $s \in S$ and every function $f : S \longrightarrow S$ define:

$$L_{s,f} = \{w \in \Sigma^* | \ sw \Longrightarrow^* f(s)\}$$

and

$$L_f = \bigcap\nolimits_{s \in S} L_{s,f}.$$

Each of the languages $L_{s,f}$ is regular and each L_f is regular as a finite intersection of regular languages.

Claim.

$$L_f = \{w \in \Sigma^* | \ f_w = f\}.$$

" $\subseteq$ " Let w be a word in L_f. As $w \in L_{s,f}$ for every state $s \in S$, the derivation $sw \Longrightarrow^* f(s)$ exists in the automaton A for every $s \in S$.

According to the definition of $f_w : S \longrightarrow S$, the derivation $sw \Longrightarrow^* f_w(s)$ exists in A for every $s \in S$. As the automaton A is a deterministic one, we deduce that $f(s) = f_w(s)$ for every state $s \in S$, that is, $f = f_w$.

" $\supseteq$ " Let $w \in \Sigma^*$ be a word such that $f(s) = f_w(s)$ for every $s \in S$. Then, for every state $s \in S$ we have:

$$w \in L_{s,f} = \{w \in \Sigma^* | \ sw \Longrightarrow^* f(s) = f_w(s)\}$$

that is,

$$w \in \bigcap\nolimits_{s\in S} L_{s,f} = L_f,$$

and the equality is proved.

The claim shows that the family $\{L_f\}_{f:S\longrightarrow S}$ determines a finite partition of Σ^* into disjoint regular languages L_f. To a set L_f belong those and only those words w such that $f_w = f$. To prove the theorem we show that for a given $w \in \Sigma^*$, L_w is a union of some of the languages L_f.

There exist $k = \text{card}(S)^{\text{card}(S)}$ different functions $f : S\longrightarrow S$. Given a word $w \in \Sigma^*$ we construct:

$$L' = \bigcup\nolimits_{i=1}^{k} \quad \{L_{f_i} |\ f_i : S\longrightarrow S \text{ and } \exists\ v \in L_{f_i} :\ \partial_v L = \partial_w L\}.$$

The language L' is nonempty, containing at least the word w. We will prove in the following that $L' = L_w$ where L_w is the language mentioned in the statement of the theorem.

Indeed, let $u \in L'$. There exist $i \leq k$ and $f_i : S\longrightarrow S$ such that $u \in L_{f_i}$ and $\partial_v L = \partial_w L$ for some $v \in L_{f_i}$. According to the previous claim, $f_u = f_v = f_i$.

According to Theorem 4.13, $\partial_u L = \partial_v L$ and therefore $\partial_u L = \partial_v L = \partial_w L$. This implies that u belongs to L_w.

For the reverse inclusion, let u be a word in L_w. There exists $i \leq k$ such that the function $f_u : S\longrightarrow S$ equals the function $f_i : S\longrightarrow S$. As, according to the definition of L_w, $\partial_u L = \partial_w L$ it follows that u belongs to the set

$$\{L_{f_i} |\ f_i : S\longrightarrow S \text{ and } \exists u \in L_{f_i} : \partial_u L = \partial_w L\}$$

that is u belongs to L'. The equality $L' = L_w$ is therefore proved. As L' is a regular language it follows that L_w is a regular language.

Using the above equality, for every word w the language L_w can be effectively constructed. Indeed, the sets $L_{s,f}$, L_f can be constructed for every $f : S\longrightarrow S$ and every state $s \in S$.

In order to construct L' we use the remark that, for a total function $f_i : S\longrightarrow S$, all the words in L_{f_i} give the same derivative with respect to L. This means that, for any function $f_i : S\longrightarrow S$ it suffices to check the equality $\partial_v L = \partial_w L$ for an arbitrary word $v \in L_{f_i}$. If the answer is YES, the set L_{f_i} is taken into the union, else the function f_{i+1} is tried. The process terminates as the number of different functions to be checked is finite.

Note that REG is closed under sequential deletion (see Corollary 3.3) and therefore under derivative. The equivalence problem is decidable for regular languages that is, the problem "Is $\partial_v L$ equal with $\partial_w L$?" is decidable for regular languages L. □

Corollary 4.12 *Let L be a regular language over an alphabet Σ. For any word $w \in \Sigma^*$ the languages:*

$$L_w^l = \{v \in \Sigma^* |\ \partial_w^l L = \partial_v^l L\},$$

$$L_w^r = \{v \in \Sigma^* |\ \partial_w^r L = \partial_v^r L\}$$

are regular and can be effectively constructed.

Proof. Let L be a language and w be a word over Σ and let \$ be a symbol which does not occur in Σ. Consider $L' = \$L$ (resp. $L\$$) and $w' = \$w$ (resp. $w\$$). Then

$$L_w^l = (\$\backslash\{v \in (\Sigma \cup \{\$\})^* |\ \partial_{w'} L' = \partial_v L'\}) \cap \Sigma^*$$

$$(\text{resp. } L_w^r = (\{v \in (\Sigma \cup \{\$\})^* |\ \partial_{w'} L' = \partial_v L'\}/\$) \cap \Sigma^*).$$

Using the preceding theorem and the fact that REG is closed under left (right) quotient and intersection with regular languages, we deduce that the languages L_w^l and L_w^r are regular and can be effectively constructed. □

We are now in position to settle the closure of the family of context-free languages under dipolar deletion with regular languages.

Theorem 4.16 *The family of context-free languages is closed under dipolar deletion with regular languages.*

Proof. Let L be a context-free and R be a regular language over an alphabet Σ. According to Theorem 3.1 and Corollary 3.2 there exist finitely many languages R_i, $1 \leq i \leq n$, that can be obtained from R by left quotient, and they are regular. Therefore, for each $w \in \Sigma^*$ there exists an index i, $1 \leq i \leq n$, such that $w\backslash R = R_i$.

The preceding Corollary assures that the languages $L_i = \{w |\ w\backslash R = R_i\}$ are regular for all i, $1 \leq i \leq n$.

Claim. $L \rightleftharpoons R = \bigcup_{i=1}^{n} (L_i\backslash L)/R_i$.

"$\subseteq$" Let w be a word in $L \rightleftharpoons R$. There exist $xy \in R$ such that $xwy \in L$. There also exists an index i, $1 \leq i \leq n$, such that $x\backslash R = R_i$. Obviously, $y \in R_i$ and $x \in L_i$. As w belongs to $(x\backslash xwy)/y$, w is an element of the set $(L_i\backslash L)/R_i$.

"$\supseteq$" Let w be a word in $(L_i\backslash L)/R_i$ for some i, $1 \leq i \leq n$. There exist $y \in R_i$ and $x \in L_i$ such that $xwy \in L$. As $x \in L_i$, $x\backslash R = R_i$. Moreover, as $y \in R_i$, it follows that $xy \in R$. This further implies that $w \in L \rightleftharpoons R$, and the proof of the claim is complete.

The theorem now follows as CF is closed under left, right quotient with regular languages and under finite union. □

4.4 The singleton case

The catenation and the right and left quotient of words are deterministic operations in the sense that the result of the operation is, in all three cases, a single word. The sequential insertion and sequential deletion (called in this section shortly insertion and deletion) are nondeterministic versions of catenation respectively right and left quotient. The result of the insertion (deletion) of one word into (from) another is in general a set whose cardinality is greater than one. A natural problem that arises is under what circumstances the insertion or the deletion of two words is deterministic, that is, produces as result a singleton set.

The structural property of words which influences the answer to this problem is whether or not they are *bordered* (the terminology is due to [14]). Before this, the notion of a primitive word is introduced.

Definition 4.3 *A word $u \in \Sigma^+$ is called a primitive word if $u = g^i$, $g \in \Sigma^+$, $i \geq 1$, implies that $i = 1$.*

Every word in Σ^+ can be expressed uniquely as a power of a primitive word (see [8], [14], p.7).

Definition 4.4 *A word $u \in \Sigma^+$ is called bordered if $u = xy = yx'$ for some $x, y, x' \in \Sigma^+$.*

A word which is not bordered will be called *unbordered*. Clearly, an unbordered word is primitive. Thus the set of unbordered words is a proper subfamily of the set of primitive words.

Example 4.6 The following words over $\Sigma = \{a, b\}$ are bordered: aba, $ababab$, $ababa$. The words aab, abb, a^2b^2 are unbordered. □

The following lemmas (see [14], pp.6-11) will be needed in the sequel:

Lemma 4.6 *Let x, y be words in Σ^* such that $xy \neq \lambda$. If $xy = yx$ then there uniquely exist a primitive word $g \in \Sigma^+$ and naturals, $i, j \geq 0$, $i+j > 0$, with the property $x = g^i$, $y = g^j$.*

Lemma 4.7 *If $g \in \Sigma^+$ is a primitive word such that $g = xy = yx$ for some $x, y \in \Sigma^*$, then $x = \lambda$ or $y = \lambda$.*

For a bordered primitive word we have the following property (see [15]):

Lemma 4.8 *Let u be a bordered primitive word in Σ^+. Then u can be expressed as $u = xyx$ for some $x, y \in \Sigma^+$.*

The following two theorems give necessary and sufficient conditions under which the result of the deletion of a word from another is a singleton.

Note. Let u, w be words in Σ^*. If $u = \lambda$ then $w \longrightarrow u$ is a singleton, namely $\{w\}$. If $w = \lambda$ then $w \longrightarrow u$ is a singleton iff $u = \lambda$. Therefore we will deal in the following only with the case where u and w are nonempty words.

Theorem 4.17 *If w, u are words in Σ^+ and u is a power of an unbordered word $g \in \Sigma^+$, $u = g^i$, $i \geq 1$, then the statements (a) and (b) are equivalent:*

(a) The set $w \longrightarrow u$ is a singleton;

(b) The word w is of the form $w = \alpha g^j \beta$, $j \geq i$, $\alpha, \beta \in \Sigma^*$, where neither α nor β contains u as a subword.

Proof. $(a) \Longrightarrow (b)$ Let us assume that $w, u \in \Sigma^+$ as in the theorem.If $w \longrightarrow u$ is a singleton, for any two decompositions of w as $w = xuy = euf$ with $x, y, e, f \in \Sigma^*$, we have $xy = ef$. Let us choose x, y, e, f in such a way that the two occurrences of u are the rightmost and the leftmost one. Consider now all the possible relative positions of x, y and e, f.

- If $\lg(eu) \leq \lg(x)$ then:

$$w = eu \underbrace{x_2 uy}_{f} = \underbrace{eux_2}_{x} uy, \ x_2 \in \Sigma^*.$$

The equality $xy = ef$ implies in this case that $eux_2y = ex_2uy$ that is, $ux_2 = x_2u$. According to Lemma 4.6, x_2 and u are powers of the same

primitive word. As $u = g^i$, g primitive, we deduce that $x_2 = g^k$, $k \geq 0$. The word w can be then written as

$$w = eg^i g^k g^i y = eg^{k+2i} y.$$

Taking now $\alpha = e$ and $\beta = y$, (b) holds.

• If $\lg(e) < \lg(x) < \lg(eu)$ then:

$$w = e \underbrace{u_1 u_2}_{u} \underbrace{u_3 y}_{f} = \underbrace{eu_1}_{x} \underbrace{u_2 u_3}_{u} y,\ u_1, u_2, u_3, \in \Sigma^+.$$

The equality $xy = ef$ implies $eu_1 y = eu_3 y$, that is, $u_1 = u_3$. As $u = u_1 u_2 = u_2 u_1$, according to Lemma 4.6 we obtain that u_1 and u_2 are powers of the same primitive word, which is g. Therefore $u_1 = g^k$, $u_2 = g^{i-k}$, $k > 0$, which implies:

$$w = eu_1 u_2 u_1 y = eg^{i+k} y, k > 0.$$

Taking now $\alpha = e$ and $\beta = y$, (b) holds.

• If $\lg(e) = \lg(x)$ then there is only one occurrence of the word u in w and (b) obviously holds.

For $(b) \Longrightarrow (a)$ assume that $w, u \in \Sigma^+$ are as in the theorem and that (b) holds. We can assume that j is maximal, that is, α does not have g as a suffix and β does not have g as a prefix. As $j \geq i$ there exists a $k \geq 0$ such that $j = i + k$. Argue indirectly and assume that $w \longrightarrow u$ is not a singleton, that is, there exists a word in $w \longrightarrow u$ which is differs from $\alpha g^k \beta$.

Remark. Because u is a power of the unbordered word g, two occurrences of u can overlap only on powers of g.

We shall consider in the following all the possible cases $w = \alpha u g^k \beta = xuy$, $x, y \in \Sigma^*$, which can lead to the situation that $xy \neq \alpha g^k \beta$.

• If $\lg(\alpha u) \leq \lg(x)$ then

$$w = \underbrace{\alpha u x_1}_{x} u \underbrace{y_1 \beta}_{y} = \alpha u \underbrace{x_1 u y_1}_{g^k} \beta,\ x_1, y_1 \in \Sigma^*.$$

Note that u cannot overlap with α or β because g is unbordered, j is maximal and u is a subword of neither α nor β.

As we have assumed that $xy \neq \alpha g^k \beta$ we have that $\alpha u x_1 y_1 \beta \neq \alpha g^k \beta$ which implies that $g^i x_1 y_1 \neq g^k$. As $g^k = x_1 g^i y_1$, this is a contradiction. Our assumption was false, therefore this case cannot hold.

- If $\lg(\alpha) < \lg(x) < \lg(\alpha u)$ then:

$$w = \underbrace{\alpha x_1}_{x}\underbrace{u_1 u_2}_{u}\underbrace{y_1\beta}_{y} = \alpha\underbrace{x_1 u_1}_{u}\underbrace{u_2 y_1}_{g^k}\beta,\ x_1, u_1 \in \Sigma^+, u_2, y_1 \in \Sigma^*.$$

As $u = x_1u_1 = u_1u_2 = g^i$ and g is an unbordered word we have that $u_1 = g^{i_1}$, $u_2 = g^{i_2} = x_1$, $i_1, i_2 > 0$. The fact that $xy \neq \alpha g^k \beta$ implies $\alpha x_1 g^{k-i_2}\beta \neq \alpha g^k \beta$ that is, $x_1 g^{k-i_2} \neq g^k$. As we have shown that $x_1 = g^{i_2}$, this is a contradiction. Our assumption was false, therefore this case cannot hold either.

As all the possible cases led to contradictions, our assumption that $w \longrightarrow u$ is not a singleton is false. The proof of the second implication, and therefore of the theorem, is complete. □

The proof of the implication $(a) \Longrightarrow (b)$ did not use the fact that g is an unbordered word.

The reverse implication does not hold if g is not unbordered. For example, if $w = ababa$ and $u = aba$, taking $\alpha = ab$, $\beta = \lambda$, $g = aba$, the condition (b) is satisfied. However the set $w \longrightarrow u = \{ab, ba\}$ is not singleton. A stronger condition than (b) is needed to assure that $w \longrightarrow u$ is a singleton, if u is a power of a primitive bordered word.

Theorem 4.18 *Let w, u be words in Σ^+. If u is a power of a primitive bordered word $g \in \Sigma^+$, $u = g^i$, $i \geq 1$, then the statements (a) and (b) are equivalent:*

(a) The set $w \longrightarrow u$ is a singleton.

(b) The word w is of the form $w = \alpha g^j \beta$, $j \geq i$, $\alpha, \beta \in \Sigma^*$, where

(1) Neither α nor β contains u as a subword,

(2) For any decomposition of g, $g = xy = yx'$ where $x, y, x' \in \Sigma^+$ we have: $\alpha \neq \alpha' g^{i-1} x$, $\forall \alpha' \in \Sigma^*$ and $\beta \neq x' g^{i-1} \beta'$, $\forall \beta' \in \Sigma^*$.

Proof. $(a) \Longrightarrow (b)$ Let w, u be as in the theorem.

If $w \longrightarrow u$ is a singleton, using the proof of Theorem 4.17 and the remark following it, $(b)(1)$ holds. Therefore w is of the form $w = \alpha g^j \beta$, $j \geq i$. As $j \geq i$ there exists $k \geq 0$ such that $j = i + k$.

Argue indirectly and assume that $(b)(2)$ does not hold. This means that one of the following cases holds:

• $\alpha = \alpha' g^{i-1} x$ where $\alpha' \in \Sigma^*$, $x \in \Sigma^+$ and there exists $y, x' \in \Sigma^+$ such that $g = xy = yx'$,

• $\beta = x' g^{i-1} \beta'$ where $\beta' \in \Sigma^*$, $x' \in \Sigma^+$ and there exist $y, x \in \Sigma^+$ such that $g = xy = yx'$.

We shall consider the first case, the other one being symmetric. The word w can be written as:

$$w = \alpha' g^{i-1} x g^{i+k} \beta = \alpha' g^{i-1} x (yx')^{i+k} \beta = \alpha' \underbrace{g^{i-1} xy}_{g^i = u} x' g^{i+k-1} \beta.$$

As $w \longrightarrow u$ is a singleton the words $\alpha' x' g^{i+k-1} \beta$ and $\alpha' g^{i-1} x g^k \beta$ are equal. This equality leads to the following chain of implications:

$$x' g^{i+k-1} = g^{i-1} x g^k \Longrightarrow x' g^{i-1} = g^{i-1} x \Longrightarrow$$

$$x' \underbrace{yx' \ldots yx'}_{i-1} = \underbrace{xy \ldots xy}_{i-1} x \text{ and, as } \lg(x') = \lg(x), \Longrightarrow$$

$$x = x' \Longrightarrow g = xy = yx$$

According to Lemma 4.7, the last equality implies that either x or y equals λ. This contradicts our assumption that $x, y \in \Sigma^+$.

All the possible cases led to contradiction and therefore our assumption that $(b)(2)$ does not hold was false.

For the implication $(b) \Longrightarrow (a)$, let w, u be words in Σ^+, satisfying (b). Therefore u and w are nonempty words, $w = \alpha g^j \beta$, $u = g^i$, $j \geq i \geq 1$, ($g \in \Sigma^+$ primitive and bordered) such that $(b)(1)$ and $(b)(2)$ hold. We can assume that j is maximal, that is, g is neither a suffix of α nor a prefix of β. As $j \geq i$ there exists $k \geq 0$ such that $j = i + k$.

From (b) and the fact that j is maximal it follows that an arbitrary occurrence of u in w overlaps with neither α nor β. Assume, for example, that u overlaps with α. Then we have:

$$w = \underbrace{\alpha_1 u_1}_{\alpha} \underbrace{u_2 u_3}_{u} g^k \beta, \alpha_1 \in \Sigma^*, u_1, u_2, u_3 \in \Sigma^+, u = u_1 u_2 = u_2 u_3.$$

As $u = u_1 u_2 = u_2 u_3$, if any of u_i, $i = 1, 2, 3$ would be a power of g then u_1 would equal u_3. This, in turn, would imply that α contains g as a suffix – a contradiction with the maximality of j. Therefore we can assume that none of u_i, $i = 1, 2, 3$ is a power of g and we have:

$$u = \underbrace{g^q g_1}_{u_1} \underbrace{g_2 g^p}_{u_2} = \underbrace{g_2 g^p}_{u_2} u_3, q + p + 1 = i, g_1, g_2 \in \Sigma^+, g = g_1 g_2.$$

If $p > 0$ and $q = 0$ then the preceding equality implies:

$$u = g_1g_2(g_1g_2)^p = g_2(g_1g_2)^p u_3,$$

and as $\lg(g_1g_2) = \lg(g_2g_1)$ we conclude that $g = g_1g_2 = g_2g_1$. According to Lemma 4.7 this implies $g_1 = \lambda$ or $g_2 = \lambda$– a contradiction with our assumption $g_1, g_2 \in \Sigma^+$.

If $p > 0$ and $q > 0$ then:

$$u = \underbrace{g_1g_2 \ldots g_1g_2}_{q \text{ times}} g_1g_2(g_1g_2)^p = g_2(g_1g_2)^p u_3,$$

which implies that $g_1g_2 = g_2g_1$ and leads to the same contradiction.

If $p = 0$ then $\underbrace{g^q g_1}_{u_1}\underbrace{g_2}_{u_2} = g_2u_3$. As $q = i-1$ we obtain that $\alpha = \alpha_1 g^{i-1}g_1$ where $g = g_1g_2 = g_2u_3$, and $g_1, g_2, u_3 \in \Sigma^+$, which contradicts $(b)(2)$.

As all cases led to contradictions, our assumption that an occurrence of u in w can overlap with α was false. Similarly we can prove that no occurrence of u in w overlaps with β.

As u can overlap with neither α nor β, an occurrence of u in w can appear only in the "g-part" of w. This means that an arbitrary occurrence of u in w can have only one of the following locations:

- $w = \alpha g^{k_1-1} g_1 \underbrace{g_2 g^{i-1} g_1}_{u} g_2 g^{k_2}\beta,\ g_1, g_2 \in \Sigma^+, g_1g_2 = g,$

where $k > 0$ and $k_1 + k_2 = k$. We have assumed here that $k_1 > 0$ and $k_2 \geq 0$. The case when $k_2 > 0$ and $k_1 \geq 0$ is similar.

As $u = g^i = (g_1g_2)^i = g_2(g_1g_2)^{i-1}g_1$ we deduce that $g = g_1g_2 = g_2g_1$ which, together with Lemma 4.7, leads to a contradiction with our assumption that $g_1,\ g_2 \in \Sigma^+$. We conclude that such a situation cannot occur.

- $w = \alpha g^{k_1} \underbrace{g^i}_{u} g^{k_2}, k \geq 0, k_1 + k_2 = k.$

In this situation, the erasing of u from w produces the word $\alpha g^k \beta$, regardless of the values of k_1 and k_2, $k_1 + k_2 = k$.

We conclude that the only possible occurrence of u in w yields $w \longrightarrow u = \{\alpha g^k \beta\}$. Therefore $w \longrightarrow u$ is a singleton, that is, (a) holds. This completes the proof of the second implication, and therefore of the theorem. □

The following theorem gives a necessary and sufficient condition under which the result of the insertion between two words is a singleton set.

Theorem 4.19 *Let u, w be words in Σ^*. The set $w \leftarrow u$ is a singleton iff one of the following cases holds:*

(i) The words w, u have the forms $w = a^p$, $u = a^i$, $a \in \Sigma$, $p, i > 0$;

(ii) Either u or w (or both) is equal with λ.

Proof. The "if"-part is obvious. For the "only if"-part let u, w be in Σ^+ such that $w \leftarrow u$ is a singleton. We will show that in this case (i) holds. The fact that $w \leftarrow u$ is a singleton implies that for any decomposition of w as $w = xy$, $x, y \in \Sigma^*$ we have that $xuy = uxy = xyu$, all being elements of the set $w \leftarrow u$. From the equality $xuy = uxy$ and using Lemma 4.6, we deduce that x and u are powers of the same primitive word, $x = g^j$, $u = g^i$, $g \in \Sigma^+$, $j \geq 0$, $i > 0$. Analogously, from $xuy = xyu$ we deduce that $y = g^k$, $k \geq 0$, being a power of the same primitive word as u. As x, y were arbitrary words with the property $xy = w$, taking for example x the first letter of w we conclude that u is of the form $u = a^i$, $a \in \Sigma$, $i > 0$ and w is of the form $w = xy = a^{j+k}$, $j \geq 0$, $k \geq 0$, $j + k > 0$. Taking $p = j + k$ the proof of the "only if"-part is complete. □

The catenation and the right and left quotient of words possess the property that given the result of the operation and one of the operands, the other operand can be recovered. Indeed, if x, y, z are words in Σ^* then $xy = z$ iff $x = z/y$ iff $y = x\backslash z$. The insertion and deletion of words do not have this property. In general, if $x \leftarrow y = z$ then $\{x\} \subseteq z \rightarrow y$ and if $x \rightarrow y = z$ then $\{x\} \subseteq z \leftarrow y$, but the reverse inclusions do not hold. The following theorems will deal with circumstances under which, given the result of the insertion and the inserted word, the other operand can be obtained. The problem can be stated shortly : "When is $(w \leftarrow u) \rightarrow u$ equal with $\{w\}$?", where $u, w \in \Sigma^*$. Besides the fact that u is a power of a primitive bordered or unbordered word, the answer to this problem is influenced by whether or not u is a subword of w.

Note. If $u = \lambda$ or $w = \lambda$ then $(w \leftarrow u) \rightarrow u = \{w\}$. Therefore we will consider in the following only the case where u and w are nonempty words.

Theorem 4.20 *Let u, w be two words in Σ^+ such that u is not a subword of w. If u is a power of an unbordered word then $(w \leftarrow u) \rightarrow u = \{w\}$.*

Proof. Let u, w be as in the theorem, such that $u = g^i$, $g \in \Sigma^+$, $i \geq 1$, and g is an unbordered word. Let xuy be an arbitrary word in $(w \leftarrow u)$, where $x, y \in \Sigma^*$, $w = xy$.

If the only occurrence of u in xuy is the one inserted, then $xuy \longrightarrow u = xy = \{w\}$.

Else, the second occurrence of u must overlap the first, as we have assumed that u is not a subword of w. Moreover, because u is a power of an unbordered word g, they must overlap on powers of g. Under these circumstances, the erasing of the second ocurrence of u from xuy produces also w.

In all the possible cases the erasing of an occurrence of u from an arbitrary word of $(w \longleftarrow u)$ produced w, and therefore we can conclude that $(w \longleftarrow u) \longrightarrow u = \{w\}$. □

The reverse implication does not hold. For example, taking $w = cd$, $u = aba$, we have that u is not a subword of w and that $(w \longleftarrow u) \longrightarrow u = \{w\}$ but u is not a power of an unbordered word.

Theorem 4.21 *Let u, w be words in Σ^+ such that u is not a subword of w. If u is a power of a primitive bordered word $g \in \Sigma^+$, $u = g^i$, $i \geq 1$ then the following statements are equivalent:*

(i) The set $(w \longleftarrow u) \longrightarrow u$ is a singleton, namely $\{w\}$.

(ii) For any decomposition of g, $g = xy = yx'$, $x, y, x' \in \Sigma^+$, the word w contains neither $g^{i-1}x$ nor $x'g^{i-1}$ as a subword.

Proof. We will prove first that $\neg(ii) \Longrightarrow \neg(i)$. Let u, w be as in the theorem such that (ii) does not hold. There exists a decomposition of g, $g = xy = yx'$ where $x, y, x' \in \Sigma^+$ such that $w = \alpha g^{i-1} x \beta$, $\alpha, \beta \in \Sigma^*$. The case where w is of the form $w = \alpha x' g^{i-1} \beta$ is symmetric. The word

$$\alpha g^{i-1} x g^i \beta = \alpha \underbrace{g^{i-1} xy}_{u} x' (yx')^{i-1} \beta$$

belongs to $w \longleftarrow u$ and therefore both words $\alpha g^{i-1} x \beta$ and $\alpha x' g^{i-1} \beta$ are in the set $(w \longleftarrow u) \longrightarrow u$.

If we assume that $(w \longleftarrow u) \longrightarrow u$ is a singleton, we obtain $g^{i-1}x = x'g^{i-1}$ which implies

$$\underbrace{xyxy \ldots xy}_{(i-1) \text{ times}} x = x' \underbrace{yx' \ldots yx'}_{(i-1) \text{ times}} .$$

The last equality shows that $x = x'$, which implies $g = xy = yx$. According to Lemma 4.7 either x or y equals λ, which contradicts our assumption $x, y \in \Sigma^+$. Consequently, we conclude that $(w \longleftarrow u) \longrightarrow u$ is not a singleton.

The implication $(ii) \Longrightarrow (i)$ follows by using Theorem 4.18. Indeed, let w_1 be a word in $w \leftarrow u$. Then w_1 is of the form $\alpha g^i \beta$, where g is primitive and bordered, and $\alpha\beta = w$. From the fact that u is not a subword of w we conclude that the condition $(b)(1)$ of Theorem 4.18 is satisfied. From (ii) we deduce that also $(b)(2)$ holds. Consequently, we can apply Theorem 4.18 which assures that $w_1 \rightarrow u$ is a singleton. As $w \in w_1 \rightarrow u$, it follows that $w_1 \rightarrow u = w$. As w_1 was an arbitrary word from $w \leftarrow u$, we conclude that $(w \leftarrow u) \rightarrow u = w$. □

Theorem 4.22 *Let u, w be words in Σ^+, u a proper subword of w. Then $(w \leftarrow u) \rightarrow u = \{w\}$ iff $w = a^p$, $u = a^i$, $a \in \Sigma$, $p > i > 0$.*

Proof. The implication " $\Longleftarrow$ " is obvious. In order to show the reverse implication, let u, w be words in Σ^+ where u a is subword of w (not necessarily proper) and $(w \leftarrow u) \rightarrow u = \{w\}$. The word w can be expressed as $w = xuy$, $x, y \in \Sigma^*$, $u \in \Sigma^+$. This implies that both words $u(xuy)$ and $(xuy)u$ belong to $w \leftarrow u$ and therefore:

$$xuy, uxy, xyu \ \in (w \leftarrow u) \rightarrow u = \{w\}.$$

From the equality $xuy = uxy$ we deduce $xu = ux$. According to Lemma 4.6, x and u are powers of the same primitive word, $u = g^i$, $x = g^k$, $g \in \Sigma^+$, $k \geq 0$, $i \geq 1$.

From the equality $xuy = xyu$ we deduce $uy = yu$. According to Lemma 4.6, y and u are powers of the same primitive word g that is, $y = g^j$, $j \geq 0$.

The primitive word g is unbordered. Indeed, assume that g is bordered. Then, according to Lemma 4.8, g can be written as $g = \gamma v \gamma$, $\gamma, v \in \Sigma^+$. As $u = (\gamma v \gamma)^i$ and $w = (\gamma v \gamma)^{k+i+j}$ we deduce that both words:

$$(\gamma v \gamma)^{k+2i+j}, \text{ and } \gamma(\gamma v \gamma)^i v \gamma (\gamma v \gamma)^{k+i+j-1} = \gamma(\gamma v \gamma)^{i-1} \gamma v (\gamma v \gamma)^i (\gamma v \gamma)^{k+j},$$

are in the set $w \leftarrow u$ (the first word was obtained by catenating w and u and the second by inserting u after the first occurrence of γ.) This implies that both words:

$$(\gamma v \gamma)^{k+i+j} \text{ and } \gamma(\gamma v \gamma)^{i-1} \gamma v (\gamma v \gamma)^{k+j},$$

belong to $(w \leftarrow u) \rightarrow u$, which is a singleton. The equality of the above mentioned words implies the equality of their prefixes $\gamma v \gamma = \gamma\gamma v$ which further implies $v\gamma = \gamma v$. According to Lemma 4.6, γ and v are powers of the same primitive word, $\gamma = \delta^r$, $v = \delta^{r'}$, $\delta \in \Sigma^+$, $r, r' > 0$. We can rewrite

g now as $g = \gamma v \gamma = \delta^{2r+r'}$, $2r + r' > 2$, which contradicts the fact that g is primitive. Our assumption was false, therefore g is an unbordered word.

Taking $p = i + j + k$ we have therefore proved that if u is a subword of w (proper or not) and $(w \leftarrow u) \rightarrow u = \{w\}$ then $u = g^i$, $w = g^p$, $g \in \Sigma^+$, $p \geq i > 0$, where g is an unbordered word.

Assume now that $u \neq \lambda$ is a proper subword of w and denote $k' = j + k$, $k' > 0$. Argue indirectly and assume that g contains at least two different letters, $g = a\alpha b\beta$, $a, b \in \Sigma$, $a \neq b$,$\alpha, \beta \in \Sigma^*$. Then both words:

$$(a\alpha b\beta)^{2i+k'} \text{ and } a\alpha(a\alpha b\beta)^i b\beta(a\alpha b\beta)^{i+k'-1},$$

are in the set $w \leftarrow u$ which implies that both:

$$(a\alpha b\beta)^{i+k'} \text{ and } a\alpha(a\alpha b\beta)^i b\beta(a\alpha b\beta)^{k'-1}$$

belong to $(w \leftarrow u) \rightarrow u$. Indeed, as $k' \geq 1$ we have another occurrence of u in $w \leftarrow u$ than the one inserted, namely the prefix of length $\lg(u)$ of $(a\alpha b\beta)^{i+k'-1}$. As $(w \leftarrow u) \rightarrow u$ is a singleton, the two words belonging to it are equal that is,

$$a\alpha b\beta(a\alpha b\beta)^{i+k'-1} = a\alpha(a\alpha b\beta)^i b\beta(a\alpha b\beta)^{k'-1}.$$

We arrive at a contradiction as, after erasing the prefix $a\alpha$, the above equality implies $a = b$ and we assumed that the letters a and b are distinct. Our assumption that g contains at least two different letters was false. As g is also primitive and unbordered we deduce that g is of the form $g = a$, $a \in \Sigma$ and consequently, $w = a^{i+k'}$, $i \geq 1$, $k' > 0$.

Taking $p = i + k'$, the proof of the second implication is complete. □

Theorem 4.23 *If u is a word in Σ^+ then $(u \leftarrow u) \rightarrow u = \{u\}$ iff u is a power of an unbordered word.*

Proof. It has been shown in the proof of Theorem 4.22 that, if $u, w \in \Sigma^+$ and u is a subword of w (proper or not) then $(w \leftarrow u) \rightarrow u = \{w\}$ implies $u = g^i$, $w = g^p$, $p \geq i > 0$, where $g \in \Sigma^+$ is an unbordered word. Taking $u = w$, this proves the implication "$\Longrightarrow$" of the theorem.

For the reverse implication let $u \in \Sigma^+$ be a power of an unbordered word $g \in \Sigma^+$, $u = g^i$, $i \geq 1$.

Assume that there exists $w \in (u \leftarrow u) \rightarrow u$, $w \neq u$. Applying the operations in the reverse order, we deduce that $u \in (w \leftarrow u) \rightarrow u$. As

$w \neq u$ but $\lg(w) = \lg(u)$, u is not a subword of w. According to Theorem 4.20 we have $(w \leftarrow u) \rightarrow u = \{w\}$, which implies $w = u$. This contradicts our assumption that $w \neq u$. Consequently, we can conclude that the set $(u \leftarrow u) \rightarrow u = \{u\}$, and therefore the proof for the second implication is complete. □

The last theorem of this section gives a necessary and sufficient condition under which the set $(w \rightarrow u) \leftarrow u$ is a singleton.

Theorem 4.24 *If u, w are words in Σ^* then $(w \rightarrow u) \leftarrow u = \{w\}$ iff one of the next cases holds:*

(i) The word w is equal with u;

(ii) The word u equals λ;

(iii) The words w, u are of the form $w = a^p$, $u = a^i$, $a \in \Sigma$, $p > i \geq 1$.

Proof. The implication " $\Longleftarrow$ " is obvious. For the reverse implication, assume that $w, u \in \Sigma^*$, such that $(w \rightarrow u) \leftarrow u = \{w\}$ and $w \neq u$, $u \neq \lambda$. We will show that in this case (iii) holds.

Let $a\alpha$ be a word in $w \rightarrow u$. The equality $au\alpha = ua\alpha$ implies that $u = a^i$, $i > 0$. The equality $a\alpha u = au\alpha$ implies that $w = a^p$, $p > 1$. As u is a proper subword of w, $p > i \geq 1$, and the proof of the second implication is complete. □

Chapter 5

Decidability

5.1 Basic decision problems

In Section 4.2 we investigated the special case where the result of the sequential insertion between two languages is regular. The main results of the section state that, if $L_1 \leftarrow L_2 = R$, for languages $L_1, L_2, R \subseteq \Sigma^*$, $R \in$REG, then either L_1 or L_2 (or both) can be replaced with regular languages yielding the same result, R. A natural problem concerns such situations, that is, when is the result of the insertion of two languages regular.

This section will be concerned with the more general problem, namely the decidability of the questions "Is $L_1 \diamond L_2$ equal with R?" and "Is $L_1 \diamond L_2$ regular?" for regular languages R, regular or context-free languages L_1, L_2, where $\diamond$ is one of the operations defined in the previous sections.

More precisely, for a binary operation $\diamond$ and for given languages L_1, L_2, regular languages R and words w, we consider the problems:
Q_0: "Is $L_1 \diamond L_2 = R$?"
Q_0^w: "Is $L_1 \diamond w = R$?"
Q: "Is $L_1 \diamond L_2$ a regular language?"
Q^w: "Is $L_1 \diamond w$ a regular language?"
For $\diamond$ denoting a controlled operation and for given languages L_1, regular languages R and control functions Δ, the following problems will be considered:
$Q_{0,\Delta}$: "Is $L_1 \diamond \Delta = R$?
Q_Δ: " Is $L_1 \diamond \Delta$ a regular language?"

If we restrict ourselves to control functions having only $\emptyset$ or singletons as their values, the corresponding problems will be denoted respectively by $Q^w_{0,\Delta}$, Q^w_{Δ}.

Theorem 5.1 *Let $\diamond$ be one of the following operations: catenation, SIN, SIN next to a letter, shuffle, PIN, PIN next to a letter, right and left quotient, SD, iterated SD, SD next to a letter, scattered SD, PD, PD next to a letter. Then the problem Q_0 is decidable for regular languages L_1, L_2, R.*

Proof. The family of regular languages is closed under all the above operations and there exist effective procedures for constructing $L_1 \diamond L_2$ from L_1, L_2 given regular languages. Indeed, this follows from the proofs [12] pp.20-22, Theorem 2.3, Theorem 2.18, [4] p.206, Theorem 2.4, Theorem 2.19, [12] p.129 and p.133, Lemma 3.1 and Corollary 3.4, Theorem 3.8 and Corollary 3.12, Theorem 3.20, Theorem 3.25, Theorem 3.3 and Corollary 3.7, Theorem 3.22, respectively. As the equivalence problem is decidable for the family of regular languages, the problem "Is $L_1 \diamond L_2 = R$?" will also be decidable. □

The above proof can be used to show that, for $\diamond$ denoting one of the operations in the preceding theorem, the problem Q^w_0 is decidable for regular languages L_1, R and words w.

The family of regular languages is closed under permuted SIN with singletons, permuted PIN with singletons, permuted scattered SIN with singletons, permuted SD with singletons, permuted PD with singletons and permuted scattered SD with singletons (see Theorems 2.13, 2.23, Theorem 3.11 and the remarks following it, Theorems 3.14, 3.32). Consequently, a similar argument can be used to show that for these operations the problem Q^w_0 is decidable for regular languages L_1 and R.

In the sequel a *singleton* (resp. *regular, context-free*) *control function* will mean a control function whose values are $\emptyset$ or singleton (resp. regular, context-free) languages.

Theorem 5.2 *Let $\diamond$ be one of the operations: controlled SIN, controlled PIN, controlled SD, controlled PD. The problem $Q_{0,\Delta}$ is decidable for regular languages L, R and regular control functions Δ.*

Proof. It follows from the fact that REG is closed under controlled SIN, controlled PIN, controlled SD, controlled PD (Theorems 2.18, 2.19, 3.20, 3.22). The proofs are constructive and the equivalence problem is decidable for the family of regular languages. □

The above proof can be used to show that, for $\diamond$ denoting one of the operations in the preceding theorem, the problem $Q^w_{0,\Delta}$ is decidable for regular languages L, R and singleton control functions Δ.

Theorem 5.3 *Let $\diamond$ be one of the operations: catenation, SIN, iterated SIN, permuted SIN, shuffle, permuted scattered SIN, PIN, iterated PIN, permuted PIN, right and left quotient, SD, iterated SD, permuted SD, scattered SD, permuted scattered SD, PD, iterated PD, permuted PD.*

Then the problem Q_0 is undecidable for context-free languages L_1, L_2 and regular languages R.

Proof. Let Σ be an alphabet with $\text{card}(\Sigma) \geq 2$. There exists a regular language, $R = \Sigma^*$, and a singleton language $L_2 = \{\lambda\}$, such that for any of the above operations, the problem "Is $L_1 \diamond L_2$ equal with R?" is undecidable for context-free languages L_1 over Σ. Indeed, for all the operations listed in the theorem, the problem "Is $L_1 \diamond \{\lambda\}$ equal with Σ^*?" amounts to the problem "Is L_1 equal with Σ^*?" which is undecidable for context-free languages L_1 over Σ.

Note that the result is stronger than the one stated in the theorem. The theorem claims the nonexistence of algorithms dealing with tuples (Σ, L_1, L_2, R). The proof shows the nonexistence of algorithms dealing with L_1 alone. For reasons of uniform presentation, this will often be the case in the sequel: the proofs actually contain more powerful results than stated in the theorems. □

For $\diamond$ denoting one of the operations in the preceding theorem, the problem Q^w_0 is undecidable for context-free languages L_1, regular languages R and words w. Indeed, this follows by noticing that in the above proof the language L_2 is a singleton.

The above theorem did not provide an answer to the question "When is $L_1 \leftarrow L_2 = R$?" for $L_1, L_2, R \subseteq \Sigma^*$, R a regular language and L_1, L_2 context-free (context-sensitive) languages. However, the class of non- regular languages whose insertion is regular is a large one. Indeed, we notice that for any language $L \subseteq \Sigma^*$, the following relation holds:

$$(L \cup \{\lambda\}) \leftarrow (L^c \cup \{\lambda\}) = \Sigma^*.$$

Consequently, any context-free language whose complement is a non-context- free language, produces an example of an insertion whose result is regular but whose left operand is a non-regular context-free language, and right operand is a non-context-free one.

Theorem 5.4 *Let $\diamond$ denote one of the operations: SIN next to a letter, PIN next to a letter, SD next to a letter, PD next to a letter. The problem Q_0 is undecidable for context-free languages L_1, L_2 and regular languages R.*

Proof. Similar to that of the preceding theorem.

Indeed, take $R = \#\Sigma^*$, $\# \notin \Sigma$, and, for every context-free language $L_1 \subseteq \Sigma^*$ take $L_1' = \#L_1$. The problem "Is $L_1' \diamond \{\lambda\} = R$?", where the control letter is $\#$, amounts to the problem "Is $L_1 = \Sigma^*$?". □

The above proof can be used to show that, for $\diamond$ denoting one of the operations in the preceding theorem, the problem Q_0^w is undecidable for context-free languages L_1, regular languages R and words w.

Theorem 5.5 *Let $\diamond$ be one of the operations: controlled SIN, controlled PIN, controlled SD, controlled PD. Then the problem $Q_{0,\Delta}$ is undecidable for context-free languages L_1, regular control functions Δ and regular languages R.*

Proof. Let Σ be an alphabet with $\text{card}(\Sigma) \geq 2$. There exists a regular language $R = \Sigma^+$ and a singleton control function Δ such that for any of the listed operations, the problem "Is $L_1 \diamond \Delta$ equal with R?" is undecidable for context-free languages L_1 over Σ.

Indeed let us choose the control function $\Delta : \Sigma \longrightarrow 2^{\Sigma^*}$, $\Delta(a) = \{\lambda\}$ $\forall a \in \Sigma$. For all the considered operations, the problem "Is $L_1 \diamond \Delta$ equal with Σ^+ ?" amounts to the problem "Is $L_1 - \{\lambda\}$ equal with Σ^+?", which is undecidable for context-free languages L_1 over Σ. We have chosen the language $R = \Sigma^+$ instead of Σ^* because the empty word does not appear in the result of any controlled operation. □

Notice that in the above proof the control function is a singleton control function. Consequently the proof can be used to show that, for $\diamond$ denoting one of the operations in the preceding theorem, the problem $Q_{0,\Delta}^w$ is undecidable for context-free languages L_1, regular languages R and singleton control functions Δ.

Theorem 5.6 *Let $\diamond$ be one of the operations: catenation, SIN, iterated SIN, permuted SIN, shuffle, permuted scattered SIN, PIN, iterated PIN, permuted PIN, right and left quotient, SD, iterated SD, permuted SD, scattered SD, permuted scattered SD, PD, iterated PD, permuted PD. Then the problem Q is undecidable for context-free languages L_1 and regular languages L_2.*

Proof. Let Σ be an alphabet with $\text{card}(\Sigma) \geq 2$. There exists a singleton language $L_2 = \{\lambda\}$ such that for any of the mentioned operations, the problem "Is $L_1 \diamond L_2$ regular?" is undecidable for context-free languages L_1 over Σ.

Indeed, the problem "Is $L_1 \diamond \{\lambda\}$ regular?" amounts, for all the considered operations, to the problem "Is L_1 regular?" which is undecidable for context-free languages L_1 over Σ. □

Notice that in the above proof the language L_2 is a singleton. Consequently, for $\diamond$ denoting one of the operations in the preceding theorem, the problem Q^w is undecidable for context-free languages L_1 and words w.

Theorem 5.7 *Let $\diamond$ denote one of the operations: SIN next to a letter, PIN next to a letter, SD next to a letter, PD next to a letter. The problem Q is undecidable for context-free languages L_1 and regular languages L_2.*

Proof. Analogous to that of the preceding theorem. For every context-free language $L_1 \subseteq \Sigma^*$ take $L_1' = \#L_1$, $\# \notin \Sigma$. Then the problem "Is $L_1' \diamond \{\lambda\}$ regular?", where the control letter is $\#$, amounts to the problem "Is L_1 regular?". □

For $\diamond$ denoting one of the operations in the preceding theorem, the problem Q^w is undecidable for context-free languages L_1 and words w. This follows by noticing that the language $L_2 = \{\lambda\}$ in the above proof is a singleton.

Theorem 5.8 *Let $\diamond$ be one of the operations: controlled SIN, controlled PIN, controlled SD, controlled PD. Then the problem Q_Δ is undecidable for context-free languages L_1 and regular control functions Δ.*

Proof. Let Σ be an alphabet with $\text{card}(\Sigma) \geq 2$. There exists a singleton control function Δ such that, for any of the operations in the theorem, the problem "Is $L_1 \diamond \Delta$ regular?" is undecidable for context-free languages L_1 over Σ. Indeed, let us take $\Delta : \Sigma \longrightarrow 2^{\Sigma^*}$, $\Delta(a) = \{\lambda\}$, $\forall a \in \Sigma$. The problem "Is $L_1 \diamond \Delta$ regular?" amounts, for all the above operations, to the problem "Is $L_1 - \{\lambda\}$ regular?". As the last problem is undecidable for context-free languages $L_1 \subseteq \Sigma^*$, our problem is also undecidable. □

The above proof can be used to show that, for $\diamond$ denoting one of the operations in the preceding theorem, the problem Q_Δ^w is undecidable for context-free languages L_1 and singleton control functions Δ.

5.2 The right operand problem for insertion

The preceding section was concerned with the problem of whether or not given languages L_1, L_2, R (R regular) satisfy the equation $L_1 \diamond L_2 = R$, for various insertion and deletion operations $\diamond$. This section will deal with the question concerning whether or not the equation $L_1 \diamond L_2 = R$ has a solution L_2, where L_1, R are given languages, R a regular one, and $\diamond$ is an insertion operation. Moreover, the existence of a singleton solution, that is, a solution L_2 in the class of singleton languages, will be investigated.

More precisely, for a binary insertion operation $\diamond$ and for given languages L_1 and R, R regular, we consider the problems:
Q_2: "Does there exist a language L_2 such that $L_1 \diamond L_2 = R$?"
Q_2^w: "Does there exist a word w such that $L_1 \diamond w = R$?".
For $\diamond$ denoting a controlled insertion operation and for given languages L_1 and R, R regular, the following problems will be considered:
$Q_{2,\Delta}$: "Does there exist a control function Δ such that $L_1 \diamond \Delta = R$?"
$Q_{2,\Delta}^w$: "Does there exist a singleton control function Δ such that $L_1 \diamond \Delta = R$?".

In the cases where the considered problem is decidable, it will follow from the proof that a solution of the equation can be effectively constructed.

Theorem 5.9 *The problem "Does there exist a language L_2 such that $L_1 L_2 = R$?" is decidable for regular languages L_1 and R.*

Proof. For given regular languages L_1, R over an alphabet Σ define:

$$R' = (L_1 \backslash R^c)^c.$$

It has been proved in Theorem 4.6 that, if there exists $L_2 \subseteq \Sigma^*$ such that $L_1 L_2 = R$, then $L_1 R' = R$. Moreover, the regular language R' can be effectively constructed (see Corollary 4.1).

The algorithm which decides our problem will start with the construction of R'. Then we find out whether or not $L_1 R'$ equals R. □

Example 5.1 Let $L_1 = \{a, ab\}$ and $R = \{ab, abb\}$. We are investigating the existence of a solution L_2 to the equation $L_1 L_2 = R$. Using the method of the preceding theorem we construct the languages:

$$\begin{aligned} R^c &= \{a,b\}^* - \{ab, abb\}, \\ L_1 \backslash R^c &= \{a,b\}^* - \{b\}, \\ R' &= \{b\}. \end{aligned}$$

After checking the equality $\{a, ab\}\{b\} = \{ab, abb\}$ we can positively answer to the question "Does there exist a language L_2 such that $L_1L_2 = R$?". Such a language is $L_2 = R' = \{b\}$.

In this particular situation R' is the only solution to our equation. This is not always the case. For example, if $L_1 = R = \Sigma^*$ then $R^c = \emptyset$, $L_1 \backslash R^c = \emptyset$ and $R' = \Sigma^*$. However the language $L_2 = \{\lambda\}$ also satisfies the equation $\Sigma^* L_2 = \Sigma^*$.

Note that if we take $L_1 = \{a, ab\}$ and $R = \{ab, abb, ba\}$ we obtain the same R' as before, that is, $R' = \{b\}$. However, in this case the equality $\{a, ab\}\{b\} = \{ab, abb, ba\}$ does not hold. According to the preceding theorem this implies that the equation $\{a, ab\}L_2 = \{ab, abb, ba\}$ has no solutions. □

Theorem 5.10 *If $\diamond$ denotes the sequential insertion the problem Q_2 is decidable for regular languages L_1 and R.*

Proof. For given regular languages L_1, R over an alphabet Σ define:

$$R' = (R^c \rightleftharpoons L_1)^c,$$

where $\rightleftharpoons$ denotes the dipolar deletion, defined in Section 4.2. It has been proved in Theorem 4.11 that, if there exists $L_2 \subseteq \Sigma^*$ such that $L_1 \leftarrow L_2 = R$, then $L_1 \leftarrow R' = R$. Moreover, the regular language R' can be effectively constructed (see Corollary 4.7).

The algorithm which decides our problem will start with the construction of R'. Afterwards, the problem "Is $L_1 \leftarrow\!\!\!\!\!- R' = R$?" is decided, and the answer is also the answer to the original problem. Note that REG is closed under SIN, the language $L_1 \leftarrow\!\!\!\!\!- R'$ can be effectively constructed and the equality $L_1 \leftarrow\!\!\!\!\!- R' = R$ can be decided as the equivalence problem is decidable for REG. □

Theorem 5.11 *If $\diamond$ denotes the shuffle operation, the problem Q_2 is decidable for regular languages L_1 and R.*

Proof. Let L_1, R be given regular languages over an alphabet Σ and construct

$$R' = (R^c \dashrightarrow L_1)^c,$$

where $\dashrightarrow$ denotes the scattered sequential deletion, defined in Section 3.5.

(i) R' is a regular language and can be effectively constructed (see Theorem 3.25, Corollary 3.22).

(ii) $L_1 \amalg R' \subseteq R$. Assume the contrary and let x be a word in L_1, y be a word in R' such that $x = x_1 \dots x_n x_{n+1}$, $y = y_1 \dots y_n$, $x_i \in \Sigma^*$, $1 \leq i \leq n+1$, $y_i \in \Sigma^*$, $1 \leq i \leq n$, and $x_1 y_1 \dots x_n y_n x_{n+1} \in R^c$. According to the definition of the scattered deletion,

$$y \in (x_1 y_1 \dots x_n y_n x_{n+1} \dashrightarrow x) \subseteq R^c \dashrightarrow L_1.$$

This contradicts the fact that y was a word in R'. Our assumption was false, therefore $L_1 \amalg R' \subseteq R$.

(iii) Every language L_2 with the property $L_1 \amalg L_2 \subseteq R$ is included in R'. Assume the contrary and let $L_2 \subseteq \Sigma^*$ be a language such that $L_1 \amalg L_2 \subseteq R$ and $L_2 - R' \neq \emptyset$. Let y be a word in $L_2 - R'$. As y belongs to $(R')^c = R^c \dashrightarrow L_1$, there exist $z \in R^c$, $x \in L_1$, such that $x = x_1 \dots x_n x_{n+1}$, $y = y_1 \dots y_n$, $x_i \in \Sigma^*$, $1 \leq i \leq n+1$, $y_i \in \Sigma^*$, $1 \leq i \leq n$, and $z = x_1 y_1 \dots x_n y_n x_{n+1}$. According to the definition of the *shuffle* operation we have $z \in x \amalg y \subseteq L_1 \amalg L_2 \subseteq R$. We arrived at a contradiction as z was a word in R^c. Consequently, our assumption that such a language L_2 exists was false.

If there exists a language L_2 such that $L_1 \amalg L_2 = R$ then, according to (iii), $R = L_1 \amalg L_2 \subseteq L_1 \amalg R'$. As (ii) states that $L_1 \amalg R' \subseteq R$, we conclude that $L_1 \amalg R' = R$.

The algorithm for deciding our problem will start with the construction of R'. Then the problem "Is $L_1 \amalg R' = R$?" is decided, and the answer to it is the answer to our problem. □

If we consider the controlled SIN, for given L_1 and R over Σ, R regular, the equation $L_1 \diamond \Delta = R$ has card(Σ) variables: $\Delta(a), a \in \Sigma$. However, the problem $Q_{2,\Delta}$ is still decidable if all the parameters are regular languages. Moreover, it will follow from the proofs that a solution can be effectively constructed.

Theorem 5.12 *If $\diamond$ denotes the controlled SIN, the problem $Q_{2,\Delta}$ is decidable for regular languages L_1 and R.*

Proof. Let L_1, R be regular languages over an alphabet $\Sigma = \{a_1, \dots a_n\}$, $n \geq 1$. For every i, $1 \leq i \leq n$, construct the gsm:

$$\begin{aligned} g_i = \quad & (\Sigma, \Sigma \cup \{\#, \$\}, \{s_0, s, s'\}, s_0, \{s'\}, P_i), \\ P_i = \quad & \{s_0 a_j \longrightarrow a_j s_0 |\ 1 \leq j \leq n\} \cup \{s_0 a_i \longrightarrow a_i \# \$ s'\} \cup \\ & \{s_0 a_i \longrightarrow a_i \# s\} \cup \{s a_j \longrightarrow a_j s |\ 1 \leq j \leq n\} \cup \\ & \{s a_j \longrightarrow a_j \$ s' |\ 1 \leq j \leq n\} \cup \{s' a_j \longrightarrow a_j s' |\ 1 \leq j \leq n\}, \end{aligned}$$

where $\#, \$$ are new symbols which do not occur in Σ. It is easy to prove that for every language $L \subseteq \Sigma^*$ and every i, $1 \leq i \leq n$, we have:

$$g_i(L) = \{ua_i\#w\$v |\ u, v, w \in \Sigma^*, 1 \leq i \leq n, \text{ and } ua_iwv \in L\}. \qquad (*)$$

Define now the morphism $h : (\Sigma \cup \{\#, \$\})^* \longrightarrow \Sigma^*$ by:

$$h(\#) = h(\$) = \lambda,\ h(a) = a, \forall a \in \Sigma.$$

After these preliminary constructions, the proof will resemble that of Theorems 5.10, 5.11. Construct, for every i, $1 \leq i \leq n$, the language:

$$\Delta(a_i) = [h((g_i(R^c) \rightleftharpoons L_1) \cap \#\Sigma^*\$)]^c,$$

where $\rightleftharpoons$ denotes the dipolar deletion, defined in Section 4.2.

(i) The languages $\Delta(a_i)$, $1 \leq i \leq n$, are regular and can be effectively constructed (see Lemma 4.5 and Corollary 4.5).

(ii) $L_1 \leftarrow\!\!\dashv \Delta \subseteq R$. Assume the contrary: there exist i, $1 \leq i \leq n$, $x \in L_1$, $w \in \Delta(a_i)$ such that $x = ua_iv$ and $ua_iwv \in R^c$. According to $(*)$, the word $ua_i\#w\$v$ belongs to $g_i(R^c)$. Following the definition of the dipolar deletion, $\#w\$$ is a word in $(ua_i\#w\$v \rightleftharpoons ua_iv) \cap \#\Sigma^*\$ \subseteq (g_i(R^c) \rightleftharpoons L_1) \cap \#\Sigma^*\$$. Consequently, $w = h(\#w\$)$ belongs to $h((g_i(R^c) \rightleftharpoons L_1) \cap \#\Sigma^*\$)$ which contradicts the fact that $w \in \Delta(a_i)$. Our assumption was false, therefore $L_1 \leftarrow\!\!\dashv \Delta \subseteq R$.

(iii) Any control function Δ' such that $L_1 \leftarrow\!\!\dashv \Delta' \subseteq R$ has the property $\Delta'(a_i) \subseteq \Delta(a_i)$, $\forall i$, $1 \leq i \leq n$.

Assume the contrary and let Δ' be a control function as before such that there exists i, $1 \leq i \leq n$, with the property $\Delta'(a_i) - \Delta(a_i) \neq \emptyset$. Let w be a word in $\Delta'(a_i) - \Delta(a_i)$. As $w \in [\Delta(a_i)]^c$, the word $\#w\$$ is in $g_i(R^c) \rightleftharpoons L_1$. Therefore there exist $x \in g_i(R^c)$ and $z \in L_1$ such that $x = ua_i\#w\$v$, $z = ua_iv$, $u, w \in \Sigma^*$.

As $x = ua_i\#w\$v \in g_i(R^c)$, according to $(*)$, the word ua_iwv belongs to R^c. This contradicts the relation $L_1 \leftarrow\!\!\dashv \Delta' \subseteq R$ which implies $ua_iwv \in R$ for every $ua_iv \in L_1$, $1 \leq i \leq n$ and $w \in \Delta'(a_i)$. Our assumption that such a function Δ' exists was false.

Return to the proof of the theorem. If there exists a control function Δ' such that $L_1 \leftarrow\!\!\dashv \Delta' = R$, according to (iii) we have $\Delta'(a_i) \subseteq \Delta(a_i)$ $\forall i$, $1 \leq i \leq n$, which implies $R = L_1 \leftarrow\!\!\dashv \Delta' \subseteq L_1 \leftarrow\!\!\dashv \Delta$. As, according to (ii), $L_1 \leftarrow\!\!\dashv \Delta \subseteq R$, we deduce that $L_1 \leftarrow\!\!\dashv \Delta = R$.

The algorithm which decides our problem will start with the construction of the control function Δ. As REG is closed under controlled SIN, the

problem "Is $L_1 \leftarrow\!\!\dashv \Delta = R$?" is decidable, and its answer is the answer to our problem. □

Let $\diamond$ denote the SIN next to the letter a. The proof of the preceding theorem can be used to show that the problem Q_2 is decidable for regular languages L_1 and R. Indeed, the only modification is that we need to define the gsm g_i and the control function $\Delta(a_i)$ only for the control letter $a_i = a$.

Theorem 5.13 *If $\diamond$ denotes SIN, the problem Q_2^w is decidable for regular languages L_1 and R.*

Proof. Let L_1, R be regular languages over an alphabet Σ and let m be the length of the shortest word in R. If there exists a word w such that $L_1 \leftarrow w = R$, then it must satisfy the condition $\lg(w) \leq m$. As REG is closed under sequential insertion (see Theorem 2.3), the problem "Is $L_1 \leftarrow w = R$?" is decidable for words w and regular languages L_1 and R. The algorithm for deciding our problem will consist of checking whether or not $L_1 \leftarrow w = R$ for all words w with $\lg(w) \leq m$. The answer is YES if such a word w is found, and NO otherwise. □

Let $\diamond$ denote one of the operations: *catenation, PIN, shuffle, permuted SIN, permuted PIN, permuted scattered SIN, SIN next to a letter and PIN next to a letter*. The proof of the preceding theorem can be used to show that in all mentioned cases the problem Q_2^w is decidable for regular languages L_1 and R.

In the following, some undecidability results are presented. For a binary insertion operation $\diamond$, the existence of both a solution and a singleton solution L_2 to the equation $L_1 \diamond L_2 = R$ is proved to be undecidable for context-free languages L_1 and regular languages R. We start our investigation with the simplest case, where $\diamond$ denotes the catenation operation.

Theorem 5.14 *The problem "Does there exist a language L_2 such that $L_1 L_2 = R$?" is undecidable for context-free languages L_1 and regular languages R.*

Proof. Let Σ be an alphabet, $\text{card}(\Sigma) \geq 2$, and let $\#$ be a letter which does not occur in Σ. There exists a regular language $R = \Sigma^*\#$ such that the problem of the theorem is undecidable for context-free languages L_1.

Indeed, we notice that the equation $(L_1\#)L_2 = \Sigma^*\#$ holds for languages L_1, L_2 over Σ exactly in case $L_1 = \Sigma^*$ and $L_2 = \{\lambda\}$. Hence, if we could decide the problem of the theorem, we would be deciding the problem "Is $L_1 = \Sigma^*$?" for context-free languages L_1, which is impossible. □

We notice that in the above proof the language $L_2 = \{\lambda\}$ is a singleton. Therefore also the problem "Does there exist a word w such that $L_1 w = R$?" is undecidable for context-free languages L_1 and regular languages R.

Let $\diamond$ denote one of the operations: *SIN, PIN, iterated SIN and iterated PIN, shuffle, permuted scattered SIN, permuted SIN, permuted PIN, SIN next to a letter and PIN next to a letter.* The proof of the previous theorem and the above remark can be used to show that in all the cases, the problems Q_2 and Q_2^w are undecidable for context-free languages L_1 and regular languages R.

Note. If $\diamond$ stands for SIN next to a letter or PIN next to a letter, we choose the letter to be #.

Theorem 5.15 *Let $\diamond$ denote the controlled sequential insertion. The problems $Q_{2,\Delta}$, $Q_{2,\Delta}^w$ are undecidable for context-free languages L_1 and regular languages R.*

Proof. Let Σ be an alphabet, $\text{card}(\Sigma) \geq 2$, and let $\#, \$$ be symbols not belonging to Σ. There exists a regular language $R = \Sigma^*\#\$$ such that the problems $Q_{2,\Delta}$, $Q_{2,\Delta}^w$ are undecidable for context-free languages L_1. Indeed, we observe that the equation $L_1 \#\leftarrow\!\!\dashv \Delta = \Sigma^*\#\$$ holds for languages $L_1 \subseteq \Sigma^*$ iff $\Delta(\#) = \$$, $\Delta(a) = \emptyset$, $\forall a \in \Sigma$ and $L_1 = \Sigma^*$. The "if"-part is obvious. For the "only if"-part we notice that if the control function wouldn't be of the above form, illegal strings would occur in the result of the controlled SIN. On the other hand, the form of Δ forces L_1 to be Σ^*.

If we could decide either one of the problems of the theorem we would be deciding the problem "Is $L_1 = \Sigma^*$?" for context-free languages L_1, which is impossible. □

5.3 The left operand problem for insertion

This section will deal with the question whether or not the equation $L_1 \diamond L_2 = R$ has a solution L_1, where L_2, R are given languages, R regular, and $\diamond$ is an insertion operation. The existence of a singleton solution will also be investigated. More specifically, if $\diamond$ denotes a binary insertion operation, given languages L_2 and R, R regular, the following problems will be considered:

Q_1:"Does there exist a language L_1 such that $L_1 \diamond L_2 = R$?"

Q_1^w:" Does there exist a word w such that $w \diamond L_2 = R$?"

If $\diamond$ denotes a controlled insertion operation and R is a given language, Δ a given control function, the following problems will be also considered:
$Q_{1,\Delta}$:" Does there exist a language L_1 such that $L_1 \diamond \Delta = R$?"
$Q^w_{1,\Delta}$:"Does there exist a word w such that $w \diamond \Delta = R$?".
In the beginning, the simplest case, where $\diamond$ denotes the catenation and all the languages involved are regular, is investigated.

Theorem 5.16 *The problem "Does there exist a language L_1 such that $L_1L_2 = R$?" is decidable for regular languages L_2 and R.*

Proof. Similarly as Theorem 5.9. □

Theorem 5.17 *If $\diamond$ denotes the sequential insertion, the problem Q_1 is decidable for regular languages L_2 and R.*

Proof. Let L_2, R be regular languages over an alphabet Σ and define

$$R' = (R^c \longrightarrow L_2)^c.$$

It has been proved in Theorem 4.12 that, if there exists $L_1 \subseteq \Sigma^*$ with the property $L_1 \longleftarrow L_2 = R$ then also $R' \longleftarrow L_2 = R$. Moreover, the regular language R' can be effectively constructed (see Corollary 4.9).

The algorithm which decides our problem will start with the construction of R'. Then the problem "Is $R' \longleftarrow L_2 = R$?" is decided, and the answer is the answer to our problem. □

Theorem 5.18 *If $\diamond$ denotes the shuffle operation, the problem Q_1 is decidable for regular languages L_2 and R.*

Proof. It follows from Theorem 5.11 and from the fact that *shuffle* is a commutative operation. □

Theorem 5.19 *If $\diamond$ denotes the controlled sequential insertion, the problem $Q_{1,\Delta}$ is decidable for regular languages R and regular control functions Δ.*

Proof. Let R be a regular language over an alphabet Σ and Δ be a regular control function. Define the language

$$R' = (R^c \longmapsto \Delta\ )^c,$$

where $\longmapsto$ denotes the controlled sequential deletion, defined in Section 3.4.

(i) The language R' is regular and can be effectively constructed (see Theorem 3.20, Corollary 3.18).

(ii) $R' \leftarrow\!\!| \Delta \subseteq R$. Assume the contrary and let x be a word in R', $x = ua_iv$, $u, v \in \Sigma^*$, and w be a word in $\Delta(a_i)$ such that $ua_iwv \in R^c$. According to the definition of the controlled sequential deletion, the word x belongs to $(ua_iwv \mapsto \Delta) \subseteq R^c \mapsto \Delta$. We arrived at a contradiction as x was a word in R'. Our assumption was false, therefore $R' \leftarrow\!\!| \Delta \subseteq R$.

(iii) Any language L_1 with the property $L_1 \leftarrow\!\!| \Delta \subseteq R$ is included in R'. Assume the contrary and let L_1 be a language as before such that $L_1 - R' \neq \emptyset$. Let y be a word in $L_1 - R'$. As y is in $R^c \mapsto \Delta$, there exist words $x \in R^c$, $x = ua_iwv$, $u, v \in \Sigma^*$, $w \in \Delta(a_i)$ such that $y = ua_iv$. According to the definition of the controlled SIN, x belongs to $y \leftarrow\!\!| \Delta \subseteq L_1 \leftarrow\!\!| \Delta \subseteq R$. We arrived at a contradiction as x was a word in R^c. Consequently, our assumption that such a language L_1 exists was false.

If there exists a language L_1 such that $L_1 \leftarrow\!\!| \Delta = R$ then, using (ii) and (iii) we deduce that $R' \leftarrow\!\!| \Delta = R$.

The algorithm which decides our problem will begin with the construction of R'. The answer to our problem will be the answer to the question "Is $R' \leftarrow\!\!| \Delta = R$?", which is decidable. □

Let $\diamond$ denote the SIN next to the letter a. The proof of the preceding theorem can be used to show that the problem Q_1 is decidable for regular languages L_2 and R. For this purpose $\leftarrow\!\!|$ will be replaced with $\overset{a}{\leftarrow\!\!|}$ and $\mapsto$ with $\overset{a}{\mapsto}$.

Theorem 5.20 *If $\diamond$ denotes the sequential insertion, the problem Q_1^w is decidable for regular languages L_2 and R.*

Proof. The proof is similar to that of Theorem 5.13 and uses the fact that REG is closed under sequential insertion (see Theorem 2.3).

Let L_2 and R be regular languages over an alphabet Σ and let m be the length of the shortest word in R. Suppose $w \leftarrow L_2 = R$ holds for some w. Then $\lg(w) \leq m$ because, otherwise, the shortest word in R would not result from the insertion. Our algorithm will check, for all words w with $\lg(w) \leq m$, whether or not $w \leftarrow L_2 = R$. The answer is YES if such a w is found and NO otherwise. □

Let $\diamond$ be one of the operations: *catenation, PIN, shuffle, SIN next to a letter, PIN next to a letter, controlled SIN and controlled PIN.* The proof

of the preceding theorem and the closure properties of REG (Theorems 2.4, 2.18, 2.19) can be used to show that, in all cases, the problem Q_1^w (respectively $Q_{1,\Delta}^w$) is decidable for regular languages L_2 (regular control functions Δ) and regular languages R.

Let $\diamond$ be one of the operations: catenation, SIN, PIN, *shuffle*, SIN next to a letter and PIN next to a letter. The following theorems will show that the existence of both a solution and a singleton solution L_1 to the equation $L_1 \diamond L_2 = R$ is undecidable for context-free languages L_2 and regular languages R. The same result is obtained for the existence of a singleton solution in the case $\diamond \in \{$controlled SIN, controlled PIN$\}$.

Theorem 5.21 *The problems "Does there exist a language L_1 such that $L_1L_2 = R$?" and "Does there exist a word w such that $wL_2 = R$?" are undecidable for context-free languages L_2 and regular languages R.*

Proof. The claim follows immediately from Theorem 5.14 and the remark following it, by using the mirror image operator. □

Theorem 5.22 *If $\diamond$ denotes the sequential insertion, the problem Q_1 is undecidable for context-free languages L_2 and regular languages R.*

Proof. Let Σ be an alphabet with $\text{card}(\Sigma) \geq 2$ and let $\#$ be a letter which does not occur in Σ. We shall show that there exists a regular language $R = \Sigma^* \cup \{\#\}$, such that the problem Q_1 is undecidable for context-free languages L_2.

We assume the contrary and show how to solve the problem "Is $L = \Sigma^*$?" for context-free languages L. For a given context-free language $L \subseteq \Sigma^*$ construct $L_2 = L \cup \{\#\}$.

Claim. For all languages $L_1 \subseteq \Sigma^*$ we have:

$$L_1 \leftarrow L_2 = R \text{ iff } L_1 = \{\lambda\}, L = \Sigma^*,$$

where L_2, R are defined as above.

The implication " $\Longleftarrow$ " is obvious. For the reverse implication, let us assume that L_1 contains a nonempty word w. Then the string $w\#$ belongs to $L_1 \leftarrow L_2$ – a contradiction with the form of the words in R. Consequently, our assumption that L_1 contains a nonempty word was false. On the other hand, $L_1 = \{\lambda\}$ implies $L = \Sigma^*$, and the proof of the claim is complete.

The claim implies that the problem "Does there exist a language L_1 such that $L_1 \leftarrow (L \cup \{\#\}) = \Sigma^* \cup \{\#\}$?" amounts to the problem "Is $L = \Sigma^*$?". The theorem now follows as the latter problem is undecidable for context-free languages L. □

Note that in the proof of the preceding theorem the language $L_1 = \{\lambda\}$ is a singleton. Consequently, the proof can be used to show that if $\diamond$ denotes SIN, the problem Q_1^w is undecidable for context-free languages L_2 and regular languages R.

Let $\diamond$ denote one of the operations *PIN, shuffle*. The proof of the preceding theorem and the above remark can be used to show that, in both cases, the problems Q_1 and Q_1^w are undecidable for context-free languages L_2 and regular languages R.

Theorem 5.23 *If $\diamond$ denotes the SIN next to a letter, the problem Q_1 is undecidable for context-free languages L_2 and regular languages R.*

Proof. Let Σ be an alphabet such that $\text{card}(\Sigma) \geq 2$ and let b and $\#$ be letters not occurring in Σ. There exists a regular language $R = b\Sigma^* \cup \{b\#\}$ such that the problem Q_1 is undecidable for context-free languages L_2.

We assume the contrary and show how to solve the problem "Is $L = \Sigma^*$?" for context-free languages L.

For a given context-free language L, define $L_2 = L \cup \{\#\}$.

Claim. For all languages $L_1 \subseteq (\Sigma \cup b)^*$ we have:

$$L_1 \overset{b}{\leftarrow\!\shortmid} L_2 = R \text{ iff } L_1 = \{b\} \cup L_1', L = \Sigma^*,$$

where $L_1' \subseteq \Sigma^*$ and L_2, R are defined as above.

The implication " $\Longleftarrow$ " is obvious. For the reverse implication, let us assume that L_1 contains a word $ubv \in (\Sigma \cup b)^* b (\Sigma \cup b)^*$ different from b. Then the word $ub\#v$ belongs to $L_1 \overset{b}{\leftarrow\!\shortmid} L_2$ – a contradiction with the form of the words in R. Consequently, our assumption that such a word belongs to L_1 was false. As the words which do not contain b do not contribute to the result, the fact that L_1 is of the above form implies that $L = \Sigma^*$. The proof of the claim is thus complete.

From the claim we deduce that the problem "Does there exist a language L_1 such that $L_1 \overset{b}{\leftarrow\!\shortmid} (L \cup \{\#\}) = b\Sigma^* \cup \{b\#\}$?" amounts to the problem "Is $L = \Sigma^*$?". The theorem now follows as the latter problem is undecidable for context-free languages L. □

Theorem 5.24 *If $\diamond$ denotes the SIN next to a letter, the problem Q_1^w is undecidable for context-free languages L_2 and regular languages R.*

Proof. The proof is similar to the preceding. The only difference is that here we ask for a singleton solution and therefore:

$$L_1 \overset{b}{\longleftarrow} L_2 = b\Sigma^* \cup \{b\#\} \text{ iff } L_1 = \{b\}, L = \Sigma^*.$$

□

The proofs of Theorems 5.23 and 5.24 can be used to show that also for $\diamond$ denoting the *PIN next to a letter*, the problems Q_1 and Q_1^w are undecidable for context-free languages L_2 and regular languages R. Note that in both cases L_1 must equal $\{b\}$.

If $\diamond$ denotes the *controlled sequential insertion* or the *controlled parallel insertion*, the problems $Q_{1,\Delta}$ and $Q_{1,\Delta}^w$ are undecidable for context-free control functions Δ and regular languages R. This follows from Theorems 5.23, 5.24 and from the fact that SIN next to a letter and PIN next to a letter are special cases of controlled SIN, respectively controlled PIN.

5.4 The right operand problem for deletion

In Section 5.2 the existence of a solution to the equation $L_1 \diamond L_2 = R$ was investigated, where L_1, R were given languages, R a regular one, and $\diamond$ was an insertion operation. A similar problem for $\diamond$ denoting a deletion operation will be studied in the sequel.

More specifically, if $\diamond$ denotes a binary deletion operation, for given languages L_1 and R, R regular, consider the problems:
Q_2:"Does there exist a language L_2 such that $L_1 \diamond L_2 = R$?"
Q_2^w: "Does there exist a word w such that $L_1 \diamond w = R$?"

If $\diamond$ stands for a Δ-controlled deletion operation, for given languages L_1 and R, R regular, the following problems will also be investigated:
$Q_{2,\Delta}$:"Does there exist a control function Δ such that $L_1 \diamond \Delta = R$?"
$Q_{2,\Delta}^w$:"Does there exist a singleton control function Δ such that $L_1 \diamond \Delta = R$?".

If the considered problem is decidable, from the proofs will follow that one can also construct a language, respectively a control function satisfying the equation.

If $\diamond$ denotes a binary deletion operation and L_1 is a given language, a word y is called *right-useful with respect to* L_1 *and* $\diamond$ if there exists $x \in L_1$ such that $x \diamond y \neq \emptyset$. A language L_2 is called right-useful with respect to L_1 and $\diamond$ if it consists only of right-useful words with respect to L_1 and $\diamond$.

Consider now the case of the controlled sequential deletion. Let Δ be a given control function and L_1 be a given language over Σ^*. A word $y \in \Delta(a)$ is called *right-useful with respect to* L_1 *and* a, if there exists an $x \in L_1$ such that $x \overset{a}{\longmapsto} y \neq \emptyset$. The function Δ is called right-useful with respect to L_1, if for every $a \in \Sigma$, $\Delta(a)$ consists only of right-useful words with respect to L_1 and a.

As we refer in this section only to the right operand problem, if L_1 and $\diamond$ are clear from the context, the word y, the language L_2 respectively the function Δ will be termed simply *useful*.

From the above definitions it follows that the problems Q_2, Q_2^w, $Q_{2,\Delta}$, $Q_{2,\Delta}^w$ are equivalent with the corresponding problems where the existence of a useful language, word, control function, singleton control function is investigated. Therefore in the sequel, when we want to prove an undecidability result, we will mean a useful language, word, control function, singleton control function when referring to the corresponding items whose existence is studied.

We will begin our investigation with the simplest case, where the operation involved is the left (right) quotient, and all the languages considered are regular.

Theorem 5.25 *The problem "Does there exist a language* L_2 *such that* $L_2 \backslash L_1 = R$*?" is decidable for regular languages* L_1 *and* R*.*

Proof. Let L_1 and R be regular languages over an alphabet Σ and consider:

$$R' = (L_1 / R^c)^c.$$

(i) R' is a regular language and can be effectively constructed (see Theorem 3.1, Corollary 3.1 and the remarks following them).

(ii) $R' \backslash L_1 \subseteq R$. Assume the contrary and let $x \in L_1$, $y \in R'$ such that $x = yz$, $z \in R^c$. This implies $y = x/z \subseteq L_1/R^c$, which contradicts the fact that y was a word in R'.

(iii) Any language L_2 with the property $L_2 \backslash L_1 \subseteq R$ is included in R'. Assume the contrary and let y be a word in such an L_2, satisfying $y \in R'^c$. Consequently there exist $x \in L_1$, $z \in R^c$ such that $x = yz$. This further

implies $z = y\backslash x \subseteq L_2\backslash L_1 \subseteq R$. We arrived at a contradiction as z was a word in R^c.

If there exists a language L_2 such that $L_2\backslash L_1 = R$, from (ii) and (iii) we deduce that also $R'\backslash L_1 = R$. The algorithm for deciding our problem will consist of constructing R' and deciding whether or not $R'\backslash L_1$ equals R. □

An analogous proof can be used to show that the problem "Does there exist a language L_2 such that $L_1/L_2 = R$?" is decidable for regular languages L_1 and R. The language R' will be in this case:

$$R' = (R^c\backslash L_1)^c.$$

Example 5.2 Let $L_1 = \{ab, a^2b^2\}$ and $R = \{b, ab^2\}$. We are investigating the existence of a solution L_2 of the equation $L_2\backslash L_1 = R$. Using the method of the preceding theorem we construct the languages:

$$\begin{aligned} R^c = &\quad \{a,b\}^* - \{b, ab^2\}, \\ L_1/R^c = &\quad \{\lambda, ab, a^2b^2, a^2\}, \\ R' = &\quad \{a,b\}^* - \{\lambda, ab, a^2b^2, a^2\}. \end{aligned}$$

In order to check whether or not $R'\backslash L_1 = R$ we notice that the set of useful words of R' is $R'_u = \{a, aab\}$. The equality

$$\{a, aab\}\backslash\{ab, a^2b^2\} = \{b, ab^2\}$$

holds, therefore there exists a solution to the equation $L_1L_2 = R$, namely the language $L_2 = R' = \{a,b\}^* - \{\lambda, ab, a^2b^2, a^2\}$. Note that R' is the largest language satisfying our equation. It includes, for example, the languages R'_u and $\{a\}$ which are also solutions.

Observe that the same R' is obtained if we take L_1 as before and $R = \{b, ab^2, ba\}$. However, in this case the equality $R'\backslash L_1 = R$ does not hold. According to the preceding theorem, this implies that the equation $L_2\backslash\{ab, a^2b^2\} = \{b, ab^2, ba\}$ has no solution. □

Corollary 5.1 *All languages that can be obtained from a regular language L_1 by left quotient can be effectively constructed. For each such language R, a regular R' such that $R'\backslash L_1 = R$ can be effectively constructed. Moreover, R' is the largest language with the property $R'\backslash L_1 = R$.*

Proof. From Corollary 3.2 follows that there are finitely many languages that can be obtained from L_1 by left quotient. Moreover, a (finite) class of languages that includes them can be effectively constructed. All languages in the class are regular.

According to the preceding theorem, for each language in the class we can check whether or not the language is actually obtained from L_1 by left quotient. Let R be such a language, that is, a language for which there exists an L_2 such that $L_2 \backslash L_1 = R$. Then the language R' from the preceding theorem satisfies the requested conditions. □

A similar corollary can be proved for the right quotient of languages. The proof uses the remarks following Corollary 3.2 and the one following the preceding theorem.

Theorem 5.26 *If $\diamond$ denotes the sequential deletion, the problem Q_2 is decidable for regular languages L_1 and R.*

Proof. Let L_1, R be regular languages over an alphabet Σ and construct:

$$R' = (L_1 \rightleftharpoons R^c)^c,$$

where $\rightleftharpoons$ is the dipolar deletion defined in Section 4.2.

(i) The language R' is regular and can be effectively constructed (see Lemma 4.5 and Corollary 4.5).

(ii) $L_1 \longrightarrow R' \subseteq R$. Assume the contrary and let $x \in L_1$, $y \in R'$ such that $x = z_1 y z_2$, $z = z_1 z_2 \in R^c$. According to the definition of the dipolar deletion, y belongs to $x \rightleftharpoons z \subseteq L_1 \rightleftharpoons R^c$. We arrived at a contradiction as y was a word in R'.

(iii) Any language L_2 satisfying $L_1 \longrightarrow L_2 \subseteq R$ is included in R'. Assume the contrary and let $y \notin R'$ be a word belonging to such an L_2. There exist $x \in L_1$, $z \in R^c$ such that $x = z_1 y z_2$, $z = z_1 z_2$. This implies that z belongs to $x \longrightarrow y \subseteq L_1 \longrightarrow L_2 \subseteq R$. We arrived at a contradiction as z was a word in R^c.

If there exists a language L_2 such that $L_1 \longrightarrow L_2 = R$ then, using (ii) and (iii), we deduce that also $L_1 \longrightarrow R' = R$. The algorithm for deciding our problem starts with the construction of R'. Then the equality $L_1 \longrightarrow R' = R$ is decided and the answer is the answer to our problem. □

Corollary 5.2 *The languages that can be obtained from a regular L_1 by sequential deletion can be effectively constructed. For each such language R, a regular R' such that $L_1 \longrightarrow R' = R$ can be effectively constructed. Moreover, the language R' is the largest one with this property.*

Proof. We use Corollary 3.5. There are finitely many languages that can be obtained from the regular L_1 by sequential deletion. Moreover, we are able to construct a (finite) class of languages that includes them. All languages in this class are regular.

According to the preceding theorem, for each language from the class, one can decide whether or not the language is actually obtained from L_1 by sequential deletion. Let R be such a language, that is, a language for which there exists an L_2 such that $L_1 \longrightarrow L_2 = R$. Then the language R' constructed in the preceding theorem satisfies the requested conditions. □

Theorem 5.27 *If $\diamond$ denotes the scattered sequential deletion, the problem Q_2 is decidable for regular languages L_1 and R.*

Proof. Let L_1 and R be regular languages over an alphabet Σ and construct:

$$R' = (L_1 \dashrightarrow R^c)^c,$$

where $\dashrightarrow$ denotes the scattered sequential deletion defined in Section 3.5.

(i) R' is a regular language and can be effectively constructed (see Theorem 3.25, Corollary 3.22).

(ii) $L_1 \dashrightarrow R' \subseteq R$. Assume the contrary and let $x \in L_1$, $y \in R'$ be words such that $x = x_1 y_1 \dots x_n y_n x_{n+1}$, $x_i \in \Sigma^*$, $1 \leq i \leq n+1$, $y_i \in \Sigma^*$, $1 \leq i \leq n$, $y = y_1 \dots y_n$ and $z = x_1 \dots x_n x_{n+1} \in R^c$. According to the definition of the scattered SD, the word y belongs to $(x \dashrightarrow z) \subseteq L_1 \dashrightarrow R^c$. We arrived at a contradiction as y was a word in R'.

(iii) Any language L_2 with the property $L_1 \dashrightarrow L_2 \subseteq R$ is included in R'. Assume the contrary and let L_2 be a language as before, such that $L_2 - R' \neq \emptyset$. Let y be a word in $L_2 - R'$. As y belongs to $L_1 \dashrightarrow R^c$ there exist words $x \in L_1$, $z \in R^c$ such that $x = x_1 y_1 \dots x_n y_n x_{n+1}$, $x_i \in \Sigma^*$, $1 \leq i \leq n+1$, $y_i \in \Sigma^*$, $1 \leq i \leq n$, and $z = x_1 \dots x_n x_{n+1}$ and $y = y_1 \dots y_n$. Consequently we have $z \in (x \dashrightarrow y) \subseteq L_1 \dashrightarrow L_2 \subseteq R$. This contradicts the fact that z was a word in R^c.

If there exists a language L_2 such that $L_1 \dashrightarrow L_2 = R$ then, according to (ii) and (iii), $L_1 \dashrightarrow R' = R$. The algorithm for deciding Q_2 will consist of constructing R' and deciding whether or not $L_1 \dashrightarrow R' = R$. □

Theorem 5.28 *If $\diamond$ denotes the controlled sequential deletion, the problem $Q_{2,\Delta}$ is decidable for regular languages L_1 and R.*

Proof. Let L_1, R be regular languages over an alphabet $\Sigma = \{a_1, \dots, a_n\}$, $n \geq 1$ and let $\#, \$$ be letters which do not occur in Σ. We use the morphism

h and the gsm's g_i, $1 \leq i \leq n$, defined in Theorem 5.12 to construct for every i, $1 \leq i \leq n$, the value of the function Δ:

$$\Delta(a_i) = [h((g_i(L_1) \rightleftharpoons R^c) \cap \#\Sigma^*\$)]^c,$$

where $\rightleftharpoons$ denotes the dipolar deletion defined in Section 4.2.

(i) $\Delta(a_i)$, $1 \leq i \leq n$, are regular languages and can be effectively constructed (see Lemma 4.5, Corollary 4.5).

(ii) $L_1 \longmapsto \Delta \subseteq R$. Assume the contrary and let $x = ua_iwv \in L_1$, $w \in \Delta(a_i)$ be words such that $ua_iv \in R^c$. According to the relation (*) of Theorem 5.12 $ua_i\#w\$v \in g_i(L_1)$ and therefore $\#w\$ \in (g_i(L_1) \rightleftharpoons R^c) \cap \#\Sigma^*\$$. This implies that $w \in h((g_i(L_1) \rightleftharpoons R^c) \cap \#\Sigma^*\$)$, which contradicts the fact that $w \in \Delta(a_i)$.

(iii) Any control function Δ' which satisfies the relation $L_1 \longmapsto \Delta' \subseteq R$ has the property $\Delta'(a_i) \subseteq \Delta(a_i)$ for all i, $1 \leq i \leq n$. Assume the contrary and let Δ' be a function as before such that $\Delta'(a_i) - \Delta(a_i) \neq \emptyset$ for some i, $1 \leq i \leq n$. Let w be a word in $\Delta'(a_i) - \Delta(a_i)$. As $w \in (\Delta(a_i))^c$ we deduce that $\#w\$ \in g_i(L_1) \rightleftharpoons R^c$. This implies the existence of the words $ua_i\#w\$v \in g_i(L_1)$, $ua_iv \in R^c$. According to the definition of the gsm g_i, the word ua_iwv belongs to L_1, that implies in turn $ua_iv \in L_1 \longmapsto \Delta' \subseteq R$. This contradicts the fact that $ua_iv \in R^c$.

If there exists a function Δ' such that $L \longmapsto \Delta' = R$ then, using (ii) and (iii), we deduce that $L \longmapsto \Delta = R$. The algorithm for deciding $Q_{2,\Delta}$ will consist of constructing Δ and deciding whether or not $L \longmapsto \Delta = R$. □

If $\diamond$ denotes the *SD next to one letter* the above proof can be used to show that Q_2 is decidable for regular languages L_1 and R. Indeed, for given regular languages $L_1, R \subseteq \Sigma^*$ and control letter a_i, one can construct

$$R' = [h(g_i(L_1) \rightleftharpoons R^c) \cap \#\Sigma^*\$]^c.$$

Then the problem "Is $L_1 \overset{a_i}{\longmapsto} R' = R$?" is decided and it can be proved as in Theorem 5.28 that this problem is equivalent with Q_2.

In the following, some undecidability results are proved. We begin with the simplest case, where the operation considered is the left (right) quotient.

Theorem 5.29 *The problem "Does there exist a language L_2 such that $L_2 \backslash L_1 = R$?" is undecidable for context-free languages L_1 and regular languages R.*

Proof. Let Σ be an alphabet, $\mathrm{card}(\Sigma) \geq 2$, and let $\#$ be a letter which does not occur in Σ. There exists a regular language $R = \#\Sigma^*$ such that the problem of the theorem is undecidable for context-free languages L_1.

Indeed, the equation $L_2 \backslash (\#L) = \#\Sigma^*$ holds for languages L and L_2 over Σ exactly in the case $L_2 = \{\lambda\}$ and $L = \Sigma^*$. (Recall our convention concerning usefulness.) Hence, if we could decide the problem of the theorem, we would be deciding the problem "Is $L = \Sigma^*$?" for context-free languages L, which is impossible. □

Notice that in the above proof the language $L_2 = \{\lambda\}$ is a singleton. Therefore also the problem "Does there exist a word w such that $w \backslash L_1 = R$?" is undecidable for context-free languages L_1 and regular languages R.

The problems "Does there exist a language L_2 such that $L_1 / L_2 = R$?" and "Does there exist a word w such that $L_1 / w = R$?" are undecidable for context-free languages L_1 and regular languages R. Indeed, if we take $R = \Sigma^*\#$ and for a context-free $L \subseteq \Sigma^*$, $L_1 = L\#$, the proof is analogous to that of the preceding theorem.

Theorem 5.30 *If $\diamond$ denotes the sequential deletion, the problem Q_2 is undecidable for context-free languages L_1 and regular languages R.*

Proof. Let Σ be an alphabet, $\mathrm{card}(\Sigma) \geq 2$, and let $\#, \$$ be letters which do not occur in Σ. There exists a regular language $R = \#\Sigma^+\# \cup \ \$\Sigma^*\$$ such that Q_2 is undecidable for context-free languages L_1. We assume the contrary and show how to solve the problem "Is $L = \Sigma^*$?" for context-free languages L.

Let $L \subseteq \Sigma^*$ be a context-free language and consider the language $L_1 = \#\Sigma^+\# \cup \$L\$$. For all languages $L_2 \subseteq \Sigma^*$, the equation:

$$\#\Sigma^+\# \cup \$L\$ \longrightarrow L_2 = \#\Sigma^+\# \cup \$\Sigma^*\$ \qquad (*)$$

holds if and only if $L_2 = \{\lambda\}$ and $L = \Sigma^*$.

The implication " $\Longleftarrow$ " is obvious. For the reverse implication assume that $(*)$ holds and that L_2 contains a nonempty useful word w. One of the next situations must hold:

– $w = \#$, which implies $\#v\# \in L_1$, for some $v \in \Sigma^+$, and therefore,

$$v\# \in (\#v\# \longrightarrow \#) \subseteq R.$$

– $w \in \Sigma^+$, which implies $\#w\# \in L_1$ and therefore,

$$\#\# \in (\#w\# \rightarrow w) \subseteq L_1 \longrightarrow L_2 = R.$$

– $w = \#v$, $v \in \Sigma^+$, which implies $\#v\# \in L_1$ and therefore,

$$\# \in (\#v\# \longrightarrow \#v) \subseteq R.$$

– $w = v\#$, $v \in \Sigma^+$, which implies $\#v\# \in L_1$ and therefore,

$$\# \in (\#v\# \longrightarrow v\#) \subseteq R.$$

– $w = \#v\#$, $v \in \Sigma^+$, which implies $\#v\# \in L_1$ and therefore,

$$\lambda \in (\#v\# \longrightarrow \#v\#) \subseteq R.$$

– $w = v\$$, $v \in \Sigma^*$, which implies $\$v'v\$ \in L_1$, for some $v' \in \Sigma^*$, and therefore,

$$\$v' \in (\$v'v\$ \longrightarrow v\$) \subseteq R.$$

– $w = \$v$, $v \in \Sigma^*$, which implies $\$vv'\$ \in L_1$, for some $v' \in \Sigma^*$, and therefore,

$$v'\$ \in (\$vv'\$ \longrightarrow \$v) \subseteq R.$$

– $w = \$v\$$, $v \in \Sigma^*$, which implies

$$\lambda \in (\$v\$ \longrightarrow \$v\$) \subseteq R.$$

As we are considering here only useful words, that is, only words w which can actually be deleted, the above list is an exhaustive one. In all the considered cases we arrived at contradictions with the form of the words in R. Consequently, our assumption that L_2 contains nonempty words was false.

The fact that $L_2 = \{\lambda\}$ implies that $L = \Sigma^*$, and the proof of the reverse implication is complete.

If we could decide the problem of the theorem, we could decide whether for given context-free languages L, there exists a solution L_2 to the equation $(*)$. According to the facts proved above, this would in turn imply that we could decide the problem "Is $L = \Sigma^*$?" for context-free languages L, which is impossible. □

Noticing that in the above proof $L_2 = \{\lambda\}$ is a singleton language, the proof can be used to show that for $\diamond$ denoting the *sequential deletion*, the problem Q_2^w is undecidable for context-free languages L_1 and regular languages R.

A similar proof using the same construction can be used to show that for $\diamond$ denoting *the permuted SD, the scattered SD, the iterated SD, the permuted scattered SD, the parallel deletion, the permuted PD, the iterated PD,* the problems Q_2 and Q_2^w are undecidable for context-free languages L_1 and regular languages R.

Theorem 5.31 *If $\diamond$ denotes the controlled sequential deletion, the problem $Q_{2,\Delta}$ is undecidable for context-free languages L_1 and regular languages R.*

Proof. Let Σ be an alphabet, $\text{card}(\Sigma) \geq 2$, and let $\#, \$$ be letters which do not occur in Σ. There exists a regular language $R = \Sigma^*\#$ such that $Q_{2,\Delta}$ is undecidable for context-free languages L_1. We assume the contrary and show how to solve the problem"Is $L = \Sigma^*$?" for context-free languages L.

For a given context-free language $L \subseteq \Sigma^*$, construct $L_1 = L\#\$$.

For all control functions Δ we have:

$$L\#\$ \longmapsto \Delta = \Sigma^*\# \quad \text{iff} \quad \begin{array}{l} \Delta(\#) = \{\$\}, \Delta(a) = \emptyset, \forall a, a \neq \# \\ \text{and} \ \ L = \Sigma^*. \end{array} \qquad (*)$$

The implication " $\Longleftarrow$ " is obvious. For the reverse implication assume that $\Delta(\#)$ contains at least a word $w \neq \$$. The only possibility is $w = \lambda$ which implies that words of the form $u\#\$ \in R$ – a contradiction. (Recall that we are looking for useful words.)

The function Δ has the value $\emptyset$ for any other letter than $\#$. Indeed, the only possible value for $\Delta(\$)$ would be λ, which would imply that words containing $\$$ would occur in R – a contradiction.

Assume that, for some $a \in \Sigma$, $\Delta(a)$ contains a word w. One of the following possibilities must occur:

– w is of the form $w = v\#\$$, $v \in \Sigma^*$, which implies $v'av \in L$ for some $v' \in \Sigma^*$ (w is useful) and therefore:

$$v' \in (v'av\#\$ \longmapsto \Delta) \subseteq R.$$

– w is of the form $v\#$, $v \in \Sigma^*$, which implies $v'av \in L$ for some $v' \in \Sigma^*$ and therefore,

$$v'a\$ \in (v'av\#\$ \longmapsto \Delta) \subseteq R.$$

– $w \in \Sigma^*$ which implies $vawv' \in L$ for some $v, v' \in \Sigma^*$ and therefore:

$$vav'\#\$ \in (vawv'\#\$ \longmapsto \Delta) \subseteq R.$$

In all cases we arrived at contradictions, therefore our assumption that Δ is nonempty for a letter $a \in \Sigma$ was false.

On the other hand, the fact that the control function is of the form mentioned in (*) implies that $L = \Sigma^*$. The second implication is proved.

From the above claim it follows that the problem "Does there exist a control function Δ such that $L\#\$ \longmapsto \Delta = \Sigma^*\#$?" amounts to the problem "Is $L = \Sigma^*$?". Consequently, if the former would be decidable then the latter would be also decidable for context-free languages L. □

Notice that the control function Δ in the proof of the preceding theorem is nonempty only for one letter and has a singleton as its value. Consequently, we can use the same proof to show that for $\diamond$ denoting *the controlled SD*, the problem $Q^w_{2,\Delta}$ is undecidable for context-free languages L_1 and regular languages R. The same remark implies that if $\diamond$ denotes the *SD next to a letter* or *PD next to a letter*, namely $\#$, the problems Q_2 and Q^w_2 are undecidable for context-free languages L_1 and regular languages R. The only difference is that, in the case of PD next to a letter, the control function Δ has the value λ for all $a \in \Sigma \cup \{\$\}$.

Theorem 5.32 *Let $\diamond$ denote the controlled parallel deletion. The problem $Q^w_{2,\Delta}$ is undecidable for context-free languages L_1 and regular languages R.*

Proof. Let Σ be an alphabet, $\mathrm{card}(\Sigma) \geq 2$, and $\#, \$$ be letters not belonging to Σ. There exists a regular language $R = \Sigma^+ \cup \{\#a |\ a \in \Sigma\}$ such that the problem $Q^w_{2,\Delta}$ is undecidable for context-free languages L_1.

We notice that the equation

$$L_1\$ \cup \{\#a\$ |\ a \in \Sigma\} \models \Delta = \Sigma^+ \cup \{\#a |\ a \in \Sigma\}$$

holds for $L_1 \subseteq \Sigma^*$ and singleton control functions Δ iff

$$\Delta(\#) = \Delta(\$) = \lambda, \Delta(a) = \$, a \in \Sigma, \text{ and } L_1 = \Sigma^+.$$

Consequently, if we could decide $Q^w_{2,\Delta}$, we would be deciding the problem "Is $L_1 = \Sigma^+$?" for context-free languages L_1, which is impossible. □

5.5 The left operand problem for deletion

This section deals with problems similar to the ones studied in Section 5.3 for insertion operations. Let $\diamond$ denote a binary deletion operation. For given

languages L_2 and regular languages R, the following decision problems are investigated:
Q_1: "Does there exist a language L_1 such that $L_1 \diamond L_2 = R$?"
Q_1^w: "Does there exist a word w such that $w \diamond L_2 = R$?".

If $\diamond$ stands for a Δ-controlled deletion operation, for given control functions Δ and regular languages R, the following problems will be also considered:
$Q_{1,\Delta}$: "Does there exist a language L_1 such that $L_1 \diamond \Delta = R$?"
$Q_{1,\Delta}^w$:" Does there exist a word w such that $w \diamond \Delta = R$?".

Let $\diamond$ be a binary deletion operation. If L_2 is a language over an alphabet Σ, the word x is called *left useful with respect to* $\diamond$ *and* L_2 (shortly, *useful*) if there exists a $y \in L_2$ such that $x \diamond y \neq \emptyset$. A language L_1 is called left useful with respect to $\diamond$ and L_2 (shortly, useful), if it consists only of useful words.

The notion is extended to the controlled sequential deletion in the obvious fashion. Let Δ be a control function defined on Σ. A word $x \in \Sigma^+$ is called *left-useful with respect to* Δ (shortly, *useful*) if $x \longmapsto \Delta \neq \emptyset$. A language L_1 is called left-useful with respect to Δ (shortly, useful) if it consists only of useful words.

From the above definitions it follows that the problems Q_1, Q_1^w, $Q_{1,\Delta}$, $Q_{1,\Delta}^w$ are equivalent with the corresponding problems where the existence of a useful language or word is investigated. Therefore in the sequel, when we want to prove an undecidability result, we will mean a useful language or word when referring to a language or word whose existence is investigated.

We start with the simplest case, where the operation is the left (right) quotient and all the languages involved are regular.

Theorem 5.33 *The problem "Does there exist a language L_1 such that $L_2 \backslash L_1 = R$?" is decidable for regular languages L_2 and R.*

Proof. Let L_2 and R be regular languages over an alphabet Σ and consider:

$$R' = (L_2 R^c)^c.$$

(i) R' is a regular language and can be effectively constructed.

(ii) $L_2 \backslash R' \subseteq R$. Assume the contrary and let $x \in R'$, $y \in L_2$ be words such that $x = yz$, $z \in R^c$. This implies that $x = yz \in L_2 R^c$ – a contradiction to the fact that x was a word in R'.

(iii) Any language L_1 with the property $L_2\backslash L_1 \subseteq R$ is included in R'. Assume the contrary and let L_1 be a language as before such that $L_1 - R' \neq \emptyset$. If x is a word in $L_1 - R'$ then $x = yz$ for some $y \in L_2$ and $z \in R^c$. This implies that $z = (y\backslash x) \in L_2\backslash L_1 \subseteq R$. We arrived at a contradiction as z was a word in R^c.

If there exists a language L_1 such that $L_2\backslash L_1 = R$ then, using (ii) and (iii), we deduce that $L_2\backslash R' = R$. The algorithm for deciding our problem will consist of constructing R' and deciding whether or not $L_2\backslash R' = R$. □

Theorem 5.34 *The problem "Does there exist a word w such that $L_2\backslash w = R$?" is decidable for regular languages L_2 and R.*

Proof. Let L_2, R be regular languages over an alphabet Σ. Notice that, if R is an infinite language, the answer to our problem is NO. If R is finite, we can effectively construct the regular set:

$$P = (L_2R^c)^c - \bigcup_{S \subset R} (L_2S^c)^c,$$

where by $\subset$ we denote strict inclusion.

Claim. For all $w \in \Sigma^*$ we have: $w \in P$ iff $L_2\backslash w = R$.

Indeed, from (ii), (iii) of the preceding theorem it follows that for given regular languages L_2 and R we have:

$$(L_2R^c)^c = \{v |\ L_2\backslash v \subseteq R\}.$$

Therefore, if $L_2\backslash w = R$ then:

$$\begin{aligned} w \in &\quad \{v |\ L_2\backslash v \subseteq R\}, \\ w \notin &\quad \{v |\ L_2\backslash v \subseteq S \subset R\}, \end{aligned}$$

and consequently $w \in P$.

For the reverse implication, let w be a word in P. As $L_2\backslash w \subseteq R$ but $L_2\backslash w$ is not included in any proper subset of R we have $L_2\backslash w = R$. The proof of the claim is thus complete.

The algorithm for deciding our problem will check first the finiteness of R. If R is infinite, the answer is NO. Else, the set P is constructed and its emptiness is decided. If $P = \emptyset$, the answer is NO. Else the answer is YES and any word w in P satisfies the equation $L_2\backslash w = R$. □

The proofs of Theorem 5.33 and Theorem 5.34 can be modified in order to show that the problems:
"Does there exist a language L_1 such that $L_1/L_2 = R$?" ,
"Does there exist a word w such that $w/L_2 = R$?"
are decidable for regular languages L_2 and R. The languages constructed in the proofs will be respectively:

$$\begin{aligned} R' = &\quad (R^c L_2)^c, \\ P = &\quad (R^c L_2)^c - \bigcup_{S \subset R} (S^c L_2)^c. \end{aligned}$$

Theorem 5.35 *If $\diamond$ denotes the sequential deletion then the problem Q_1 is decidable for regular languages L_2 and R.*

Proof. Let L_2 and R be regular languages over an alphabet Σ and consider

$$R' = (R^c \leftarrow L_2)^c,$$

where $\leftarrow$ denotes the sequential insertion.

(i) R' is a regular language and can be effectively constructed (see Theorem 2.3).

(ii) $R' \rightarrow L_2 \subseteq R$. Assume the contrary and let $x \in R'$, $y \in L_2$ be words such that $x = x_1 y x_2$, $x_1, x_2 \in \Sigma^*$ and $z = x_1 x_2 \in R^c$. According to the definition of the sequential insertion we have $x \in (z \leftarrow y) \subseteq R^c \leftarrow L_2$. We arrived at a contradiction as x was a word in R'.

(iii) Any language L_1 with the property $L_1 \rightarrow L_2 \subseteq R$ is included in R'. Assume the contrary and let x be a word in $L_1 - R'$. As x belongs to $(R')^c$, there exist words $z \in R^c$, $y \in L_2$ such that $x = x_1 y x_2$, $x_1, x_2 \in \Sigma^*$, $z = x_1 x_2$. According to the definition of the sequential deletion, $z \in (x \rightarrow y) \subseteq L_1 \rightarrow L_2 \subseteq R$. This contradicts the fact that z was a word in R^c.

If there exists a language L_1 such that $L_1 \rightarrow L_2 = R$ then, using (ii) and (iii), we deduce that $R' \rightarrow L_2 = R$. The algorithm for deciding Q_1 will consist of constructing R' and deciding whether or not $R' \rightarrow L_2 = R$. □

Theorem 5.36 *If $\diamond$ denotes the sequential deletion, the problem Q_1^w is decidable for regular languages L_2 and R.*

Proof. Let L_2 and R be regular languages over an alphabet Σ.

We notice first that, if R is an infinite language, the answer to our problem is NO.

If R is finite, we can effectively construct the regular set:

$$P = (R^c \leftarrow L_2)^c \; - \; \bigcup_{S \subset R} (S^c \leftarrow L_2)^c.$$

Claim. For all $w \in \Sigma^*$ we have: $w \in P$ if and only if $w \longrightarrow L_2 = R$.

Indeed, using the proof of (ii) and (iii) from Theorem 5.35 we can deduce that for given regular languages L_2 and R:

$$(R^c \longleftarrow L_2)^c = \{v |\ v \longrightarrow L_2 \subseteq R\}.$$

Consequently, if $w \longrightarrow L_2 = R$ then:

$$\begin{array}{ll} w \in & \{v |\ v \longrightarrow L_2 \subseteq R\}, \\ w \notin & \{v |\ v \longrightarrow L_2 \subseteq S \subset R\}, \end{array}$$

and therefore $w \in P$.

For the reverse implication, let w be a word in P. We have that $w \longrightarrow L_2$ is included in R, but it is not included in any proper subset S of R. This implies that $w \longrightarrow L_2 = R$ and the proof of the claim is complete.

The algorithm for deciding Q_1^w will check first the finiteness of R. If R is infinite, the answer to Q_1^w is NO. Else, the set P is constructed and the emptiness of P is decided. If $P = \emptyset$, the answer is NO. Else, the answer is YES and any word w in P satisfies the equation $w \longrightarrow L_2 = R$. □

If $\diamond$ denotes *the scattered sequential deletion*, the problems Q_1 and Q_1^w are decidable for regular languages L_2 and R. The proofs can be obtained from those of Theorem 5.35 and Theorem 5.36 by replacing the sequential deletion by scattered sequential deletion and the sequential insertion by *shuffle*.

Theorem 5.37 *If $\diamond$ denotes the iterated sequential deletion, the problem Q_1^w is decidable for regular languages L_2 and R.*

Proof. Let L_2 and R be regular languages over an alphabet Σ. If there exists a word w such that $w \longrightarrow^* L_2 = R$ then R is a finite language and $w \in R$. Consequently, the algorithm for deciding Q_1^w will begin by deciding the finiteness of R. If R is infinite, the answer is NO. Else, for every w in R the problem of whether or not $w \longrightarrow^* L_2$ equals R is decided. (Recall that, according to Theorem 3.8 and Corollary 3.12 the result of the iterated sequential deletion $w \longrightarrow^* L_2$ is regular and can be effectively constructed.) If such a w is found the answer is YES, else it is NO. □

Theorem 5.38 *If $\diamond$ denotes the controlled sequential deletion, the problem $Q_{1,\Delta}$ is decidable for regular languages R and regular control functions Δ.*

Proof. Let R be a regular language over an alphabet Σ and Δ be a regular function over Σ. Define

$$R' = (R^c \leftarrow\!\!\dashv \Delta)^c,$$

where $\leftarrow\!\!\dashv$ denotes the controlled sequential insertion.

(i) R' is a regular language and can be effectively constructed (see Theorem 2.18).

(ii) $R' \mapsto \Delta \subseteq R$. Assume the contrary and let $x \in R'$, $x = uawv$, $u, v, w \in \Sigma^*$, $a \in \Sigma$, $w \in \Delta(a)$ be words such that $uav \in R^c$. Then $x \in (uav \leftarrow\!\!\dashv \Delta) \subseteq R^c \leftarrow\!\!\dashv \Delta$, which contradicts the fact that $x \in R'$.

(iii) Any language L_1 with the property $L_1 \mapsto \Delta \subseteq R$ is included in R'. Assume the contrary and let x be a word in $L_1 - R'$. As x belongs to $R^c \leftarrow\!\!\dashv \Delta$ there exist words $z \in R^c$, $z = uav$, $u, v \in \Sigma^*$, $a \in \Sigma$, and $w \in \Delta(a)$, such that $x = uawv$. Consequently, $z \in (x \mapsto \Delta) \subseteq L_1 \mapsto \Delta \subseteq R$, which contradicts the fact that $z \in R^c$.

If there exists a language L_1 such that $L_1 \mapsto \Delta = R$ then, using (ii) and (iii), we can deduce that $R' \mapsto \Delta = R$. The algorithm for deciding our problem will consist of constructing R' and deciding whether or not $R' \mapsto \Delta = R$. □

Theorem 5.39 *If $\diamond$ denotes the controlled sequential deletion, the problem $Q^w_{1,\Delta}$ is decidable for regular languages R and regular control functions Δ.*

Proof. Analogous to that of Theorem 5.36. In the case of controlled deletion the set P is defined as

$$P = (R^c \leftarrow\!\!\dashv \Delta)^c - \bigcup_{S \subset R} (S^c \leftarrow\!\!\dashv \Delta)^c,$$

and the proof of Theorem 5.38 is used. □

The proofs of Theorem 5.38 and Theorem 5.39 can be used to show that for $\diamond$ denoting *sequential deletion next to a letter*, the problems Q_1 and Q^w_1 are decidable for regular languages L_2 and R.

The languages constructed in the proofs will be respectively:

$$\begin{aligned} R' = &\quad (R^c \overset{b}{\leftarrow\!\!\dashv} L_2)^c, \\ P = &\quad (R^c \overset{b}{\leftarrow\!\!\dashv} L_2)^c - \textstyle\bigcup_{S \subset R} (S^c \overset{b}{\leftarrow\!\!\dashv} L_2)^c. \end{aligned}$$

Theorem 5.40 *The problem "Does there exist a language L_1 such that $L_2 \backslash L_1 = R$?" is undecidable for context-free languages L_2 and regular languages R.*

Proof. Let Σ be an alphabet, $\mathrm{card}(\Sigma) \geq 2$, and let $\#, \$_1, \$_2$ be letters which do not occur in Σ. There exists a singleton language $R = \{\$_2\}$ such that the problem of the theorem is undecidable for context-free languages L_2. We assume the contrary and show how to solve the problem "Is $L - L' \neq \emptyset$?" for context-free languages L and L'. For given context-free L, L', define:

$$L_2 = \#(L \cup L')\$_1 \cup \#L'\$_1\$_2.$$

Claim. There exists a language L_1 such that $L_2 \backslash L_1 = R$ iff $L - L' \neq \emptyset$, where L_2 and R are defined as above.

"$\Longrightarrow$" Let L_1 be a language such that $L_2 \backslash L_1 = R$. As $R = \{\$_2\}$, $\$_2$ has to be a suffix of all the words in L_1. (Recall that we are talking about useful languages L_1.) Let us consider all the possible cases:
– If L_1 contains a word of the form $\#u\$_1\$_2\$_2$, $u \in L'$, then

$$\$_2\$_2 \in (\#u\$_1 \setminus \#u\$_1\$_2\$_2) \subseteq R.$$

– If L_1 contains a word of the form $\#u\$_1\$_2$, $u \in L'$, then

$$\lambda \in (\#u\$_1\$_2 \setminus \#u\$_1\$_2) \subseteq R.$$

Both possibilities lead to contradictions with the fact that $R = \{\$_2\}$. Consequently, L_1 does not contain such words. Taking into account the form of the words in L_2 and R, the only remaining possibility is that:

$$L_1 \subseteq \{\#u\$_1\$_2 |\ u \in L - L'\},$$

which further implies $L - L' \neq \emptyset$, since L_1 cannot be empty.

"$\Longleftarrow$" Assume that $L - L' \neq \emptyset$ and let u be a word in $L - L'$. The language $L_1 = \{\#u\$_1\$_2\}$ satisfies the relation $L_2 \backslash L_1 = R$. The second implication and therefore the proof of the claim is complete. □

The proof of the preceding theorem can be used to show that the problem "Does there exist a word w such that $L_2 \backslash w = R$" is undecidable for context-free languages L_2 and regular languages R.

An analogous proof can be used to show that the problems:
"Does there exist a language L_1 such that $L_1/L_2 = R$?"
"Does there exist a word w such that $w/L_2 = R$?"
are undecidable for context-free languages L_2 and regular languages R. The languages used in the proof will be respectively:

$$R = \{\$_2\},\ L_2 = \$_1(L \cup L')\# \cup \$_2\$_1L'\#.$$

Theorem 5.41 *If $\diamond$ denotes the sequential deletion, the problem Q_1^w is undecidable for context-free languages L_2 and regular languages R.*

Proof. Let Σ be an alphabet, $\mathrm{card}(\Sigma) \geq 2$, and let $\#_1, \#_2, \$_1, \$_2$ be letters which do not occur in Σ. There exists a finite language $R = \{\#_1, \$_2\}$ such that the problem Q_1^w is undecidable for context-free languages L_2. We assume the contrary and show how to solve the problem " Is $L \cap L' \neq \emptyset$?" for context-free languages L, L'. For given context-free languages $L, L' \subseteq \Sigma^*$, define the language:

$$L_2 = \#_1\#_2 L\$_1 \cup \#_2 L'\$_1\$_2.$$

Claim. There exists a word w such that $w \longrightarrow L_2 = R$ iff the intersection $L \cap L'$ is nonempty, where L_2 and R are defined as before.

"$\Longleftarrow$" Let u be a word in $L \cap L'$ and take $w = \#_1\#_2 u\$_1\$_2$. The following relations hold:

$$\#_1\#_2 u\$_1\$_2 \longrightarrow \#_1\#_2 u\$_1 = \{\$_2\},$$

$$\#_1\#_2 u\$_1\$_2 \longrightarrow \#_2 u\$_1\$_2 = \{\#_1\}.$$

Moreover, because of the markers, no other words of L_2 are subwords of w and therefore $w \longrightarrow L_2 = \{\#_1, \$_2\} = R$.

"$\Longrightarrow$" Let w be a word with the property $w \longrightarrow L_2 = R$. As $R = \{\#_1, \$_2\}$, either $\$_2$ or $\#_1$ is a prefix of w.

If $\$_2$ is a prefix of w then, necessarily, $\#_1$ is a suffix of w, and therefore w has the form $w = \$_2\alpha\#_1$. As $\{\$_2\} \subseteq w \longrightarrow L_2$, the word $\alpha\#_1$ has to be a subword of L_2 – a contradiction with the form of the words in L_2. Consequently, $\$_2$ is not a prefix of w.

If $\#_1$ is a prefix of w then, necessarily, $\$_2$ is a suffix of w, and therefore w has to be of the form $w = \#_1\alpha\$_2$. As $\{\$_2\} \subseteq w \longrightarrow L_2$ it results that w has the form $w = \#_1\#_2 u\$_1\$_2$ where u belongs to L. As $\{\#_1\} \subseteq w \longrightarrow L_2$ it results that w has the form $w = \#_1\#_2 u'\$_1\$_2$, where u' belongs to L'. We conclude that $u = u' \in L \cap L'$, which is therefore nonempty. The proof of the claim is complete.

The claim implies that the problem "Does there exist a word w such that:

$$w \longrightarrow (\#_1\#_2 L\$_1 \cup \#_2 L'\$_1\$_2) = \{\#_1, \$_2\}?".$$

amounts to the problem "Is $L \cap L' = \emptyset$?". □

Remark. The operations of insertion and deletion are associated with several notions rather basic in the combinatorics of words. While a more detailed study of such notions lies outside the scope of this Thesis, we want to briefly mention here one of them.

A language L is called a *deletion set* if

$$L = w \longrightarrow L',$$

for some word w and language L'. Clearly, every deletion set is finite. If m is the length of the longest word in a deletion set L, then L contains at most $(m+1)(m+2)/2$ words, this upper bound being the best possible in the general case. It is also not difficult to prove that it is decidable whether or not a given finite language is a deletion set. This result should be contrasted with the undecidability result of Theorem 5.41.

Theorem 5.42 *If $\diamond$ denotes the sequential deletion the problem Q_1 is undecidable for context-free languages L_2 and regular languages R.*

Proof. Let Σ be an alphabet, $\mathrm{card}(\Sigma) \geq 2$, and let $\$_1, \$_2, \#$ be letters which do not occur in Σ. There exists a singleton language $R = \{\$_2\}$ such that the problem Q_1 is undecidable for context-free languages L_2. We assume the contrary and show how to solve the problem "Is $L - L' = \emptyset$?" for context-free languages L and L'. Let L, L' be context-free languages and consider the language:

$$L_2 = \#(L \cup L')\$_1 \cup \#L'\$_1\$_2 \cup \$_2\#L'\$_1.$$

Claim. There exists a language L_1 such that $L_1 \longrightarrow L_2 = R$ iff $L - L' \neq \emptyset$, where L_2 and R are defined as before.

"$\Longrightarrow$" Let L_1 be a language such that $L_1 \longrightarrow L_2 = R$. Every word in L_1 has to be of the form $w\$_2$ or $\$_2 w$ where w is a word in L_2. (Recall that we are considering only useful languages L_1.) We take into account all the possible cases:

– If L_1 contains a word of the form $\#u\$_1\$_2\$_2$, $u \in L'$, then

$$\$_2\$_2 \in (\#u\$_1\$_2\$_2 \longrightarrow \#L'\$_1) \subseteq R.$$

– If L_1 contains a word of the form $\$_2\#u\$_1\$_2$, $u \in L'$, then

$$\$_2\$_2 \in (\$_2\#u\$_1\$_2 \longrightarrow \#L'\$_1) \subseteq R.$$

– If L_1 contains a word of the form $\$_2\$_2\#u\$_1$, $u \in L'$, then

$$\$_2\$_2 \in (\$_2\$_2\#u\$_1 \longrightarrow \#L'\$_1) \subseteq R.$$

– If L_1 contains a word of the form $\#u\$_1\$_2$, $u \in L'$, then

$$\lambda \in (\#u\$_1\$_2 \longrightarrow \#L'\$_1\$_2) \subseteq R.$$

– If L_1 contains a word of the form $\$_2\#u\$_1$, $u \in L'$, then

$$\lambda \in (\$_2\#u\$_1 \longrightarrow \$_2\#L'\$_1) \subseteq R.$$

All the mentioned cases led to contradictions with the fact that $R = \{\$_2\}$. Consequently, L_1 does not contain such words. Taking into account the form of the words in L_2 and R, the only remaining possibility is that:

$$L_1 \subseteq \{\#u\$_1\$_2 |\ u \in L - L'\} \cup \{\$_2\#u\$_1 |\ u \in L - L'\}.$$

This further implies that if L_1 with the desired property exists then $L - L' \neq \emptyset$.

"$\Longleftarrow$" Assume that $L - L' \neq \emptyset$ and let u be a word in $L - L'$. The language $L_1 = \{\#u\$_1\$_2\}$ satisfies the relation $L_1 \longrightarrow L_2 = R$. The proof of the claim is thus complete. □

Theorem 5.43 *If $\diamond$ denotes the controlled sequential deletion, the problem $Q_{1,\Delta}$ is undecidable for context-free control functions Δ and regular languages R.*

Proof. Let Σ be a language, $\text{card}(\Sigma) \geq 2$, and let $\#_1, \#_2, \$_1, \$_2$ be letters which do not occur in Σ. There exists a singleton language $R = \{\#_1\$_2\}$ such that the problem $Q_{1,\Delta}$ is undecidable for context-free control functions Δ. We assume the contrary and show how to solve the problem "Is $L - L' = \emptyset$?" for context-free languages L, L'.

For given L, L' as before consider the control function defined by:

$$\Delta(\#_1) = \#_2(L \cup L')\$_1 \cup \#_2 L'\$_1\$_2,\ \Delta(a) = \emptyset, \forall a \neq \#_1.$$

Claim. There exists a language L_1 such that $L_1 \longmapsto \Delta = R$, iff $L - L' \neq \emptyset$, where Δ, R are defined as before.

"$\Longrightarrow$" If L_1 would contain a word of the form $\#_1\#_2u\$_1\$_2\$_2$, $u \in L'$, then

$$\#_1\$_2\$_2 \in (\#_1\#_2u\$_1\$_2\$_2 \mapsto \Delta) \subseteq R,$$

– a contradiction.

If L_1 would contain a word of the form $\#_1\#_2u\$_1\$_2$, $u \in L'$, then

$$\#_1 \in (\#_1\#_2u\$_1\$_2 \mapsto \Delta) \subseteq R,$$

– a contradiction.

Consequently, the only possibility that remains is that

$$L_1 \subseteq \{\#_1\#_2u\$_1\$_2 |\ u \in L - L'\},$$

and, as $L_1 \neq \emptyset$, this implies $L - L' \neq \emptyset$.

"$\Longleftarrow$" If u is a word in $L - L'$ take $L_1 = \{\#_1\#_2u\$_1\$_2\}$. The language L_1 satisfies the equality $L_1 \mapsto \Delta = R$. The proof of the claim and therefore of the theorem is complete. □

The previous proof can be used to show that for $\diamond$ denoting *the controlled SD* the problem $Q^w_{1,\Delta}$ is undecidable for context-free control functions Δ and regular languages R.

Noticing that the control function Δ used in the previous proof has as value $\emptyset$ for all letters except $\#$, the proof can be used to show that, for $\diamond$ denoting *the SD next to a letter*, the problems Q_1 and Q^w_1 are undecidable for context-free languages L_2 and regular languages R.

Theorem 5.44 *If $\diamond$ denotes the scattered sequential deletion, the problems Q_1, Q^w_1 are undecidable for context-sensitive languages L_2 and regular languages R.*

Proof. We shall prove a stronger result: there exists a singleton language, $R = \{\lambda\}$, such that the problem Q_1 is undecidable for context-sensitive languages L_2. We assume the contrary and show how to solve the emptiness problem for context-sensitive languages. This follows noticing that the problem "Does there exist a language L_1 such that $L_1 \dashrightarrow L_2 = \{\lambda\}$?" is equivalent with the problem "Is $L_2 \neq \emptyset$?". Indeed, if $L_2 \neq \emptyset$ we can take $L_1 = \{w\}$, where w is one of the shortest words in L_2. It is easy to see that $\{w\} \dashrightarrow L_2 = \{\lambda\}$. The reverse implication is obvious. □

Appendix A

Operations: abbreviations and notations

Insertion Operations

Name of operation	**Abbreviation**	**Notation**
sequential insertion	SIN	$\leftarrow$
parallel insertion	PIN	$\Leftarrow$
iterated sequential insertion	iterated SIN	$\leftarrow^*$
iterated parallel insertion	iterated PIN	$\Leftarrow^*$
permuted sequential insertion	permuted SIN	$\leftsquigarrow$
permuted parallel insertion	permuted PIN	$\Leftsquigarrow$
controlled sequential insertion	controlled SIN	$\leftarrow\!\!\dashv$
left controlled sequential insertion	left controlled SIN	$\leftarrow\!\!<$
controlled parallel insertion	controlled PIN	$\Leftarrow\!\!\dashv$
left controlled parallel insertion	left controlled PIN	$\Leftarrow\!\!<$
iterated controlled sequential insertion	iterated controlled SIN	$\leftarrow\!\!\dashv^*$
iterated controlled parallel insertion	iterated controlled PIN	$\Leftarrow\!\!\dashv^*$
sequential insertion next to the letter a	SIN next to a	$\overset{a}{\leftarrow\!\!\dashv}$
parallel insertion next to the letter a	PIN next to a	$\overset{a}{\Leftarrow\!\!\dashv}$
shuffle		$\amalg$
permuted scattered (sequential) insertion	permuted scattered SIN	$\leftsquigarrow$
commutative closure		*com*

Deletion operations

Name of operation	**Abbreviation**	**Notation**
sequential deletion	SD	$\Rightarrow$
parallel deletion	PD	$\Rightarrow$
iterated sequential deletion	iterated SD	$\Rightarrow^*$
iterated parallel deletion	iterated PD	$\Rightarrow^*$
permuted sequential deletion	permuted SD	$\rightsquigarrow$
permuted parallel deletion	permuted PD	$\rightsquigarrow$
controlled sequential deletion	controlled SD	$\vdash\!\!\Rightarrow$
left controlled sequential deletion	left controlled SD	$\succ\!\!\Rightarrow$
controlled parallel deletion	controlled PD	$\models\!\!\Rightarrow$
left controlled parallel deletion	left controlled PD	$\succ\!\!\Rightarrow$
sequential deletion next to the letter a	SD next to a	$\overset{a}{\vdash\!\!\Rightarrow}$
parallel deletion next to the letter a	PD next to a	$\overset{a}{\models\!\!\Rightarrow}$
scattered (sequential) deletion	scattered SD	$\Rightarrow$
permuted scattered (sequential) deletion	permuted scattered SD	$\rightsquigarrow$
dipolar deletion		$\rightleftharpoons$

Appendix B

Closure properties

Insertion Operations

Closed under	**REG**	**CF**	**CS**
catenation	YES	YES	YES
SIN	YES	YES	YES
PIN	YES	YES	YES
iterated SIN	NO/NO	YES	YES
iterated PIN	NO/NO	NO (no/no)	YES
permuted SIN	NO/YES	NO (no/yes)	YES
permuted PIN	NO/YES	NO (no/yes)	YES
controlled SIN	YES	YES	YES
SIN next to a letter	YES	YES	YES
controlled PIN	YES	YES	YES
PIN next to a letter	YES	YES	YES
shuffle	YES	NO (yes/yes)	YES
permuted scattered SIN	NO/YES	NO (no/yes)	YES
iterated controlled SIN	NO/NO	YES	YES
iterated controlled PIN	NO/NO	NO (no/no)	YES

In all tables, dash stands for *unsettled*. For the other notations, see also the remarks on the next page.

Deletion Operations

Closed under	**REG**	**CF**	**CS**
left (right) quotient	YES	NO (yes/yes)	NO(no/yes)
SD	YES	NO (yes/yes)	NO (no/yes)
PD	YES	NO (yes/yes)	NO (no/no)
iterated SD	YES	NO (no/no)	NO (no/no)
iterated PD	–	NO (no/no)	NO (no/no)
permuted SD	YES	NO (no/yes)	NO (no/yes)
permuted PD	NO/YES	NO (no/yes)	NO (no/no)
controlled SD	YES	NO (yes/yes)	NO (no/yes)
SD next to a letter	YES	NO (yes/yes)	NO(no/yes)
controlled PD	YES	NO (yes/yes)	NO (no/yes)
PD next to a letter	YES	NO (yes/yes)	NO(no/yes)
scattered SD	YES	NO (yes/yes)	NO (no/yes)
permuted scattered SD	NO/YES	NO (no/yes)	NO (no/yes)
dipolar deletion	YES	NO (yes/yes)	NO (no/yes)

1. RE is closed under all the listed operations except PD, iterated PD, permuted PD, controlled PD, PD next to a letter.
2. In case REG is not closed under an operation, its closure under the operation with singletons is stated. For example, REG is not closed under permuted SIN, but it is closed under permuted SIN with singletons.
3. In case CF, CS are not closed under an operation, their closure under the operation with regular, respectively singleton languages is stated in parentheses. For example, CS is not closed under scattered SD, it is not closed under scattered SD with regular languages but it is closed under scattered SD with singletons.

Appendix C
Decision problems

Basic decision problems for insertion

Operation	**Problem**	**REG**	**CF**
catenation	$Q_0(Q_0^w)$	D (D)	U (U)
	$Q(Q^w)$	T (T)	U (U)
SIN	$Q_0(Q_0^w)$	D (D)	U (U)
	$Q(Q^w)$	T (T)	U (U)
PIN	$Q_0(Q_0^w)$	D (D)	U (U)
	$Q(Q^w)$	T (T)	U (U)
iterated SIN	$Q_0(Q_0^w)$	– (–)	U (U)
	$Q(Q^w)$	– (–)	U (U)
iterated PIN	$Q_0(Q_0^w)$	– (–)	U (U)
	$Q(Q^w)$	– (–)	U (U)
permuted SIN	$Q_0(Q_0^w)$	– (D)	U (U)
	$Q(Q^w)$	– (T)	U (U)
permuted PIN	$Q_0(Q_0^w)$	– (D)	U (U)
	$Q(Q^w)$	– (T)	U (U)
controlled SIN	$Q_{0,\Delta}(Q_{0,\Delta}^w)$	D (D)	U (U)
	$Q_\Delta(Q_\Delta^w)$	T (T)	U (U)
controlled PIN	$Q_{0,\Delta}(Q_{0,\Delta}^w)$	D (D)	U (U)
	$Q_\Delta(Q_\Delta^w)$	T (T)	U (U)
SIN next to a letter	$Q_0(Q_0^w)$	D (D)	U (U)
	$Q(Q^w)$	T (T)	U (U)
PIN next to a letter	$Q_0(Q_0^w)$	D (D)	U (U)
	$Q(Q^w)$	T (T)	U (U)
shuffle	$Q_0(Q_0^w)$	D (D)	U (U)
	$Q(Q^w)$	T (T)	U (U)
permuted scattered SIN	$Q_0(Q_0^w)$	– (D)	U (U)
	$Q(Q^w)$	– (T)	U (U)

In all the following tables, U means *undecidable*, *D decidable*, T *true for the whole class*, and dash *unsettled*. For the meaning of the symbols Q_0, Q_0^w, $Q_{0,\Delta}$, $Q_{0,\Delta}^w$, Q, Q^w, Q_Δ and Q_Δ^w, see Section 5.1 .

Basic decision problems for deletion

Operation	**Problem**	**REG**	**CF**
right(left) quotient	$Q_0(Q_0^w)$	D (D)	U (U)
	$Q(Q^w)$	T (T)	U (U)
SD	$Q_0(Q_0^w)$	D (D)	U (U)
	$Q(Q^w)$	T (T)	U (U)
PD	$Q_0(Q_0^w)$	D (D)	U (U)
	$Q(Q^w)$	T (T)	U (U)
iterated SD	$Q_0(Q_0^w)$	D (D)	U (U)
	$Q(Q^w)$	T (T)	U (U)
iterated PD	$Q_0(Q_0^w)$	– (–)	U (U)
	$Q(Q^w)$	– (–)	U (U)
permuted SD	$Q_0(Q_0^w)$	– (D)	U (U)
	$Q(Q^w)$	T (T)	U (U)
permuted PD	$Q_0(Q_0^w)$	– (D)	U (U)
	$Q(Q^w)$	– (T)	U (U)
controlled SD	$Q_{0,\Delta}(Q_{0,\Delta}^w)$	D (D)	U (U)
	$Q_\Delta(Q_\Delta^w)$	T (T)	U (U)
controlled PD	$Q_{0,\Delta}(Q_{0,\Delta}^w)$	D (D)	U (U)
	$Q_\Delta(Q_\Delta^w)$	T (T)	U (U)
SD next to a letter	$Q_0(Q_0^w)$	D (D)	U (U)
	$Q(Q^w)$	T (T)	U (U)
PD next to a letter	$Q_0(Q_0^w)$	D (D)	U (U)
	$Q(Q^w)$	T (T)	U (U)
scattered SD	$Q_0(Q_0^w)$	D (D)	U (U)
	$Q(Q^w)$	T (T)	U (U)
permuted scattered SD	$Q_0(Q_0^w)$	– (D)	U (U)
	$Q(Q^w)$	– (T)	U (U)

See Section 5.1 for the meaning of the symbols Q_0, Q_0^w, $Q_{0,\Delta}$, $Q_{0,\Delta}^w$, Q, Q^w, Q_Δ, Q_Δ^w and the preceding page for the meaning of U, D, T, –.

Operand problems for insertion

Operation	**Problem**	**REG**	**CF**
catenation	$Q_1(Q_1^w)$	D (D)	U (U)
	$Q_2(Q_2^w)$	D (D)	U (U)
SIN	$Q_1(Q_1^w)$	D (D)	U (U)
	$Q_2(Q_2^w)$	D (D)	U (U)
PIN	$Q_1(Q_1^w)$	– (D)	U (U)
	$Q_2(Q_2^w)$	– (D)	U (U)
iterated SIN	$Q_1(Q_1^w)$	– (–)	– (–)
	$Q_2(Q_2^w)$	– (–)	U (U)
iterated PIN	$Q_1(Q_1^w)$	– (–)	– (–)
	$Q_2(Q_2^w)$	– (–)	U (U)
permuted SIN	$Q_1(Q_1^w)$	– (–)	– (–)
	$Q_2(Q_2^w)$	– (D)	U (U)
permuted PIN	$Q_1(Q_1^w)$	– (–)	– (–)
	$Q_2(Q_2^w)$	– (D)	U (U)
controlled SIN	$Q_{1,\Delta}(Q_{1,\Delta}^w)$	D(D)	U (U)
	$Q_{2,\Delta}(Q_{2,\Delta}^w)$	D (–)	U (U)
controlled PIN	$Q_{1,\Delta}(Q_{1,\Delta}^w)$	– (D)	U (U)
	$Q_{2,\Delta}(Q_{2,\Delta}^w)$	– (–)	– (–)
SIN next to a letter	$Q_1(Q_1^w)$	D (D)	U (U)
	$Q_2(Q_2^w)$	D (D)	U (U)
PIN next to a letter	$Q_1(Q_1^w)$	– (D)	U (U)
	$Q_2(Q_2^w)$	– (D)	U (U)
shuffle	$Q_1(Q_1^w)$	D (D)	U (U)
	$Q_2(Q_2^w)$	D (D)	U (U)
permuted scattered SIN	$Q_1(Q_1^w)$	– (–)	– (–)
	$Q_2(Q_2^w)$	– (D)	U (U)

See Section 5.2 (5.3) for the meaning of the symbols Q_2, Q_2^w, $Q_{2,\Delta}$, $Q_{2,\Delta}^w$ (Q_1, Q_1^w, $Q_{1,\Delta}$ and $Q_{1,\Delta}^w$, respectively) and the table "Basic decision problems for insertion" for the meaning of U, D, T, –.

Operand problems for deletion

Operation	**Problem**	**REG**	**CF**
right(left) quotient	$Q_1(Q_1^w)$	D (D)	U (U)
	$Q_2(Q_2^w)$	D (–)	U (U)
SD	$Q_1(Q_1^w)$	D (D)	U (U)
	$Q_2(Q_2^w)$	D (–)	U (U)
PD	$Q_1(Q_1^w)$	– (–)	– (–)
	$Q_2(Q_2^w)$	– (–)	U(U)
iterated SD	$Q_1(Q_1^w)$	– (D)	– (–)
	$Q_2(Q_2^w)$	– (–)	U (U)
iterated PD	$Q_1(Q_1^w)$	– (–)	– (–)
	$Q_2(Q_2^w)$	– (–)	U (U)
permuted SD	$Q_1(Q_1^w)$	– (–)	– (–)
	$Q_2(Q_2^w)$	– (–)	U (U)
permuted PD	$Q_1(Q_1^w)$	– (–)	– (–)
	$Q_2(Q_2^w)$	– (–)	U (U)
controlled SD	$Q_{1,\Delta}(Q_{1,\Delta}^w)$	D (D)	U (U)
	$Q_{2,\Delta}(Q_{2,\Delta}^w)$	D (–)	U (U)
controlled PD	$Q_{1,\Delta}(Q_{1,\Delta}^w)$	– (–)	– (–)
	$Q_{2,\Delta}(Q_{2,\Delta}^w)$	– (–)	– (U)
SD next to a letter	$Q_1(Q_1^w)$	D (D)	U (U)
	$Q_2(Q_2^w)$	D (–)	U (U)
PD next to a letter	$Q_1(Q_1^w)$	– (–)	– (–)
	$Q_2(Q_2^w)$	– (–)	U (U)
scattered SD	$Q_1(Q_1^w)$	D (D)	– (–)
	$Q_2(Q_2^w)$	D (–)	U (U)
permuted scattered SD	$Q_1(Q_1^w)$	– (–)	– (–)
	$Q_2(Q_2^w)$	– (–)	U (U)

See Section 5.4 (5.5) for the meaning of the symbols Q_2, Q_2^w, $Q_{2,\Delta}$ and $Q_{2,\Delta}^w$ (Q_1, Q_1^w, $Q_{1,\Delta}$ and $Q_{1,\Delta}^w$, respectively) and the table " Basic decision problems for insertion" for the meaning of U, D, T, – .

Bibliography

[1] M.Andraşiu, A.Atanasiu, Gh.Păun, A.Salomaa. A new cryptosystem based on formal language theory. *Bull.Math.Soc.Sci.Math.Roumanie*, to appear.

[2] M.Andraşiu, Gh.Păun, A.Salomaa. Language-theoretical problems arising from Richelieu cryptosystems. Submitted for publication.

[3] S.Ginsburg. *Algebraic and Automata-Theoretic Properties of Formal Languages*. North-Holland, Amsterdam, 1975.

[4] M.A.Harrison. *Introduction to Formal Language Theory*. Addison Wesley, Reading, Massachusetts, 1978.

[5] J.Hopcroft, J.Ulmann. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, Reading, Massachusetts, 1979.

[6] H.C.M.Kleijn, G.Rozenberg. Context-free like restrictions on selective rewriting. *Theoretical Computer Science*, vol.16, no.3(1981), pp.237-269.

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

[8] R.C.Lyndon, M.P.Schutzenberger. The equation $a^M = b^N c^P$ in a free group, *Michigan Math.J.*, no.9(1962), pp.289-298.

[9] Gh.Păun, A.Salomaa. Semi-commutativity sets – a cryptographically grounded topic. *Bull.Math.Soc.Sci.Math.Roumanie*, to appear.

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

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

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

[13] L.Sântean[1]. Six arithmetic-like operations on languages. *Revue Roumaine de Linguistique*, Tome XXXIII, 1988, Cahiers de linguistique theorique et applique, Tome XXV, 1988, No.1, Janvier-Juin, pp.65-73.

[14] H.J.Shyr. *Free Monoids and Languages*. Lecture Notes, Institute of applied mathematics, National Chung-Hsing University, Taichung, Taiwan, 1991.

[15] H.J.Shyr, G.Thierrin, S.S.Yu. Monogenic e-closed languages and dipolar words. To appear.

[1]The maiden name of the author of this Thesis