String Operations Suggested by DNA Biochemistry: The Balanced Cut Operation[1]

Lila KARI, Andrei PĂUN
Department of Computer Science
University of Western Ontario
London, Ontario, Canada N6A 5B7

Gheorghe PĂUN
Institute of Mathematics of the Romanian Academy
PO Box 1 – 764, 70700 Bucureşti, Romania

Abstract. We introduce and investigate an operation with strings suggested by DNA processing (by means of exonucleases): the operation of cutting strings of equal length from the beginning and the end of a string. A related operation is that of cutting a square of a string from the prefix of a string. The closure properties of families in the Chomsky hierarchy are investigated (and, with one exception, for the case of linear languages, settled).

Keywords: DNA computing, language operations, closure properties.

1 Introduction

This paper deals with DNA computing *in info*: inspired from what happens *in vivo* and what can be accomplished *in vitro*, we define certain operations with strings and languages and we study them as formal operations. More precisely, we consider the action of certain exonucleases on DNA molecules, resulting in cutting off nucleotides from the two ends of a (linear) DNA molecule. An example is given in Figure 1. In a known elementary time unit, *Bal*31 cuts one pair of complementary nucleotides from the beginning of a DNA molecule and a pair from the end. Thus, in a given interval of time, (approximately) the same number of nucleotide pairs is removed from each end of the molecule.

[1]Research supported by the Natural Sciences and Engineering Research Council of Canada, Grant R2824A01.

This is a rather interesting operation with strings. We extend it to languages in the natural way and investigate the closure properties of families of languages under this new operation.

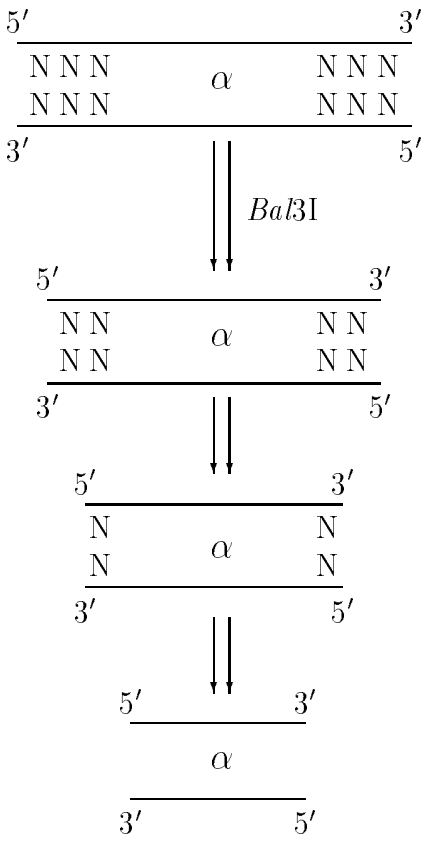

Figure 1: Balanced cut by an exonuclease

2 Language Theory Prerequisites

We mainly introduce here the notations which we shall use in the sequel; for further details of formal language theory we refer to [6].

For an alphabet V we denote by V^* the free monoid generated by V under the operation of concatenation; the empty string is denoted by λ and $V^* - \{\lambda\}$ is denoted by V^+. The length of $x \in V^*$ is denoted by $|x|$. If $x = x_1x_2x_3$, then we say that x_1 is a prefix, x_2 is a substring, and x_3 is a suffix of x. The sets of substrings, prefixes, and suffixes of a string $x \in V^*$ are denoted by $Sub(x), Pref(x), Suf(x)$, respectively.

The *circular permutation* of a string $x \in V^*$ is defined by $cp(x) = \{vu \mid x = uv,$ for $u, v \in V^*\}$.

The *left quotient* of a language $L_1 \subseteq V^*$ by a language $L_2 \subseteq V^*$ is $L_2 \backslash L_1 = \{x \in V^* \mid wx \in L_1 \text{ for some } w \in L_2\}$; the *right quotient* of L_1 by L_2 is $L_1 / L_2 = \{x \in V^* \mid xw \in L_1 \text{ for some } w \in L_2\}$. The *left derivative* of a language $L \subseteq V^*$ with respect to a string $x \in V^*$ is defined by $\partial_x^l(L) = \{w \in V^* \mid xw \in L\}$; the *right derivative* of L with respect to x is defined by $\partial_x^r(L) = \{w \in V^* \mid wx \in L\}$.

A finite automaton is given in the form $A = (K, V, s_0, F, P)$, where K is the set of states, V is the alphabet, s_0 is the initial state, F is the set of final states, and P is the set of transitions, presented as rewriting rules of the form $sa \to s'$ (in the state s, the automaton reads the symbol a and changes its state to s').

A *gsm* (= generalized sequential machine) is a finite automaton with output: $g = (K, V_1, V_2, s_0, F, P)$, where K is the set of states, V_1, V_2 are the input and the output alphabets, s_0 is the initial state, F is the set of final states, and P is the set of transitions of the form $sa \to xs'$, for $s, s' \in K, a \in V_1, x \in V_2^*$ (in state s, the machine reads the symbol a, changes its state to s' and produces the output string x). If in all rules $sa \to xs'$ we have $x \neq \lambda$, then g is said to be λ-*free*.

A Chomsky grammar is denoted by $G = (N, T, S, P)$, where N is the nonterminal alphabet, T is the terminal alphabet, $S \in N$ is the axiom, and P is the finite set of rewriting rules, given in the form $x \to y$, with $x, y \in (N \cup T)^*$ and x containing at least a nonterminal.

Finally, by *REG, LIN, CF, CS, RE* we denote the families of regular, linear, context-free, context-sensitive, recursively enumerable languages, respectively. It is worth noting that all these families are closed under union, intersection with regular languages, restricted morphisms, left and right derivatives, and inverse morphisms; *REG, CF, CS, RE* are also closed under concatenation, Kleene closure, and circular permutation, but LIN is not closed under these three operations. All families above but CS are closed under arbitrary gsm mappings and under left and right quotients by regular languages; CS is closed under λ-free gsm mappings only.

3 The Balanced Cut Operation

The basic operation we deal with in this paper, a model of the exonuclease action as shown in Figure 1, is defined as follows: for $x \in V^*$, we consider the set of strings

$$bc(x) = \{x_2 \mid x = x_1 x_2 x_3, \text{ for } x_1, x_2, x_3 \in V^* \text{ with } |x_1| = |x_3|\}.$$

We extend this operation – called *balanced cut* – to languages in the natural way: for $L \subseteq V^*$,

$$bc(L) = \bigcup_{x \in L} bc(x).$$

This operation is related to the *double prefix cut* operation: for $x \in V^*$, we define

$$dpc(x) = \{x_2 \mid x = x_1 x_1 x_2, \text{ for some } x_1, x_2 \in V^*\}.$$

The relation between the two operations is specified in the following lemma:

Lemma 1. *If F is a family of languages closed under double prefix cut, circular permutation, and λ-free gsm mappings, then F is also closed under balanced cut.*

Proof. Let us first note that the closure under λ-free gsm mappings also ensures the closure under intersection with regular languages.

Consider a language $L \subseteq V^*$ and two new symbols, a, b. Consider the gsm g which maps any string $x \in V^*$, nondeterministically, into a string of the form $a^i b y a^j b$, for some $i, j \geq 0$ such that all transitions of g are of the form $s\alpha \to \beta s'$, where s, s' are states and $\alpha \in V, \beta \in V \cup \{a, b\}$ (that is, g is λ-free and if $a^i b y a^j b \in g(x)$, then $|x| = |y| + i + j + 2$).

We obtain the equality:

$$bc(L) = (dpc(cp(g(L)) \cap a^* b a^* b V^*)) \cap V^*.$$

Indeed, g transforms a prefix x_1 and a suffix x_3 of a string $x_1 x_2 x_3 \in L$ into $a^i b, a^j b$, respectively, with $|x_1| = i+1, |x_3| = j+1, i, j \geq 0$; by a circular permutation followed by the intersection with the regular language $a^* b a^* b V^*$ we obtain strings of the form $a^i b a^j b x_2$; because no prefix zz of such a string can strictly contain the string $a^i b a^j b$, the double cut operation followed by the intersection with V^* means cutting the prefix $a^i b a^j b$; the only possibility is to have $i = j$, that is $|x_1| = |x_3|$, which is equivalent to $x_2 \in bc(x)$. □

We now investigate the closure properties of families in the Chomsky hierarchy under the operations *bc* and *dpc*.

The family of regular languages is closed under both these operations, as a consequence of the following result (a proof of it can be found in [7]):

Lemma 2. *The family of regular languages is closed under left and right quotients with arbitrary languages.*

Because we have

$$dpc(L) = \{xx \mid x \in V^*\} \backslash L,$$

we obtain the closure of REG under double prefix cut; Lemma 1 ensures that we also have the closure of REG under balanced cut.

The proof of Lemma 2 is not constructive, hence it makes sense to give a direct, effective proof of the closure of REG under our operations.

Lemma 3. *REG is effectively closed under the operation dpc.*

Proof. Let $A = (K, V, s_0, F, P)$ be a finite automaton. For $s \in K$, let $\bar{s}$ be a new state and let $\bar{K} = \{\bar{s} \mid s \in K\}$.

We construct the gsm

$$g = (K', V, V, s_0, \bar{F}, P'),$$

where

$$K' = K \cup \bar{K} \cup (K \times K \times K),$$

and P' contains the following transitions:

1. $sa \to s'$, for each $sa \to s' \in P$,

2. $sa \to (s_1, s_1, s_0)$, for each $sa \to s_1 \in P$,

3. $(s_1, s_2, s_3)a \to (s_1, s_2', s_3')$, for all $s_1 \in K, s_2a \to s_2' \in P, s_3a \to s_3' \in P$,

4. $(s_1, s_2, s_3)a \to \bar{s}_4$, for $s_2a \to s_4 \in P, s_3a \to s_1 \in P$,

5. $\bar{s}_1 a \to a\bar{s}_2$, for each $s_1a \to s_2 \in P$.

We have the equality $g(L(A)) = dpc(L(A))$'. Indeed, transitions of type 1 (followed by a transition of type 2) remove a prefix x of the scanned string such that $s_0x \Longrightarrow^* s_1$ in the automaton A, for some $s_1 \in K$; one introduces the state (s_1, s_1, s_0) and one continues by using transitions of type 3 (followed by one transition of type 4); the state s_1 is memorized and one scans a string z such that $s_1z \Longrightarrow^* s_4$ and $s_0z \Longrightarrow^* s_1$; therefore, also $s_0zz \Longrightarrow^* s_4$ is a correct sequence of transitions with respect to the automaton A; the use of transitions of type 5 follows a path in A which scans a string w. To summarize, $zzw \in L(A)$ and the output of g under input zzw is w. Therefore, $w \in dpc(L(A))$. □

Of course, also the family RE is closed under the operations *dpc, bc*. In contrast, no other family in the Chomsky hierarchy is closed under these operations – with the note that it is an *open problem* whether or not LIN is closed under the balanced cut operation.

Lemma 4. *The family CF is not closed under the bc operation.*

Proof. Let us consider the context-free language

$$L = \{a^n b^n c b^m a^m \mid n, m \geq 1\}.$$

Obviously, we have

$$bc(L) \cap a^+b^+cb^+a^+ = \{a^{n-k}b^ncb^ma^{m-k} \mid n, m \geq 1, k \geq 0, k < n, k < m\}.$$

This is not a context-free language.

Indeed, suppose that $L = L(G)$ for a context-free grammar $G = (N, \{a, b, c\}, S, P)$. All strings of the form $a^{n-k}b^ncb^ma^{m-k}$, for all possible n, m, k, are in L. That is, substrings $b^n, b^m, a^{n-k}, a^{m-k}$ can be arbitrarily large, and also the difference $|b^n| - |a^{n-k}| = k = |b^m| - |a^{m-k}|$ can be arbitrarily large. In order to generate such strings, we need derivations in G of the form

$$S \Longrightarrow^* u_1Xu_2Yu_3 \Longrightarrow^* u_1a^iXb^ju_2b^rYa^su_3 \Longrightarrow^* u_1x_1u_2x_2u_3 = a^{n-k}b^ncb^ma^{m-k},$$

with $u_1, u_2, u_3 \in \{a, b, c\}^*$, $1 \leq i < j$ and $1 \leq s < r$: by recurrent derivations of the form $Z \Longrightarrow^* a^gZb^h$ with $g \geq h$ intercalated with non-recurrent derivations (whose number is bounded, because N is a finite set), we can produce only prefixes $a^{n-k}b^n$ of strings in L with a bounded value for k; similarly for suffixes b^ma^{m-k}.

Thus, any derivation of the form

$$\begin{aligned} S &\Longrightarrow^* u_1Xu_2Yu_3 \Longrightarrow^* \\ &\Longrightarrow^* u_1a^{ih}Xb^{jh}u_2Yu_3 \Longrightarrow^* u_1a^{ih}x_1b^{jh}u_2x_2u_3 \\ &= \quad a^{n-k+ih}b^{n+jh}cb^ma^{m-k} \end{aligned}$$

is possible for each $h \geq 1$ (we have iterated the first subderivation and not the second one). Because $i < j$, we have $(n + jh) - (n - k + ih) = (j - i)h + k \neq k$, that is the obtained string is not in the language L. This contradicts the equality $L = L(G)$, hence L cannot be context-free. □.

Corollary 1. *The family CF is not closed under the operation dpc.*

An upper bound for the family of languages of the form $bc(L)$, for $L \in CF$, is provided by the family of matrix context-free languages.

A matrix grammar is a construct $G = (N, T, S, M)$, where N, T, S are as in a context-free grammar and M is a finite set of matrices, that is sequences $(A_1 \to x_1, \ldots, A_n \to x_n)$ of context-free rules. Using such a matrix means to apply the rules $A_1 \to x_1, \ldots, A_n \to x_n$ one by one, in this order. The family of languages generated by such grammars is denoted by MAT^λ; when only λ-free rules are allowed, the superscript is removed. It is known that $CF \subset MAT \subset CS, CF \subset MAT^\lambda \subset RE, CS - MAT^\lambda \neq \emptyset$ (see [2] for details), and that each one-letter language in MAT^λ is regular (see [4]).

Lemma 5. *If L is a context-free language, then $bc(L) \in MAT^\lambda$.*

Proof. Let $L \subseteq V^*$ be a context-free language. Consider the gsm g which transforms strings $x_1x_2x_3 \in V^*$ into strings of the form $c^ix_2d^j$, with $i = |x_1|, j = |x_3|$; c, d are new symbols. The language $g(L)$ is context-free. Let $G = (N, V \cup \{c, d\}, S, P)$ be a context-free grammar for $g(L)$. Denote by h the morphism which leaves all symbols in $N \cup V$ unchanged and maps the symbols c, d into C, D, respectively.

We construct the matrix grammar $G' = (N \cup \{C, D\}, V, S, M)$, where

$$\begin{aligned} M = &\{(X \to h(x)) \mid X \to x \in P\} \\ &\cup \{(C \to \lambda, D \to \lambda)\}. \end{aligned}$$

The use of a matrix of the form $(C \to \lambda, D \to \lambda)$ erases one occurrence of C and one occurrence of D. Therefore, a string C^iyD^j with $y \in V^*$ (hence a from $h(g(L))$) is transformed into y only when $i = j$. Consequently, $L(G') = bc(L)$. □

By an easy modification of the proof above, we get:

Corollary 2. *The family MAT^λ is closed under the operation bc.*

The above statement is not true if the operation *bc* is replaced by *dpc*. In fact, a much stronger result is true, also proving the non-closure of LIN under the operation *dpc*:

Lemma 6. *There are linear languages L such that $dpc(L) \notin MAT^{\lambda}$.*

Proof. Let us consider the following language:

$$L = \{a^{i_1}ba^{i_2}b \dots ba^{i_{2k-1}}ca^{2i_{2k-1}}b \dots ba^{2i_3}ba^{2i_2}c^2a^{2i_1} \mid k \geq 1, i_j \geq 1, \text{ for all } 1 \leq j \leq 2k-1\}.$$

Clearly, this is a linear language.

Consider also the gsm g which works as follows when scanning a string in L:

- we scan the prefix $wc, w \in \{a,b\}^*$, and we leave it unchanged,
- when scanning the substring $czc^2, z \in \{a,b\}^*$, we replace one occurrence of b by bab (that is, a substring ab is inserted in an arbitrary place in z); all other symbols are left unchanged;
- we leave the suffix a^{2i_1} unchanged.

The language $g(L)$ is linear.

Let us note that the strings in L have two "halves", separated by the central occurrence of c; the blocks of symbols a in the right half are of double length as compared to the corresponding blocks in the left half; the substring c^2 separates the last blocks of a occurrences in the right half. When generating $g(L)$, one more block of a occurrences is introduced, consisting of one symbol only. In this way, the substring delimited by the occurrences of c have the same number of blocks of symbols a as the string placed at the left of the central occurrence of c.

We have the equality

$$dpc(g(L)) \cap ca^+ = \{ca^{2^{2n-1}} \mid n \geq 1\}. \qquad (*)$$

Let us examine the way of producing a string in ca^+ by a double prefix cut operation, starting from a string in $g(L)$.

The strings in $g(L)$ are of the form

$$w = a^{i_1}ba^{i_2}b \dots ba^{i_{2k-1}}ca^{2i_{2k-1}}b \dots ba^{2i_r}baba^{2i_{r-1}} \dots ba^{2i_2}c^2a^{2i_1},$$

for some $k \geq 1, i_j \geq 1, 1 \leq j \leq 2k-1$, and $3 \leq r \leq 2k-1$. In order to get the string ca^{2i_1}, we have to cut a prefix $xcxc$ of w, that is

$$x = a^{i_1}ba^{i_2}b \dots ba^{i_{2k-1}} = a^{2i_{2k-1}}b \dots ba^{2i_r}baba^{2i_{r-1}} \dots ba^{2i_2}.$$

With a string $w \in g(L)$ with this property, we associate an undirected graph $\Gamma(w)$ as follows:

- associate the nodes $\alpha_1, \dots, \alpha_{2k-1}$ with the blocks $a^{i_1}, \dots, a^{i_{2k-1}}$ and the nodes $\beta_1, \dots, \beta_{2k}$ with the blocks $a^{2i_{2k-1}}, \dots, a^{2i_r}, a, a^{2i_{r-1}}, \dots, a^{2i_2}, a^{2i_1}$;

– draw an arc (α_i, β_i) for each $i = 1, 2, \ldots, 2k-1$; call these arcs *lower* arcs; they express the equality of the substrings a^s of w as imposed by the fact that $w = xcxcca^{2i_1}$;

– draw an arc (α_s, β_t) for each pair $(s,t), 1 \leq s \leq 2k-1, 1 \leq t \leq 2k$, such that $j_t = 2i_s$ and a^{j_t}, a^{i_s} are blocks which correspond to each other in the definition of L; call these arcs *upper* arcs; they express the relation between substrings a^i of w placed to the left and to the right of the "central" occurrence of c, as imposed by the definition of L.

Figure 2 presents the graph for the case of $k = 3$; the upper and the lower arcs are drawn in the corresponding positions.

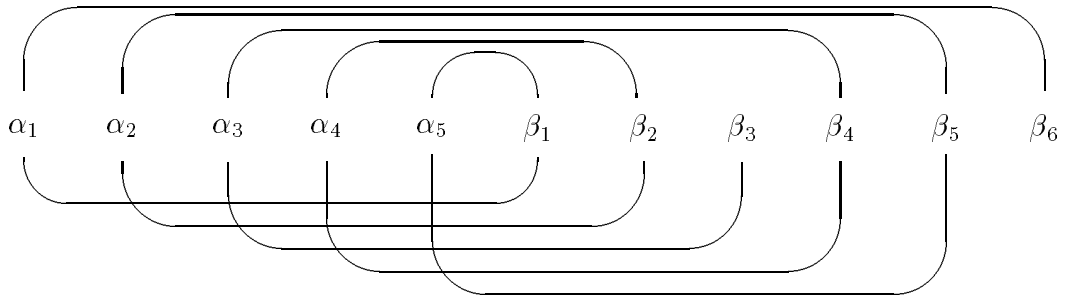

Figure 2: The graph $\Gamma(w)$ for $k = 3$

Let us denote by $val(\alpha_i), val(\beta_i)$ the length of the substring a^s of w asssociated as mentioned above with α_i, β_i, respectively.

Several facts about the graph $\Gamma(w)$ are useful for the subsequent reasoning:

– The node β_{2k} has the degree 1 (one upper arc and no lower arc reaches it).

– The node β_p having $va;(\beta_p) = 1$ (that is, corresponding to the substring inserted by the gsm g in the strings of L) cannot be reached by an upper arc: otherwise, $val(\beta_p)$ has to be an even number, the double of $val(\alpha_t)$ for some t, which is not the case. Because all nodes $\beta_1, \ldots, \beta_{2k-1}$ are reached by a lower arc, it follows that β_k is reached by a lower arc only, it has the degree 1.

– If α_i, β_j are linked by an upper arc, then $val(\beta_j) = 2 \cdot val(\alpha_i)$; if they are linked by a lower arc, then $val(\alpha_i) = val(\beta_j)$.

– All nodes different from β_{2k} and β_p mentioned above have the degree 2: all nodes $\alpha_i, 1 \leq i \leq 2k-1$, are reached both by upper and lower arcs; all nodes different from β_p are reached by an upper arc, all nodes different from β_{2p} are reached by a lower arc.

– There is no cycle in $\Gamma(w)$. Indeed, this is a bi-partite graph, always nodes α_i are linked by nodes β_j. Assume that there is a cycle. It must contain the same number of nodes of type α_i as nodes of type β_j. Let $\alpha_{k_1}, \ldots, \alpha_{k_s}, \beta_{l_1}, \ldots, \beta_{l_s}$ be these nodes. Because of the links by lower arcs, we must have the equality $\{val(\alpha_{k_1}), \ldots, val(\alpha_{k_s})\} = \{val(\beta_{l_1}), \ldots, val(\beta_{l_s})\}$. Let q be the maximum of $val(\alpha_{k_t}), 1 \leq t \leq s$. Because of the upper arcs, $2q$ should be an element of the set $\{val(\beta_{l_t}) \mid 1 \leq t \leq s\}$. However, $2q \notin \{val(\alpha_{k_t}) \mid 1 \leq t \leq s\}$, therefore $\{val(\alpha_{k_1}), \ldots, val(\alpha_{k_s})\} \neq \{val(\beta_{l_1}), \ldots, val(\beta_{l_s})\}$, a contradiction.

Because $\Gamma(w)$ contains no cycle and all nodes have the degree one or two, it follows that it is a connected graph. According to Euler theorem (a connected graph with nodes of even degree with the exception of two nodes contains an Eulerian path, starting in one of the two nodes of odd degree and ending in the other node of odd degree), $\Gamma(w)$ contains a path starting in β_p, ending in β_{2k} and using all arcs (an Eulerian path). As we have seen above, $val(\beta_p) = 1, val(\beta_{2k}) = 2i_1$. On this path, all the $2k-1$ upper arcs are used. They relate nodes α_i, β_j such that $val(\beta_j) = 2 \cdot val(\alpha_i)$. Consequently, the values are doubled $2k-1$ times. We start from $val(\beta_p) = 1$, hence $val(\beta_{2k}) = 2^{2k-1} \cdot val(\beta_p) = 2^{2k-1}$.

This concludes the proof of the equality $(*)$.

The language $\{ca^{2^{2n-1}} \mid n \geq 1\}$ is not in the family MAT^λ (the family MAT^λ is closed under arbitrary morphisms and $\{a^{2^{2n-1}} \mid n \geq 1\}$ is a one-letter non-regular language). The family MAT^λ is also closed under intersection with regular languages. Consequently, $dpc(g(L)) \notin MAT^\lambda$. □

Corollary 3. *The families LIN and MAT^λ are not closed under the operation dpc.*

As we have mentioned above, the closure of LIN under the balanced cut operation remains to be clarified. Note that, because LIN is not closed under circular permutation, the non-closure of LIN under the operation bc does not imply – via Lemma 1 – the non-closure under the operation dpc.

The case of the family CS is easy to be settled. The following more general result is true:

Lemma 7. *If a family F is closed under concatenation with regular sets, right derivative, and the operation bc, then it is closed under the operation Suf, of taking suffixes.*

Proof. For $L \subseteq V^*$ and $c, d \notin V$, we can write

$$Suf(L) = \partial_d^r(bc(Ldc^*)).$$

The equality can be easily checked: by a balanced cut operation, any prefix x of a string $w \in L$ can be cut, as a prefix of wdc^i; only when $i = |x|$ the derivative by d is defined. Thus, all and exactly the suffixes of strings in L can be obtained. □

Corollary 3. *The family CS is not closed under the operations dpc and bc.*

For the sake of readability, we collect the results in the previous lemmas in a theorem:

Theorem 1. *The closure properties in Table 1 hold.*

Table 1: Closure properties under operations *bc* and *dpc*

	REG	LIN	CF	CS	MAT^{λ}	RE
bc	YES	?	NO	NO	YES	YES
dpc	YES	NO	NO	NO	NO	YES

Note the interesting case of the family MAT^{λ}, which is closed under the operation *bc* but not under the operation *dpc*.

4 Related Operations

Several operations related to the previous ones can be imagined.

For instance, instead of cutting a prefix and a suffix of the same length, we can cut a prefix and a suffix which are one the mirror image of the other (*mirror balanced cut*, *mbc*), or even identical strings (*double balanced cut*, *dbc*). All the closure properties proved in the previous section, with the exception of those referring to the family LIN, remain true for these new operations with similar proofs. (For instance, in the proof of Lemma 7 we can take V^* instead of c^* and we obtain the same result for each of *mbc* and *dbc*.) For the family of linear languages we need new proofs.

Lemma 8. *The family LIN is not closed under the operation mbc.*

Proof. Consider the linear language

$$L = \{a^{i_1}ba^{i_2}b\dots ba^{i_k}ba^{2i_k}b\dots ba^{2i_2}ba^{2i_1}ba \mid k \geq 1, i_j \geq 1, 1 \leq j \leq k\}.$$

We obtain

$$mbc(L) \cap ba^+b = \{ba^{2^n}b \mid n \geq 1\}.$$

Indeed, a string in ba^+b is obtained by cutting from a string

$$w = a^{i_1}ba^{i_2}b\dots ba^{i_k}ba^{2i_k}b\dots ba^{2i_2}ba^{2i_1}ba$$

a prefix and a suffix which are one the mirror image of the other only when $a^{i_1}ba^{i_2}b\dots ba^{i_k} = mi(a^{2i_k-1}b\dots ba^{2i_2}ba^{2i_1}ba)$. This implies that $i_1 = 1, i_j = 2i_{j-1}, 2 \leq j \leq k$, which means that $i_k = 2^{k-1}$. Because $mbc(w) \cap ba^+b = ba^{2i_k}$, we get $mbc(w) \cap ba^+b = ba^{2^k}$. □

Lemma 9. *The family LIN is not closed under the operation dbc.*

Proof. Consider the linear language

$$L = \{a^{i_1}ba^{i_2}b\dots ba^{i_{2k-1}}ba^{2i_{2k-1}}b\dots ba^{2i_3}ba^{2i_2}b^2a^{2i_1} \mid k \geq 1, i_j \geq 1, \text{ for all } 1 \leq j \leq 2k-1\}.$$

We proceed as in the proof of Lemma 6. Let again be g the gsm which inserts a new substring a^ib in the "right half" of strings in L, namely with $i = 1$.

We obtain

$$dbc(g(L)) \cap ba^+b = \{ba^{2^{2n-1}}b \mid n \geq 1\}.$$

This can be seen as in the proof of Lemma 6. For instance, the graph describing the links between the blocks a^i of the strings in $g(L)$ which lead by a double balanced cut to a string in ba^+b looks like that in Figure 3, where we have considered the case $k = 3$. Ths substring to be obtained after the double balanced cut and the intersection with ba^+b is the central one, corresponding to β_1 in the graph. □

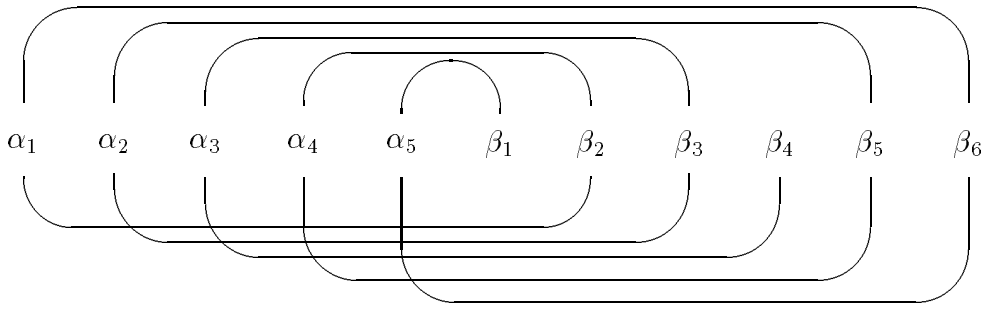

Figure 3: The graph for the proof of Lemma 9

Another possibility is not to cut but to grow the strings at the two ends. Thus, we define the *balanced growth* of $x \in V^*$ by

$$bg(x) = \{x_1xx_2 \mid x_1, x_2 \in V^*, |x_1| = |x_2|\}.$$

It is very easy to prove that REG is not closed under this operation, but all other families considered above are closed. For instance, if $G = (N, T, S, P)$ is a context-free grammar, the grammar with the rules $P \cup \{S' \to aS'b \mid a, b \in T\} \cup \{S' \to S\}$, with S' as the new axiom, generates the language $bg(L(G))$. This also shows that $bg(L) \in LIN$ for each regular language L.

Finally, instead of deleting or adding strings at the ends of a string, we can only mark a prefix and a suffix of the same length. More formally, we consider the operation of *balanced marking*, defined by

$$bm(x) = \{x_1cx_2cx_3 \mid x = x_1x_2x_3, |x_1| = |x_3|\}.$$

(For $x \in V^*$, c is a symbol not in V.)

Again, it is clear that REG is not closed under this operation, but CS and RE are closed. Neither LIN and CF are closed: for the linear language

$$L = \{a^nb^{2n} \mid n \geq 1\}$$

we have

$$bm(L) \cap a^+cb^+cb^+ = \{a^ncb^ncb^n \mid n \geq 1\},$$

which is a non-context-free language.

However, $bm(L) \in LIN$ for $L \in REG$: consider a finite automaton $A = (K, V, s_0, F, P)$ and construct the linear grammar $G = (N, T, S, P')$, where

$$K = \{S\} \cup \{(s, s'), [s, s'] \mid s, s \in K\},$$

and P' contains the following rules:

1. $S \to [s_0, s_f]$, for all $s_f \in F$,
2. $[s_1, s_2] \to a[s_3, s_4]b$, for all $s_1a \to s_3 \in P, s_4b \to s_2 \in P$,
3. $[s_1, s_2] \to c(s_1, s_2)c$, for all $s_1, s_2 \in K$,
4. $S \to c9s_0, s_f)c$, for all $s_f \in F$,
5. $(s_1, s_2) \to a(s_3, s_2)$, for all $s_1a \to s_3 \in P$ and $s_2 \in K$,
6. $(s_1, s_2) \to a$, for $s_1a \to s_2 \in P$,
7. $[s_1, s_2] \to cc$, for $s_1 = s_2$,
8. $S \to cc$, if $\lambda \in L(A)$.

The equality $L(G) = bm(L(A))$ is obvious.

We conclude this paper by emphasizing the fact that the biochemistry of DNA suggests many new and interesting problems from the formal language theory point of view. In particular, many new operations with strings and languages can be found in this area. Such operations have already been studied for example in [5], [1], but the investigation is by no means complete, [3].

References

[1] J. Dassow, V. Mitrana, On some operations suggested by genome evolution, *Proc. Second Pacific Symposium on Biocomputing* (R. B. Altman, A. K. Dunker, L. Hunter, T. Klein, eds.), World Scientific, Singapore, 1997, 97 – 108.

[2] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin, Heidelberg, 1989.

[3] J. Dassow, Gh. Păun, Remarks on operations suggested by mutations in genomes, *Fundamenta Informaticae*, 2-3 (1998), 183-200.

[4] D. Hauschild, M. Jantzen, Petri nets algorithms in the theory of matrix grammars, *Acta Informatica*, 31 (1994), 719 – 728.

[5] Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, Springer-Verlag, Berlin, 1998.

[6] G. Rozenberg, A. Salomaa, *Handbook of Formal Languages*, Springer-Verlag, Berlin, 1997, vols. 1-3.

[7] S. Yu, Regular languages, chapter vol.1 of [6], 41-111.