SIX ARITHMETIC-LIKE OPERATIONS ON LANGUAGES

LILA SÂNTEAN

Operations on languages are intensively studied in formal language theory. For example, there are representations of some families of languages starting from simpler languages and using suitable operations, finding of counterexamples often uses operations on languages, the theory of abstract families of languages (AFL) studies just operations, many operations appear in formal language theory applications [1], and so on.

The existing operations can be roughly clustered in three classes : set operations (union, intersection, complementation), algebraic operations (homomorphism, substitution) and purely language theoretical operations (Kleene closure, shuffle). Within this frame, it is obvious to ask for language operations corresponding to the arithmetic operations on numbers : sum, product, power, factorial, square root, and so on. Six such operations will be defined and investigated in the following, namely the compact subtraction, the literal subtraction, the generalized subtraction, the multiplication, the power, and the factorial.

The aim of this paper is to examine the closure of an abstract family of languages (when positive results are true) or directly of families in Chomsky hierarchy (when negative results hold) under these operations.

Generally, the results are the expected ones, in the sense that the family of context-sensitive languages is not closed under erasing operations, whereas for the families of context-free and regular languages, the situation is just the opposite.

1. Compact subtraction

For a vocabulary V, we denote by V^* the free monoid generated by V under the concatenation operation; the null element of V is λ and $|x|$ denotes the length of the string $x \in V^*$. The four families in Chomsky hierarchy are denoted by $\mathscr{L}_i$, i $= 0, 1, 2, 3$ ($\mathscr{L}_{lin}$ denotes the family of linear languages). For other notation and terminologies in formal language theory, the reader is referred to [2].

Definition 1.1. Let L_1, L_2 be languages on V^*. We define the ***compact subtraction*** of L_1 and L_2 by :

$$L_1 \ominus L_2 = \bigcup_{\substack{x \in L_1 \\ y \in L_2}} (x \ominus y), \text{ where}$$

$$x \ominus y = \{z \in V^* / z = x_1 x_2,\ x = x_1 y x_2\}.$$

Compact subtraction is a generalization of right or left quotient : instead of extracting the word y from the left or right extremity of x, we extract it from an arbitrary place in x.

Theorem 1.1. *$\mathscr{L}_1$ is not closed under compact subtraction.*

Proof. If L_1, L_2 are two languages on V^*, we notice that $\{c\}L_1 \ominus \{c\}L_2 = L_2 \setminus L_1$, where c is a symbol which doesn't belong to V.

As the family $\mathscr{L}_1$ is not closed under left quotient with regular languages, it follows that it its not closed under operation $\ominus$, either.

Theorem 1.2. *If L_1 and L_2 are languages on V^*, L_2 a regular one, then there is a gsm g (with erasing) so that $L_1 \ominus L_2 = g(L_1)$.*

Proof. Let $A = (K, V, s_0, F, P)$ be a finite automaton that recognizes L_2. We construct the *gsm*:

$$g = (V, V, K \cup \{s'_0, s_f\}, s'_0, \{s_f\}, P'), \text{ where}$$

$$P' = \{s'_0 a \to a s'_0 / a \in V\} \cup P \cup \{s'_0 a \to s / s_0 a \to s \in P\}$$

$$\cup \{sa \to s_f / sa \to s' \in P, s' \in F\} \cup \{s_f a \to a s_f / a \in V\}$$

$$\cup \{s'_0 a \to s_f / s_0 a \to s \in P, s \in F\} \cup \{s'_0 a \to a s_f / a \in V, \lambda \in L_2\}$$

Clearly, $g(L_1) = L_1 \ominus L_2$ and thus the proof is finished.

Corollary. *$\mathscr{L}_2$, $\mathscr{L}_{lin}$, $\mathscr{L}_3$ are closed under compact subtraction with regular languages.*

Open problems: The closure of the families $\mathscr{L}_2$ and $\mathscr{L}_{lin}$ under compact subtraction.

Probably, these families are not closed under compact subtraction, or, if they are, this result cannot be proved in a constructive way, because we have:

Theorem 1.3. *There is no algorithm to decide whether $L_1 \ominus L_2$ is empty or not, for L_1, L_2 arbitrary in $\mathscr{L}_{lin}$.*

Proof. Let us consider the linear languages

$$L_1 = \{ada^{i_k}b \ldots ba^{i_2}ba^{i_1}bcx_{i_1}x_{i_2} \ldots x_{i_k}d / k \geq 1, i_1, i_2, \ldots, i_k \in \{1, 2, \ldots, n\}\}$$

$$L_2 = \{da^{i_k}b \ldots ba^{i_2}ba^{i_1}bc\, y_{i_1}y_{i_2} \ldots y_{i_k}d / k \geq 1, i_1, i_2, \ldots, i_k \in \{1, 2, \ldots, n\}\}.$$

The statement: $L_1 \ominus L_2 \neq \emptyset$ iff there is a sequence of indexes $i_1, i_2, \ldots, i_k \in \{1, 2, \ldots, n\}$ so that $x_{i_1}x_{i_2} \ldots x_{i_k} = y_{i_1}y_{i_2} \ldots y_{i_k}$, is obvious.

Therefore, we have $L_1 \ominus L_2 \neq \emptyset$ iff the POST correspondence problem has a solution, which is undecidable.

Concluding, we cannot construct in an algorithmic way a context-free grammar G so that $L(G) = L_1 \ominus L_2$, L_1, $L_2 \in \mathscr{L}_{lin}$, as, otherwise we can decide if $L_1 \ominus L_2 = \emptyset$ (the problem if $L(G)$ is empty, finite or infinite is decidable for context-free grammars) — contradiction.

2. Literal subtraction

Definition 2.1. Let L_1, L_2 be languages on V^*. We define the *literal subtraction*, $L_1 \mathbin{-\!\cdot} L_2$, by

$$L_1 \mathbin{-\!\cdot} L_2 = \bigcup_{\substack{x \in L_1 \\ y \in L_2}} (x \mathbin{-\!\cdot} y), \text{ where}$$

$$x \mathbin{-\!\cdot} y = \{x_1 x_2 \ldots x_k / x_1 b_1 x_2 b_2 \ldots b_{k-1} x_k = x, b_1 b_2 \ldots b_{k-1} = y, k \geq 2,$$

$b_i \in V$, $i \in \{1, 2, \ldots, k-1\}$, $x_j \in V^*$, $j \in \{1, 2, \ldots, k\}$ (if the letters of y can also be found in x, in the same order, then the literal subtraction erases them from x, without taking into account their places; else we cannot subtract y from x).

Theorem 2.1. If L_2 is a regular language, then the literal subtraction $L_1 -\cdot L_2$ can be attained by a gsm (with erasing).

Proof. Let $A = (K, V, s_0, F, P)$ be a finite automaton that recognizes the language L_2 (therefore P contains rules of the form $sa \to s'$, $s, s' \in K$, $a \in V$).

We construct the *gsm* $g = (V, V, K, s_0, F, P')$ with K, V, s_0, F according to A and $P' = P \cup \{sa \to as / s \in K,\ a \in V\}$.

One can easily prove that $L_1 -\cdot L_2 = g(L_1)$ (the rules of P erase the symbols which come from y, in the correct order, and those of the form $sa \to as$ cross the symbols that will remain in $x -\cdot y$).

Corollary. $\mathscr{L}_3$, $\mathscr{L}_{lin}$, $\mathscr{L}_2$ are closed under literal subtraction with regular languages.

Theorem 2.2. $\mathscr{L}_1$ is not closed under literal subtraction with regular languages.

Proof. We define the *gsm* $g = (V, V \cup V', K, s_0, F, P')$, where $K = \{s_0, s\}$, $F = \{s\}$, $V' = \{a' / a \in V\}$, $P' = \{s_0 a \to a s_0 / a \in V\} \cup \{s_0 a \to a' s / a \in V\} \cup \{sa \to a' s / a \in V\}$.

If $L \subseteq V^*$, we have the relation:

$g(L) = \{w_1 w_2' / w_1 w_2 \in L\}$ (the *gsm* g marks the symbols that are situated on the right side of the strings of L).

We also have the relation:

$L_1 / L_2 = [g(L_1) -\cdot h(L_2)] \cap V^*$, where $L_1, L_2 \subseteq V^*$ and h is a homomorphism, $h : V^* \to V'^*$, $h(a) = a'$.

As $\mathscr{L}_1$ is closed under intersection but it is not closed under right (and left) quotient with regular languages, it follows that $\mathscr{L}_1$ is not closed under operation $-\cdot$

Theorem 2.3. $\mathscr{L}_2$ and $\mathscr{L}_{lin}$ are not closed under literal subtraction with linear languages.

Proof. Let L_1, L_2 be the linear languages

$$L_1 = \{a^n (bc)^n (df)^m / n,\ m \geqslant 1\},$$

$$L_2 = \{c^n d^n / n \geqslant 1\}.$$

One can easily see that:

$$[L_1 -\cdot L_2] \cap \{a\}^* \{b\}^* \{f\}^* = \{a^n b^n f^n / n \geqslant 1\}.$$

As $\mathscr{L}_2$ and $\mathscr{L}_{lin}$ are closed under intersection by regular sets but $\{a^n b^n f^n / n \geqslant 1\}$ is not a context-free language, it follows that these families are not closed under literal subtraction.

In fact, we have obtained a stronger result, namely that there are linear languages L_1, L_2 such that $L_1 -\cdot L_2$ is not a context-free language.

3. Generalized subtraction

Definition 3.1. Let L_1, L_2 be languages on V^*. We define the *generalized subtraction* $L_1 \not- L_2$ by :

$$L_1 \not- L_2 = \bigcup_{\substack{x\in L_1\\ y\in L_2}} (x \not- y), \text{ where}$$

$x \not- y = \{x_1x_2 \ldots x_{k+1}/x = x_1b_1x_2b_2 \ldots x_kb_kx_{k+1}$, where y is a permutation of the word $b_1b_2 \ldots b_k$, $k \geqslant 1\}$ (if the letters of y can also be found in x, then the generalized subtraction erases the letters of y from x without taking into account their places; else we cannot subtract y from x). Notice that the generalized subtraction is a generalization of the compact and literal subtraction.

Theorem 3.1. $\mathscr{L}_3$ is not closed under generalized subtraction.

Proof. Let L_1, L_2 be the regular languages

$$L_1 = \{(bc)^m\, (df)^p/m,\, p \geqslant 1\},$$

$$L_2 = \{(cd)^n/n \geqslant 0\}.$$

One can prove that

$$(L_1 \not- L_2) \cap \{b\}^*\{f\}^* = \{b^mf^m/m \geqslant 1\}.$$

As $\mathscr{L}_3$ is closed under intersection by regular languages but $\{b^mf^m/m \geqslant 1\}$ is not regular, it follows that $\mathscr{L}_3$ is not closed under operation $\not-$.

Theorem 3.2. $\mathscr{L}_{lin}$, $\mathscr{L}_2$ are not closed under generalized subtraction with regular languages.

Proof. Let L_1, L_2 be the linear languages :

$$L_1 = \{a^n(bc)^n\,(df)^m/n,\, m \geqslant 1\},$$

$$L_2 = \{(cd)^n/n \geqslant 1\}.$$

The relation

$$(L_1 \not- L_2) \cap \{a\}^*\{b\}^*\{f\}^* = \{a^nb^nf^n/n \geqslant 1\} \text{ is obvious.}$$

As $\mathscr{L}_2$ and $\mathscr{L}_{lin}$ are closed under intersection by regular languages but $\{a^nb^nc^n/n \geqslant 1\}$ is not context-free, it follows that $\mathscr{L}_{lin}$ and $\mathscr{L}_2$ are not closed under generalized subtraction with regular languages.

Theorem 3.3. $\mathscr{L}_1$ is not closed under generalized subtraction with regular sets.

Proof. For each $L_0 \in \mathscr{L}_0$(hence also for $L_0 \in \mathscr{L}_0 - \mathscr{L}_1$, $L_0 \subseteq V^*$), there is $L_1 \in \mathscr{L}_1$, $L_1 \subseteq a^*bL_0$, $a, b \notin V$, such that for each $x \in L_0$ there is a natural n such that $a^nbx \in L_1$ ([2]). Consider such a language $L_1 \in \mathscr{L}_1$. We have

$$L_0 = (L_1 \not- a^*b) \cap V^*.$$

As $\mathscr{L}_1$ is closed under intersection, it follows that it connot be closed under generalized subtraction with regular sets.

4. Multiplication

Definition 4.1. Let L_1, L_2 be languages on V^*. We define their *multiplication* by :

$L_1 * L_2 = \{x^{|y|}/x \in L_1, y \in L_2\}$ on condition that $\lambda^{|\alpha|} = \lambda$, $\alpha \in L_2$ and $\beta^{|\lambda|} = \lambda$, $\beta \in L_1$.

Theorem 4.1. $\mathscr{L}_3$ is not closed under multiplication.

Proof. Let L_1, L_2 be the regular languages

$$L_1 = \{a^n b/n \geqslant 1\},$$

$$L_2 = \{aaa\}.$$

In accordance with definition 4.1 we have

$L_1 * L_2 = \{a^n b a^n b a^n b/n \geqslant 1\}$, which is not even context-free.

Corollary. The families $\mathscr{L}_2$ and $\mathscr{L}_{lin}$ are not closed under multiplication.

Theorem 4.2. $\mathscr{L}_1$ is closed under multiplication.

Proof. A standard (straightforward, but long) construction would prove this statement ; we omit the details. For a similar proof, see theorem 5.2, below.

5. Power

Definition 5.1. If L_1 and L_2 are languages on V^*, we define $L_1{**}L_2$ (L_1 *power* L_2) by :

$$L_1{**}L_2 = \{x_1^{|x_2|\cdot|x_3|\cdots|x_{|z|}|}/x_i \in L_1,\ 1 \leqslant i \leqslant z,\ z \in L_2\}$$

on condition that if $\lambda \in L_1$ or $\lambda \in L_2$, we put λ in $L_2{**}L_1$}.

*Theorem 5.1. $\mathscr{L}_3$ is not closed under operation**.*

Proof. Let L_1, L_2 be the regular languages :

$$L_1 = \{aa\},$$

$$L_2 = \{a^n/n \geqslant 1\}.$$

Then, $L_1{**}L_2 = \{a^{2^n}/n \geqslant 1\}$, language that is not even context-free.

*Corollary. $\mathscr{L}_2$, $\mathscr{L}_{lin}$ are not closed under operation **.*

Theorem 5.2. If $L_1 \subseteq V^2V^*$, $L_2 \subseteq V^*$, $L_1, L_2 \in \mathscr{L}_1$, then $L_1{**}L_2 \in \mathscr{L}_1$.

Proof. Let L_1, L_2 be two languages which satisfy the requested conditions, and G_1, G_2 the generating grammars :

$$G_1 = (V_N^1, V_T^1, S_1, P_1)$$

$$G_2 = (V_N^2, V_T^2, S_2, P_2).$$

We construct the grammar $G = (V_N, V_T, S, P)$, where $V_N = V_N^1 \cup \cup V_N^2 \cup V_T^2 \cup \{S, C, E, A, C', C'', C''', B, D, H, B', B'', F, I\}$, $V_T = V_T^1$ and P is constructed as follows:

P contains $P_1 \cup P_2$, on condition that if P_1 or P_2 contain rules giving the null word, we eliminate them from P, and introduce in its stead the rule $S \to \lambda$.

Moreover, we shall add to P the rules (1)–(24), which will be explained in the sequel.

First, we generate $z \in L_2$, bordering it with FC to its left, and with E to its right:

(1) $S \to FCS_2E$.

Because we want to obtain $|z|$ words from L_1, we change every letter of z into S_1, separating them by A:

(2) $Ca \to CS_1A, \qquad a \in V_T^2$

(3) $Aa \to AS_1A, \qquad a \in V_T^2$

(4) $AaE \to AS_1E, \qquad a \in V_T^2$.

Deriving on with rules from P_1 we get:

$$FCx_1Ax_2A \ldots Ax_{|z|}E, \; x_i \in L_1.$$

Now, we try to obtain the word $x_1^{|x_2|} = y$, then $y^{|x_3|} = x_1^{|x_2||x_3|}$, and so on, till we get $x_1^{|x_2|\cdots|x_{|z|}|}$.

During the first step, we work only with the first two words. Thus, we limit the working zone:

(5) $C \to C'C''$.

F marks the left extremity of the whole word, C' the left extremity of x_1, and C'' the other limits, in the following way: C'' goes to the right, crossing only the terminals; when it meets the left extremity of x_2, it points this out by turning itself into C''' and A into B; C''' goes to the right, and, when it meets the right extremity of x_2, it turns A into D, if x_2 is not the last word, or E into H, if x_2 is the last word, and disappears:

(6) $C''a \to aC'', \qquad a \in V_T^1$

(7) $C''A \to BC'''$

(8) $C'''A \to aC''', \qquad a \in V_T^1$

(9) $C'''A \to D$

(10) $C'''E \to H$

After using these rules we get either the word

$$FC'\underbrace{a_1a_2 \ldots a_n}_{x_1}B\underbrace{b_1b_2 \ldots b_m}_{x_2}D, \text{ or } FC'x_1Bx_2H.$$

To obtain $x_1^{|x_2|}$ we have to generate a word x_1 for every letter of x_2. We bring a letter b to the left of B, marking it:

(11) $Bb \to b'B, \qquad b \in V_T^1.$

This b' goes to the left, adding a marked lining to every letter of x_1, and disappears when attaining the extremity of x_1:

(12) $ab' \to b'a''a, \qquad a, b \in V_T^1$

(13) $C'ab' \to C'a''a, \qquad a \in V_T^1,\ b \in V_T^1.$

The marked symbols move to the left, in order, and when they attain C', cross it, loosing their marks:

(14) $ba'' \to a''b, \qquad a, b \in V_T^1$

(15) $C'a'' \to aC', \qquad a \in V_T^1.$

After using these rules we obtain either

$$Fa_1a_2 \ldots a_nC'a_1a_2 \ldots a_nBb_1b_2 \ldots b_mD \text{ or}$$

$$Fa_1a_2 \ldots a_nC'a_1a_2 \ldots a_nBb_1b_2 \ldots b_mH.$$

We repeat these rules for every letter b_i. When we reach the last one, we destroy it:

(16) $BbD \to B'A, \qquad b \in V_T^1.$

Afterwards, if x_2 is not the last word of L_1 — in the word we are talking about —, to use the set of rules (5)—(16) again, we must bring the current word to the initial form:

(17) $aB' \to B'a, \qquad a \in V_T^1$

(18) $C'B' \to B''.$

We continue the moving of B'' to the left, until it reaches F:

(19) $aB'' \to B''a, \qquad a \in V_T^1$

(20) $FB'' \to FC.$

Now, the current word is:

$$FCx_1^{|x_2|}Ax_3A \ldots Ax_kE,$$

and we can resume the set of rules, beginning with (5).

If x_2 is the last occurrence of a word of L_1 in the current word, then the last b_i disappears, and B and H turn into I, which moves to the left, erasing all nonterminals:

(21) $BbH \to I, \qquad b \in V_T^1$

(22) $aI \to Ia, \qquad a \in V_T^1$

(23) $C'I \to I$

(24) $FI \to \lambda.$

From the above explanations, it easily results that $L(G) = L_1 ** L_2$.

To show that $L(G) \in \mathscr{L}_1$ we shall use the work-space theorem ([2]).

Let z be a word in $L(G)$, $z \neq \lambda$, $z = x_1^{|x_2| \cdot |x_3| \cdots |x_{|y|}|}$, and a derivation $D : w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n = z$.

The only places $w \Rightarrow w'$ where we can have $|w'| < |w|$ are the places where we apply :

— rule (9), (16) or (18); each of them decreases w with one letter and can be applied $|y| - 1$ times in D.

— rule (10), (23), or (24); each of them decreases w with one letter and can be applied once in D.

— rule (21) which decreases w with two letters and can be applied once in D.

Consequently, we conclude that the greatest length of a word in D cannot be larger than $|z| + 3(|y| - 1) + 2 + 2 + 1$.

For $k = 4$, and taking into account that the words from L_1 have the length greater or equal to two, we have :

$$WS(z, G) \leqslant \min_{D} WS(D, G) \leqslant WS(D, G) = \max_{1 \leqslant i \leqslant n} |w_i| =$$

$$= |z| + 3|y| + 1 + 1 \leqslant 2^{|y|} \cdot k \leqslant k|x_1||x_2| \ldots |x_{|y|}| = k|z|.$$

According to the work-space theorem, $L(G) = L_1 ** L_2 \in \mathscr{L}_1$.

Open problem. Is $\mathscr{L}_1$ closed under operation $**$?

6. Factorial

Definition 6.1. Let L be a language on V^*. We define L *factorial* by :

$$L! = \{x! / x \in L\} \text{ where, if } x = a_1 a_2 \ldots a_n, \text{ then}$$

$x! = a_1\, a_1 a_2\, a_1 a_2 a_3 \ldots a_1 a_2 a_3 \ldots a_n$, on condition that $\lambda! = \lambda$ and $a! = a$, $a \in V$.

Theorem 6.1. $\mathscr{L}_3$ is not closed under operation !.

Proof. Let L be the regular language $L = \{a^n / n \geqslant 1\}$.

In accordance to definition 6.1, $L! = \{a^{n(n+1)/2} / n \geqslant 1\}$, language which is not even context-free.

Corollary. $\mathscr{L}_2$, $\mathscr{L}_{lin}$ are not closed under operation !.

Theorem 6.2. $\mathscr{L}_1$ is closed under operation !.

Proof. Let L be a language in $\mathscr{L}_1$, and $G = (V_N, V_T, S, P)$ the generating grammar.

Let G' be a grammar, $G' = (V'_N, V'_T, S', P')$, where $V'_N = \{S', X_0, X_1, X_2\} \cup V_N$, $V'_T = V_T \cup \{c\}$ and P' is constructed as follows :

P' contains P. We shall also introduce into P' the rules (1)—(7) constructed in the following way :

First, we produce a word from L :

(1) $S' \rightarrow X_0 X_1 S X_2$.

A derivation will continue only with rules from P, until we obtain $X_0X_1xX_2$, $x \in L$. Assuming that $x = a_1a \ldots a_n$, we'll try to produce a lining of the first $n-1$ letters. If on the right side of X_1 there are at least two letters, the first one passes on the left side of X_1 and produces a marked lining :

(2) $X_1ab \to a'aX_1b, \quad a, b \in V_T$

When we attain the last letter of x, which must not be doubled, we pass it to the right side of X_2, and point this out by marking X_1 :

(3) $X_1aX_2 \to X_1'X_2a, \quad a \in V_T$.

All the unmarked symbols pass, in order, to the right side of X_2 :

(4) $ad \to da, \quad a \in V_T, d \in \{a'/a \in V_T\} \cup \{X_1', X_2\}$.

In this moment, on the right side of $X_1'X_2$ we have the initial word, and on the left side, the first $n-1$ letters, marked.

We have to repeat the preceding operations and, with this end in view, we move X_1' to the left, until it reaches the extremity, when it turns back into X_1. In this way, X_1' erases all the marks, so that, when it reaches the left extremity and becomes X_1, we can repeat our method for the $n-1$ letters between X_0X_1 and X_2 :

(5) $a'X_1' \to X_1'a, \quad a \in V_T$

(6) $X_0X_1' \to X_0X_1$.

Finally, when we have no more letters between X_0X_1 and X_2 :

(7) $X_0X_1X_2 \to ccc$.

One can easily see, from the above explanations, that $L(G') = \{ccc\}\ L!$. $L(G')$ is, clearly context-sensitive.

Let h be the homomorphism $h : (V_T \cup \{c\})^* \to V_T^*$, defined by

$$h(a) = a,\ a \in V_T,\ h(c) = \lambda.$$

We have that $h(L(G')) = L!$.

$\mathscr{L}_1$ is closed under restrictedh omomorphisms, so $h(L(G')) \in \mathscr{L}_1$ therefore $L! \in \mathscr{L}_1$. Thus, the proof is complete.

NOTE: I wish to express my gratitude to dr. Gheorghe Păun for his suggestions and remarks.

REFERENCES

1. Gh. Păun, *Mecanisme generative ale proceselor economice* (Generative Mechanisms for Economic Processes), Bucureşti, Editura Tehnică, 1980.
2. A. Saloman, *Formal Languages*, New York, London, Academic Press, 1973.

Regional Computing Centre
Tulcea