International Journal of Foundations of Computer Science

STATE COMPLEXITY OF THE SUBWORD CLOSURE OPERATION WITH APPLICATIONS TO DNA CODING*

CEZAR CÂMPEANU
Department of Computer Science and Information Technology, The University of Prince Edward Island,
550 University Avenue, Charlottetown, PE, C1A 4P3 Canada
cezar@sun11.math.upei.ca

and

STAVROS KONSTANTINIDIS
Department of Mathematics and Computing Science, Saint Mary's University,
Halifax, Nova Scotia, B3H 3C3 Canada
s.konstantinidis@smu.ca

Received (received date)
Revised (revised date)
Communicated by Editor's name

ABSTRACT

We are interested in the state complexity of languages that are defined via the subword closure operation. The subword closure of a set S of fixed-length words is the set of all words w for which any subword of w of the fixed length is in S. This type of constraint appears to be useful in various situations related to data encodings and in particular to DNA encodings. We present a few results related to this concept. In particular we give a general upper bound on the state complexity of a subword closed language and show that this bound is tight infinitely often. We also discuss the state complexity of DNA computing related cases of the subword closure operation.

Keywords: subword closure, state complexity, regular language, automaton, balanced words, DNA languages, gc-ratio.

1. Introduction

A language L is any set of words over some alphabet Σ. Two examples of alphabets are the binary alphabet $\{0,1\}$ and the DNA alphabet $\{a,c,g,t\}$. A *subword* of L is any word that occurs in some word of L; that is, u is a subword of L if there is a word of the form xuy in L. The expression $\mathrm{Sub}_k(L)$ denotes the set of all subwords of L of length k. We are interested in languages L whose subwords of length k, for some fixed parameter k, satisfy some desirable constraint. This topic

*Research supported by NSERC.

is motivated by various questions related to DNA encodings and combinatorial channels.

Definition 1 *A* subword constraint *is any nonempty set S of words of length k, for some fixed $k \geq 1$. We say that a language L* satisfies the subword constraint S *if every subword of L of length k belongs to S; that is,* $\mathrm{Sub}_k(L) \subseteq S$.

The symbols $\Sigma^*, \Sigma^m, \Sigma^{\geq m}$ denote, respectively, the sets of all words; all words of length m; and all words of length at least m. The *empty word* is denoted by λ. If $L \subseteq \Sigma^*$, then $L^m = \{w \in \Sigma^* \mid w = w_1 \dots w_m, w_i \in L, 1 \leq i \leq m\}$.

In [1, 10], a particular type of subword constraint was used to model bond-free DNA languages, and the study of this constraint led to the technical concept of subword closure operation $\otimes$:

$$S^{\otimes} = \{w \in \Sigma^* : \mathrm{Sub}_k(w) \subseteq S\}.$$

Thus, $S^{\otimes}$ is the language of all words w such that every subword of w of length k belongs to S. In [10], it is required that all words in $S^{\otimes}$ must be of length at least k. Here, however, we drop this requirement. This implies that $S^{\otimes}$ contains all words of length less than k.

Remark 1 *By the definition of the subword closure operation, it follows immediately that a language L satisfies the subword constraint S, that is* $\mathrm{Sub}_k(L) \subseteq S$, *if and only if $L \subseteq S^{\otimes}$.*

Since $\lambda \in S^{\otimes}$, we have that $S^{\otimes} \subseteq (S^{\otimes})^n$, for all $n \geq 1$,

In this work we are interested in the state complexity of languages $S^{\otimes}$. The *state complexity* $\mathrm{sc}(R)$ of a regular language R is the number of states in the minimal complete deterministic automaton accepting R [15]. We recall from [10] that, for any subword constraint S, the language $S^{\otimes}$ is regular.

We list now a few examples to demonstrate that the concepts of subword constraint and subword closure operation are interesting objects of study.

In [6], a language L is called a θ-k-code, where θ is an antimorphic involution, if $\theta(x) \neq y$ for any subwords x, y in $\mathrm{Sub}_k(L)$. An antimorphic involution is a mapping $\theta : \Sigma^* \to \Sigma^*$ such that $\theta^2(x) = \theta(x)$ and $\theta(xy) = \theta(y)\theta(x)$ for all words x, y. The relationship $\theta(x) = y$ indicates that the molecules corresponding to x and y can form chemical bonds between them – see [10] for further details. Obviously, if a language L is a θ-k-code, then $L \subseteq S^{\otimes}$, for some subword constraint S with $\theta(u) \neq v$ for all u, v in S.

The gc-ratio of the strands involved in DNA computing is an important parameter for the success of the melting operation [8]. In particular, it is recommended that each of the DNA blocks that make up DNA strands to contain a ratio of g and c's which is about 50% of the total length of the block. Here, let S be the set of all DNA words of length k such that the ratio of g and c's over k is about 50%. Then a language L has a gc-ratio of about 50% for subwords of length k, if it satisfies the subword constraint S; thus, $L \subseteq S^{\otimes}$.

The continuity constraint is also important in reliable DNA computing and expresses the requirement that, no k consecutive identical bases (alphabet symbols)

appear in the DNA molecules (words) involved [14]. Let C_k be the set of all words of length k containing at least two different symbols. A language L satisfies the continuity constraint if $L \subseteq C_k^{\otimes}$.

Another example where the subword constraint and operation are relevant is when describing a combinatorial channel [9] via a set of edit strings. Briefly, a channel is a set of pairs (w, z) of words, where w is the input (or transmitted) word and z is the output (or received) word. An edit string is a word over the alphabet of basic edit operations $E = \{(x/y), (\lambda/y), (x/\lambda) : x, y \in \Sigma\}$, where λ is the empty word over Σ. Any basic edit operation other than (x/x) is called an *error*. For example, the edit string $(a/t)(c/c)(g/g)(t/\lambda)(a/a)(c/c)$ says that the word *acgtac* can be transformed to *tcgac* using one substitution, (a/t), and one deletion, (t/λ). Now we only consider the alphabet $E_0 = \{(x/y) : x, y \in \Sigma\}$. Let S be the following subword constraint: all edit strings over E_0 of length k containing at most m errors. Then, the channel that permits at most m substitution errors in any subword of length k of the input word is described by the subword closure of S, which is $S^{\otimes}$. In [9] it is shown that

$$1 + \binom{k-1}{m-1} \leq \mathrm{sc}(S^{\otimes}) \leq 1 + \sum_{i=0}^{m} \binom{k-1}{i}.$$

There are probably other situations where the subword constraint and closure concepts might be useful. In cryptography, for example, a long message w is encrypted using some key s. The key s should have high complexity so that when an attacker gets hold of any part of the encrypted message, that part should look too complex for the attacker to analyze. Complexity here could be defined in terms of random strings – see [11] for a relevant discussion. In this scenario, the subword constraint is some set S of complex words and the key will be chosen from the language $S^{\otimes}$.

The paper is organized as follows. The next section presents a general upper bound on the state complexity of the subword closure language $S^{\otimes}$. It is also shown that this upper bound is tight infinitely often. In Section 3, we focus on the case of d-balanced binary words, that is, binary words for which the absolute difference between the numbers of 0s and 1s in those words is at most d. This constraint is directly related to the *gc*-ratio constraint for DNA words. Section 4 considers the continuity constraint and discusses briefly the problem of combining constraints. Finally, Section 5 contains a few concluding remarks.

2. A General Bound on the Subword Closure Complexity

In this section we obtain a general upper bound on the state complexity of $S^{\otimes}$, for any subword constraint S of some length k. We also show that the bound is tight infinitely often and also far from tight infinitely often.

For a set A, the expression $|A|$ denotes the *cardinality* of the set A. For any word w, the expression $|w|$ denotes the *length* of the word w.

A *trie* is an automaton accepting a finite language F, and has the structure of a tree. The states are all the prefixes of F with the root being the start state

(corresponding to the empty word) and the words of F are the final states – see [3] for further details on tries.

Lemma 1 *Let T be a trie accepting n words of length k. The number of states in T is upper bounded by*

$$\frac{|\Sigma|^{\lceil \log_{|\Sigma|} n \rceil} - 1}{|\Sigma| - 1} + (k - \lceil \log_{|\Sigma|} n \rceil + 1)n.$$

Moreover, this upper bound is tight infinitely often.

Proof. Let n_i be the number of states in the trie at level $i = 0, \ldots, k$, and let r be the first level of the trie containing n states, that is, $r = \lceil \log_{|\Sigma|} n \rceil$. Obviously $n_i \leq |\Sigma|^i$ for all i. Hence,

$$\sum_{i=0}^{r-1} n_i \leq 1 + |\Sigma| + \cdots |\Sigma|^{r-1}.$$

Also, as $n_i \leq n_{i+1}$ and $n_k = n$, we have that

$$\sum_{i=r}^{k} n_i \leq (k - r + 1)n,$$

and the upper bound follows now easily from these observations. That the upper bound is tight infinitely often follows from the fact that, for each $k \geq 2$, the number of states in the trie accepting the words $\Sigma^{k-1}a$, where a is in Σ, is equal to

$$1 + \cdots + |\Sigma|^{k-1} + |\Sigma|^{k-1} = 1 + \cdots + |\Sigma|^{r-1} + (k - r + 1)n,$$

where $n = |\Sigma|^{k-1}$ and $r = k - 1 = \lceil \log_{|\Sigma|} n \rceil$. □

Theorem 1 *For any subword constraint S of some length k, there is a complete deterministic automaton accepting $S^{\otimes}$ with at most*

$$U(S) = \frac{|\Sigma|^{\lceil \log_{|\Sigma|} |S_1| \rceil} - 1}{|\Sigma| - 1} + (k - \lceil \log_{|\Sigma|} |S_1| \rceil) \cdot |S_1| \ + 1$$

states, where S_1 is the set of all prefixes of S of length $k - 1$. We denote this automaton by $[S]^{\otimes}$.

Proof. First, we construct the trie for S_1; hence, the states of the trie are exactly all the prefixes of S_1 (these include the words in S_1). The automaton $[S]^{\otimes}$ results when we add transitions between states of length $k - 1$ as follows:

- for any two states of the form ax, xb with $a, b \in \Sigma$ the transition (ax, b, xb) is added if the word axb is in S.

Note that state ax "remembers" the last $k - 1$ symbols read by the automaton. All states of the automaton $[S]^{\otimes}$ are final. It is easy to see that the constructed automaton accepts $S^{\otimes}$. We also note that this construction is similar to that of the de Bruijn diagram for words of length $|\Sigma|^k$ containing only distinct occurrences of

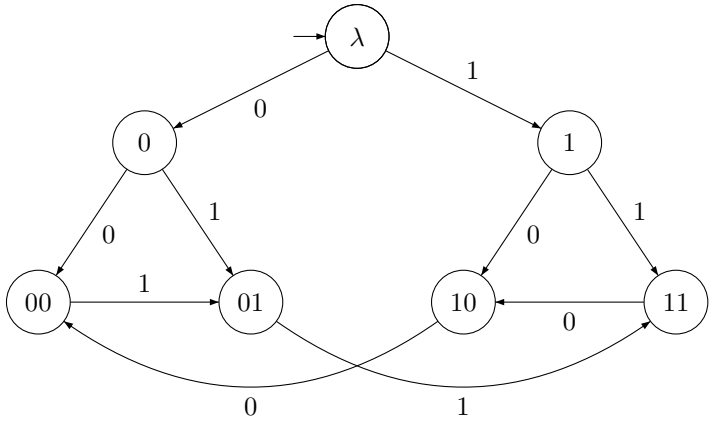

Figure 1: Automaton accepting all binary words for which every subword of length 3 is in $\{001, 011, 100, 110\}$. All states are final.

subwords of length k [12]. The bound on the number of states follows from Lemma 1 when we recall that the states of $[S]^{\otimes}$ are exactly those of the trie for S_1 plus one rejecting sink state. □

The upper bound $U(S)$ given in the above theorem is not tight, in general. In particular, there are examples where the bound is tight and others where the bound is far from tight. For example, let w be a word of some length k and consider the language $\Sigma^* - \Sigma^* w \Sigma^*$ consisting of all words that contain no subword equal to w. For simplicity, assume that $|\Sigma| = 2$. This language can be accepted by a deterministic automaton of $k+1$ states. Indeed, there is a complete deterministic automaton of $k+1$ states for $\Sigma^* w$ (called the dictionary-matching automaton of $\{w\}$ in [3]) having a single final state. This automaton can be modified easily to accept the language $\Sigma^* - \Sigma^* w \Sigma^*$ by adding no extra states. On the other hand, this language is equal to $S^{\otimes}$, where $S = \Sigma^k - \{w\}$. As the set of prefixes of S of length $k-1$ consists of $|\Sigma|^{k-1}$ words, the automaton $[S]^{\otimes}$ in Theorem 1 has $2^{k-1} - 1 + 2^k$ states, which is far from $k+1$.

Next we show that the upper bound $U(S)$ of Theorem 1 is tight infinitely often. A particular case of our construction is shown in Fig. 1, where $k = 3$ and the automaton in this figure accepts the language $S^{\otimes}$ over $\{0, 1\}$, where $S = \{001, 011, 100, 110\}$.

Definition 2 *Let $k \geq 3$ and let S be a set of words of length k. We say that S is* k-ps-full *if, for every word $w \in \Sigma^{k-1}$, the following two properties are satisfied:*

1. *there is exactly one $a \in \Sigma$ such that $aw \in S$; and*
2. *there is exactly one $b \in \Sigma$ such that $wb \in S$.*

Let S be a k-ps-full set. By the above definition, for every possible word w of length $k-1$, we can find exactly two words x, y in S such that w is a prefix of x

and is also a suffix of y. Moreover, it follows that the set of prefixes of S is equal to Σ^{k-1}.

Lemma 2 *Let S be a k-ps-full set and let $w \in S^{\otimes}$. For every $n \geq 0$ there is a word $z \in \Sigma^n$ such that $wz \in S^{\otimes}$.*

Proof. For $n = 0$ the statement is obvious. Assume now the statement holds for some $n \geq 0$. One shows that it holds for $n + 1$ as well, using the definition of a ps-full set. □

Lemma 3 *For every integer $i \geq 3$, there exists an i-ps-full set S of words of length i such that $|S| = |\Sigma|^{i-1}$.*

Proof. We prove the statement by induction on i. For $i = 3$ the set $S = \{001, 011, 100, 110\}$ is 3-ps-full – see also Fig. 1.

Now we make the construction for the general case. Suppose the alphabet Σ is equal to $\{a_1, \ldots, a_p\}$, and let $g : \Sigma \longrightarrow \Sigma$ be a bijective function with no fixed point, i.e., $g(x) \neq x$ for all $x \in \Sigma$. First we enumerate all words of length $i - 1$ in lexicographical order as: $w_1, w_2, \ldots, w_{p^{i-1}}$. Of course, the first p^{i-2} words have a_1 as the first letter, the next p^{i-2} words have a_2 as the first letter, and so on; the last p^{i-2} words have a_p as the first letter. The first letter of a word w will be denoted as $f(w)$. We define the set $S' = \{w \in \Sigma^i \mid w = w_j g(f(w_j)), j = 1, \ldots, p^{i-1}\}$, which is obviously of cardinality p^{i-1}.

We show that the set S' is an i-ps-full set. Let w be any word of length $i - 1$. First we show that $a_h w \in S'$ for some $a_h \in \Sigma$. Indeed, w can be written as $w_1 a$ for some $a \in \Sigma$. Let $a_h = g^{-1}(a)$. As $a_h w_1 \in \Sigma^{i-1}$, we have that $a_h w_1 a$ must be in S'. In fact, this a_h is unique, as for any $a_{h'} \in \Sigma$ with $a_{h'} w_1 a \in S'$ it must be that $a_{h'} = g^{-1}(a)$. This proves the first property of Definition 2. The second property can be shown analogously. □

Example 1 *For $i = 4$ and $\Sigma = \{0, 1, 2\}$, we give two examples of 4-ps-full sets:*

$S'_1 = \{$ 0001, 0011, 0021, 0101, 0111, 0121, 0201, 0211, 0221,
1002, 1012, 1022, 1102, 1112, 1122, 1202, 1212, 1222,
2000, 2010, 2020, 2100, 2110, 2120, 2200, 2210, 2220 $\}$,

using $g = \{(0, 1), (1, 2), (2, 0)\}$, *and*

$S'_2 = \{$ 0002, 0012, 0022, 0102, 0112, 0122, 0202, 0212, 0222,
1000, 1010, 1020, 1100, 1110, 1120, 1200, 1210, 1220,
2001, 2011, 2021, 2101, 2111, 2121, 2201, 2211, 2221 $\}$,

using $g = \{(0, 2), (1, 0), (2, 1)\}$.

Some Notation: We recall (see [15], for instance) that, for any given regular language R, the minimal automaton accepting R consists of states $[w]$ such that $[w]$ represents the *class* of all words w' that are equivalent to w, $w \equiv_R w'$, which means that $wz \in R$ if and only if $w'z \in R$, for all words z. Moreover, the state $[w]$ represents the set of all words formed in all the paths from the start state $[\lambda]$ to $[w]$. We also recall that, in this notation, when the automaton starts at some state $[w]$ with a word v as input, then it arrives at state $[wv]$ after consuming the input v.

Lemma 4 *Let S be a subword constraint of some length k. The following statements hold true.*

1. *For every word u, if $u \notin S^{\otimes}$, then $xuy \notin S^{\otimes}$ for all words x, y.*

2. *For every word u, if $u \notin S^{\otimes}$, then $[u] = [xuy]$ for all words x, y.*

Proof. Follows easily from the definitions. □

Lemma 5 *Let S be a subword constraint of some length k. For all words $w \in \Sigma^{k-1}$ and $uw \in S^{\otimes}$, we have that $[uw] = [w]$.*

Proof. The statement follows easily when we note that, for any word z, whether uwz is in $S^{\otimes}$ depends on whether wz is in $S^{\otimes}$. □

Theorem 2 *For every integer $k \geq 3$, there exists a subword constraint S of length k such that $\mathrm{sc}(S^{\otimes}) = U(S)$.*

Proof. Let S be a k-ps-full subword constraint of length $k \geq 3$ according to Lemma 3. The fact that $\mathrm{sc}(S^{\otimes}) \leq U(S)$ follows from Theorem 1. Now consider the set $S^{\otimes}$ and the equivalence relation $\equiv_{S^{\otimes}}$ associated with this language. We show that, for every two different words x_1, x_2 that are proper prefixes of S, we have that $x_1 \not\equiv_{S^{\otimes}} x_2$.

Indeed, we can assume that $|x_1| \leq |x_2|$. We distinguish two cases:

1. Case $|x_1| < |x_2|$. In this case, there exist $y_2 \in \Sigma^{k-1-|x_2|}$ and $a \in \Sigma$ such that $x_2 y_2 a \in S$. Since S is k-ps-full, $x_2 y_2 b \notin S$ for any $b \in \Sigma - \{a\}$; hence as $x_2 y_2 b$ is of length k it is not in $S^{\otimes}$. On the other hand as $|x_1| < |x_2|$, it follows that $|x_1 y_2 b| < k$, therefore, $x_1 y_2 b \in S^{\otimes}$.

2. Case $|x_1| = |x_2|$. In this case, $x_1 = y_1 a y$ and $x_2 = y_2 b y$, for some $a, b \in \Sigma$, with $a \neq b$, and $y \in \Sigma^*$. As x_1 is a proper prefix of S, there is a nonempty word s such that $x_1 s \in S$. By Lemma 2, there is a word $z \in \Sigma^{k-|ays|}$ such that $x_1 sz \in S^{\otimes} \Rightarrow y_1 aysz \in S^{\otimes} \Rightarrow aysz \in S \Rightarrow sz \in L_{[y_1 ay]}$. But as S is a k-ps-full set, $bysz \notin S$, therefore, $y_2 bysz \notin S$. Then, as $y_2 by \in S^{\otimes}$, we have that $sz \notin L_{[y_2 by]}$.

It follows that $S^{\otimes}$ has at least $1 + |\Sigma| + \ldots + |\Sigma|^{k-1}$ non-sink states (classes) that are not pairwise equivalent. Moreover, as the set S_1 of the length $k-1$ prefixes of S is equal to Σ^{k-1}, we have that $|S_1| = |\Sigma|^{k-1}$, which implies that $1 + |\Sigma| + \ldots + |\Sigma|^{k-1} + 1 = U(S)$. Hence, $\mathrm{sc}(S^{\otimes}) \geq U(S)$. □

3. Balanced languages

In this section, the set S is the *d-balanced subword constraint* of length k, that is, the set of all words w of length k over $\{0, 1\}$ such that $||w|_0 - |w|_1| \leq d$. Moreover, we require that $||w|_0 - |w|_1| = d$ for at least one word w in S. Here, for any alphabet symbol b, $|w|_b$ is the number of b's occurring in the word w. The quantity $|w|_0 - |w|_1$ is called the *balance* of the word w and is denoted by $\mathrm{bal}(w)$.

Our study of the balance constraint is motivated by the importance of the gc-ratio constraint in DNA computing as explained in the introduction. Although we assume that the constraint S is over the alphabet $\{0, 1\}$, we can use the results herein for other alphabets as well. For example, consider the gc-ratio constraint T over the DNA alphabet $\{a, c, g, t\}$ such that T consists of all words w of some

length k with $-d \leq |w|_a + |w|_t - |w|_c - |w|_g \leq d$. In the next theorem we establish that the automata accepting $T^{\otimes}$ can be studied via the automata accepting $S^{\otimes}$.

Theorem 3 *Let β be any surjective mapping of Σ onto $\{0,1\}$ that is extended to a morphism of Σ^* into $\{0,1\}^*$. Let S be the d-balanced subword constraint of some length k and let T be the subword constraint*

$$\{w \in \Sigma^k \mid -d \leq \mathrm{bal}(\beta(w)) \leq d\}.$$

We have that $\mathrm{sc}(T^{\otimes}) = \mathrm{sc}(S^{\otimes})$.

Proof. It is sufficient to show the following two statements: (i) for any automaton A accepting $T^{\otimes}$ there is an automaton A^{β} accepting $S^{\otimes}$ such that A and A^{β} have the same states; and (ii) for any automaton B accepting $S^{\otimes}$ there is an automaton $B^{-\beta}$ accepting $T^{\otimes}$ such that B and $B^{-\beta}$ have the same states.

First note that $S = \beta(T)$. Indeed, by the definition of T, $\beta(T) \subseteq S$. Conversely, if $z \in S$, then one verifies that $\beta^{-1}(z) \subseteq T$, therefore, $z \in \beta(T)$. Now, for each automaton A over Σ the automaton A^{β} has exactly the same states and has the transition $(p, \beta(x), q)$ for each transition (p, x, q) of A. It is obvious that, if $w \in L(A)$, then $\beta(w) \in L(A^{\beta})$. Moreover, one can verify that if A accepts $T^{\otimes}$ the automaton A^{β} accepts $S^{\otimes}$. For the second statement, for each automaton B over $\{0,1\}$ the automaton $B^{-\beta}$ over Σ has exactly the same states and, for each transition (p, b, q) of B, the automaton has the transitions (p, x, q) for all $x \in \beta^{-1}(b)$. It should be clear that, if $\beta(z) \in L(B)$ then $z \in L(B^{-\beta})$. Moreover one can verify that if B accepts $S^{\otimes}$, the automaton $B^{-\beta}$ accepts $T^{\otimes}$. □

If w is a word of length k such that $|w|_0 \geq |w|_1$ and $\mathrm{bal}(w) = l$, then $|w|_0 = \frac{k+l}{2}$ and $|w|_1 = \frac{k-l}{2}$. If $|w|_0 \leq |w|_1$ and $\mathrm{bal}(w) = -l$ then $|w|_0 = \frac{k-l}{2}$ and $|w|_1 = \frac{k+l}{2}$. Obviously, as the fraction $\frac{k \pm l}{2}$ evaluates to an integer, we have that l is odd when k is odd and l is even when k is even.

Remark 2

1. *For every binary word w, the integer $|w| - \mathrm{bal}(w)$ is even.*
2. *If S is the d-balanced subword constraint of some length k, then k and d are either both even or both odd.*

Lemma 6 *Let S be the d-balanced subword constraint of some length k. For all words u, v with $uv \in S$, we have that $vuv \in S^{\otimes}$.*

Proof. Any subword w of vuv of length k is of the form sup for some words p, s with $ps = v$. As $\mathrm{bal}(sup) = \mathrm{bal}(ups)$, it follows that w must be in S, therefore $vuv \in S^{\otimes}$. □

Some Notation: For any state q of some automaton, we denote by L_q the language accepted by the automaton when q is used as the start state. For any nonnegative integer m and language L, the expression $\mathrm{Pref}_m(L)$ denotes the set of all prefixes of length m of the words in L.

Lemma 7 *Let R be a regular language, and let w_1, w_2 be two non-equivalent words: $w_1 \not\equiv_R w_2$. If there are a word v and two words u_1, u_2 such that $[u_1 v] = [w_1]$ and $[u_2 v] = [w_2]$, then $u_1 \not\equiv_R u_2$.*

Proof. We use the terminology in the paragraph preceding Theorem 2. As the states (classes) $[w_1], [w_2]$ are not equivalent we can assume, without loss of generality, that there is a word w in $L_{[w_1]} - L_{[w_2]}$. Hence, $[w_1 w]$ is a final state, but $[w_2 w]$ is not. Now the word vw must be in $L_{[u_1]}$, but not in $L_{[u_2]}$ – else $[u_2 vw] = [w_2 w]$ would be a final state. Hence, u_2 is not equivalent to u_1. □

Lemma 8 *Let S be the d-balanced subword constraint of some length k. For any two different words w, w' in $\mathrm{Pref}_{k-d}(S)$, we have that $w \not\equiv_{S^\otimes} w'$.*

Proof. First note that every word $w \in \mathrm{Pref}_{k-d}(S)$ is of even length $k-d$ and there is a word x of length d such that wx is in S. Moreover, as $\mathrm{bal}(wx) = \mathrm{bal}(w) + \mathrm{bal}(x)$ and $-d \leq \mathrm{bal}(x) \leq d$, we have that $\mathrm{bal}(w)$ is an even integer in $\{-2d, -2d+2, \ldots, -2, 0, 2, \ldots, 2d\}$. The statement follows from the following sequence of facts.

1. If $\mathrm{bal}(w) = 2h$, with $h > 0$, then $0^{d-h}1^h \in L_{[w]}$ and $0^{d-h+1}1^{h-1} \notin L_{[w]}$, since $\mathrm{bal}(w0^{d-h}1^h) = 2h + d - h - h = d$ and $\mathrm{bal}(w0^{d-h+1}1^{h-1}) = 2h + d - h + 1 - h + 1 = d + 2$.

2. If $\mathrm{bal}(w) = -2h$, with $h > 0$, then $1^{d-h}0^h \in L_{[w]}$ and $1^{d-h+1}0^{h-1} \notin L_{[w]}$, since $\mathrm{bal}(w1^{d-h}0^h) = -2h - d + h + h = -d$ and $\mathrm{bal}(w1^{d-h+1}0^{h-1}) = -2h - d + h - 1 + h - 1 = -d - 2$.

3. If $\mathrm{bal}(w) = 2h$, with $h \geq 0$, then $1^d \in L_{[w]}$. Indeed, we have that $wx \in S$, hence, $-d \leq \mathrm{bal}(wx) \leq d$. Also, as $-d \leq \mathrm{bal}(x) \leq d$, we have that $\mathrm{bal}(wx) \geq 2h - d$. Then, $\mathrm{bal}(w1^d) = \mathrm{bal}(w) - d = 2h - d \leq \mathrm{bal}(wx) \leq d$. On the other hand, $\mathrm{bal}(w1^d) = 2h - d \geq -d$.

4. If $\mathrm{bal}(w) = 0$, then $0^d, 1^d \in L_{[w]}$.

5. If two words w, w' in $\mathrm{Pref}_{k-d}(S)$ are such that $\mathrm{bal}(w) \neq \mathrm{bal}(w')$, then $w \not\equiv_{S^\otimes} w'$. Indeed, we distinguish the following cases:

   (a) If $0 < \mathrm{bal}(w) = 2h < \mathrm{bal}(w') = 2h'$ then $0^{d-h}1^h \in L_{[w]}$ and $0^{d-h}1^h \notin L_{[w']}$, since $\mathrm{bal}(w'0^{d-h}1^h) = 2h' + d - h - h = d + 2(h' - h) \geq d + 2$.

   (b) If $0 > \mathrm{bal}(w) = -2h > \mathrm{bal}(w') = -2h'$ then $1^{d-h}0^h \in L_{[w]}$ and $1^{d-h}0^h \notin L_{[w']}$, since $\mathrm{bal}(w'1^{d-h}0^h) = -2h' - d + h + h = -d - 2(h' - h) \leq -d - 2$.

   (c) If $\mathrm{bal}(w) = 2h > 0$ and $\mathrm{bal}(w') = -2h' < 0$ then $1^d \in L_{[w]}$ and $1^d \notin L_{[w']}$, since $\mathrm{bal}(w'1^d) = -2h' - d < -d$.

   (d) If $\mathrm{bal}(w) = 2h > 0$ and $\mathrm{bal}(w') = 0$ then $0^d \notin L_{[w]}$ and $0^d \in L_{[w']}$, since $\mathrm{bal}(w0^d) = 2h + d \geq d + 2 > d$.

   (e) If $\mathrm{bal}(w) = -2h < 0$ and $\mathrm{bal}(w') = 0$ then $1^d \notin L_{[w]}$ and $1^d \in L_{[w']}$, since $\mathrm{bal}(w1^d) = -2h - d \leq -d - 2 < -d$.

6. Here we assume that the words $w, w' \in \mathrm{Pref}_{k-d}(S)$ are different and have the same balance, that is, $\mathrm{bal}(w) = \mathrm{bal}(w')$. We show that $w \not\equiv_{S^\otimes} w'$. Indeed, these words are of the form $x0y$ and $x1y'$. We distinguish the following cases:

(a) $\mathrm{bal}(x0y) = \mathrm{bal}(x1y') = 2h \geq 0$: Here we have that

$$\mathrm{bal}(y0^{d-h}1^h x0) = \mathrm{bal}(x0y) + d - h - h = d$$

$$\text{and } \mathrm{bal}(y'0^{d-h}1^h x0) = (\mathrm{bal}(x1y') + 1) + 1 + d - h - h = d + 2.$$

Hence, $y0^{d-h}1^h x0 \in S$ and $y'0^{d-h}1^h x0 \notin S$, which implies that $y0^{d-h}1^h x0 \not\equiv_{S^\otimes} y'0^{d-h}1^h x0$. Also, Lemma 5 implies that

$$[x0y0^{d-h}1^h x0] = [y0^{d-h}1^h x0] \text{ and } [x1y'0^{d-h}1^h x0] = [y'0^{d-h}1^h x1].$$

Hence, $x0y \not\equiv_{S^\otimes} x0y'$, using Lemma 7.

(b) $\mathrm{bal}(x0y) = \mathrm{bal}(x1y') = -2h < 0$: this case is symmetric to the previous one using the balance of the words $y1^{d-h}0^h x1$ and $y'1^{d-h}0^h x1$.

□

Theorem 4 *Let S be the d-balanced subword constraint of some length k.*

1. *If $d = 1$ and $k = 2r + 1$ then* $\mathrm{sc}(S^\otimes) \geq \frac{3r+1}{r+1} \binom{2r}{r}$.
2. *If $d = 0$ and $k = 2r$ then* $\mathrm{sc}(S^\otimes) \geq \binom{2r}{r}$.

Proof. By Lemma 8, all the states (classes) $[w]$ of $S^\otimes$ with $w \in \mathrm{Pref}_{k-d}(S)$ are pairwise nonequivalent. Hence, $\mathrm{sc}(S^\otimes) \geq |\mathrm{Pref}_{k-d}(S)|$. For calculating $|\mathrm{Pref}_{k-d}(S)|$, it is sufficient to count all words w of length $k - d$ with $\mathrm{bal}(w) \in \{-2d, -2d + 2, \ldots, 0, 2, \ldots, 2d\}$. This count is equal to

$$\sum_{h=-d}^{d} Bal(k - d, h),$$

where $Bal(k - d, h)$ is the cardinality of $\{w \in \Sigma^{k-d} \mid |w|_0 - |w|_1 = 2h\}$ which is equal to $\binom{k-d}{h + \frac{k-d}{2}}$. Now, if $d = 1$ and $k = 2r + 1$, the first statement follows when we note that

$$\binom{2r}{r-1} = \binom{2r}{r+1} = \frac{r}{r+1}\binom{2r}{r}.$$

If $d = 0$ and $k = 2r$ the second statement follows immediately. □

We note that for k even and $d = 0$, the above lower bound is of order $\Theta(2^k/\sqrt{k})$, using the fact $\binom{2r}{r} \sim c2^{2r}/\sqrt{r}$, that is, $\binom{2r}{r}$ is asymptotically equal to $c2^{2r}/\sqrt{r}$, for some constant c. On the other hand, the upper bound $U(S)$ of Theorem 1 applied in this case is of order $\Theta(\log k \cdot 2^k/\sqrt{k})$. Indeed, first note that $U(S) = \Theta(|S_1|) + (k - \lceil \log_{|\Sigma|} |S_1| \rceil) \cdot |S_1|$ and the set S_1 in Theorem 1 is of cardinality equal to $|S| = \binom{2r}{r}$. This is because, if we drop the last symbol from any

two different words in S, the resulting words are also different. Now, as $|\Sigma| = 2$ and $\log |S_1| \sim k-(1/2)\log k+\log c$, we have that $(k-\lceil \log_{|\Sigma|} |S_1|\rceil)\cdot|S_1| = \Theta(\log k)|S_1| = \Theta(\log k \cdot 2^k/\sqrt{k})$.

4. Other Subword Constraints

Let the alphabet be $\Sigma = \{a_1, \ldots, a_p\}$, with $p \geq 2$. The continuity constraint C_k is the set of all words of length k containing at least two different letters. Then $C_k^{\otimes}$ is the set of all words containing no run of k identical symbols. Unlike the case of balanced languages, the state complexity of $C_k^{\otimes}$ can be found easily as shown next.

Theorem 5 *For every $k \geq 2$, $\mathrm{sc}(C_k^{\otimes}) = 2 + |\Sigma|(k-1)$.*

Proof. It is sufficient to demonstrate the minimal automaton A for the complement of $C_k^{\otimes}$, which consists of all words containing at least one run of k identical symbols. The states of A are: s, the start state; f, the only final state, which is also a sink state; (a_i, j), for every alphabet symbol a_i and $j \in \{1, \ldots, k-1\}$, which means that A has just read j consecutive a_i's. The transition function is

$$\delta((a_i, j), a_r) = \begin{cases} (a_i, j+1), & \text{if } r = i \text{ and } j < k-1; \\ f, & \text{if } r = i \text{ and } j = k-1; \\ (a_r, 1), & \text{if } r \neq i. \end{cases}$$

One verifies that the automaton A has exactly the required number of states and is minimal. For example, if $j < j'$, we have that $\delta((a_{i'}, j'), a_{i'}^{k-j'}) = f$, but $\delta((a_i, j), a_{i'}^{k-j'}) \neq f$. □

In practice, it is desirable to design DNA languages satisfying two or more constraints. So let S_1 and S_2 be two subword constraints. Here we only discuss briefly the case where both constraints are of the same word length k. We are interested in the language $S_1^{\otimes} \cap S_2^{\otimes}$ satisfying both S_1 and S_2. An automaton for this language can be obtained by using the product construction for the intersection of the two automata for $S_1^{\otimes}$ and $S_2^{\otimes}$. The number of states in the resulting automaton is at most equal to the product of the numbers of states in the automata for $S_1^{\otimes}$ and $S_2^{\otimes}$. Thus,

$$\mathrm{sc}(S_1^{\otimes} \cap S_2^{\otimes}) \leq \mathrm{sc}(S_1^{\otimes})\mathrm{sc}(S_2^{\otimes}).$$

The above upper bound for $\mathrm{sc}(S_1^{\otimes} \cap S_2^{\otimes})$ is not particularly good, however. For example, if $S_1 \subseteq S_2$, then $\mathrm{sc}(S_1^{\otimes} \cap S_2^{\otimes}) = (S_2^{\otimes})$. A better estimate for the state complexity of $S_1^{\otimes} \cap S_2^{\otimes}$ follows from the next lemma whose proof can easily be verified using the definition of the subword closure operation.

Lemma 9 *For any two subword constraints S_1, S_2 of the same length k, we have*

$$S_1^{\otimes} \cap S_2^{\otimes} = (S_1 \cap S_2)^{\otimes}.$$

The above equation raises the question of how the two languages involved are related when the operation of intersection is replaced with union '$\cup$' or catenation '$\cdot$'. In the case of union it is easy to see that

$$S_1^{\otimes} \cup S_2^{\otimes} \subseteq (S_1 \cup S_2)^{\otimes}.$$

Obviously, if $S_1 = S_2$ the above containment becomes an equality. On the other hand, there are cases where the containment is proper. For example, for the set S of Fig. 1 we have that $S = \{001, 011\} \cup \{100, 110\}$, and each of $\{001, 011\}^{\otimes}$ and $\{100, 110\}^{\otimes}$ is finite, whereas $S^{\otimes}$ is infinite.

The case of catenation is more complex. Here we settle the case of catenation when $S_1 = S_2$.

Theorem 6 *Let S be a subword constraint of some length k. For every word w of length other than $2k-1$ we have*

$$w \in (SS)^{\otimes} \longrightarrow w \in S^{\otimes}S^{\otimes}.$$

Proof. We consider three cases. First assume that w is of length at most $2k-2$. Then it can be written as w_1w_2 with each of w_1, w_2 being of length less than k. Hence, $w_1w_2 \in S^{\otimes}S^{\otimes}$. Now assume that w is of length $2k + l$ for some $l \in \{0, \ldots, k-1\}$. Then we can write w as

$$a_1 \cdots a_{k+l} b_1 \cdots b_k,$$

with each a_i and b_j being in Σ. Let $w_2 = b_1 \cdots b_k$. As $a_{l+1} \cdots a_{k+l} w_2$ is a subword of w of length $2k$ we have that $w_2 \in S$ and, therefore, $w_2 \in S^{\otimes}$. We show that $a_1 \cdots a_{k+l} \in S^{\otimes}$. Let

$$v = a_{i+1} \cdots a_{i+k}$$

be any subword of $a_1 \cdots a_{k+l}$ of length k. As $va_{i+k+1} \cdots a_{k+l} b_1 \cdots b_{i+k-l} \in SS$, we have that $v \in S$ as required. The third case is when w is of length at least $3k$. Using Remark 1, it is sufficient to show that $w \in S^{\otimes}$. Let $w = xvy$ with v being a subword of length k. We show that $v \in S$. One at least of x and y, say y, must be of length at least k. Thus we can write $y = y_1y_2$ with $|y_1| = k$, therefore, vy_1 is a factor of w of length $2k$. This implies that $vy_1 \in SS$; hence, $v \in S$ as required. □

Consider the set S of Fig. 1. One can verify that $01010 \in (SS)^{\otimes} - S^{\otimes}S^{\otimes}$, which shows that, in the above theorem, the assumption that the length of w is not $2k-1$ is essential. Using the same S we can show that the containment in the theorem can be proper. Since $(110)(010) \notin SS$, it follows $(011001)(001) \notin (SS)^{\otimes}$, thus

$$(011001)(001) \in S^{\otimes}S^{\otimes} - (SS)^{\otimes}.$$

5. Concluding Remarks

We have argued that the subword closure operation is an interesting concept and deserves to be studied from the point of view of formal language theory. Here in particular, we presented some initial results on the state complexity of this operation, including cases related to DNA coding. We believe that similar studies should be done for various other subword constraints such as those discussed in the introduction.

In closing, we note that the set $S^{\otimes}$ is equal to

$$\Sigma^* - \Sigma^*(\Sigma^k - S)\Sigma^*,$$

that is, the set of all words containing no subword in $\Sigma^k - S$. Given S, one can construct an automaton accepting $\Sigma^* - \Sigma^*(\Sigma^k - S)\Sigma^*$ and this could possibly lead to another general bound for the state complexity of $S^{\otimes}$. As this question appears to be non-trivial, we leave it to the reader for future research.

References

1. B. Cui and S. Konstantinidis, "DNA Coding using the Subword Closure Operation," in [4], pp. 284–289.
2. J. Chen and Reif (eds), *Pre-proceed. 9th International Workshop on DNA-Based Computers, Madison, Wisconsin, 2003; Lecture Notes in Computer Science* **2943**, Springer-Verlag, Berlin Heidelberg New York, 2004.
3. M. Crochemore and C. Hancart," Automata for Matching Patterns," in [13], vol. II, pp. 399-462.
4. M.H. Garzon and H. Yan (eds), *Proceed. 13th International Meeting on DNA Computing, Memphis, USA, 2007; Lecture Notes in Computer Science* **4848**, Springer-Verlag, Berlin Heidelberg, 2008.
5. M. Hagiya and A. Ohuchi (eds), *Pre-proceed. 8th International Workshop on DNA-Based Computers, Saporo, Japan, 2002; Lecture Notes in Computer Science* **2568**, Springer-Verlag, Berlin Heidelberg New York, 2003.
6. N. Jonoska and K. Mahalingam, "Languages of DNA based code words," in [2], pp. 58–68.
7. N. Jonoska and N.C. Seeman (eds), *Pre-proceed. 7th International Workshop on DNA-Based Computers, Tampa, Florida, 2001; Lecture Notes in Computer Science* **2340**, Springer-Verlag, Berlin Heidelberg, 2002.
8. S. Kobayashi, T. Kondo, and M. Arita, "On template method for DNA sequence design," in [5], pp. 205–214.
9. L. Kari and S. Konstantinidis, "Descriptional complexity of error/edit systems," *Journal of Automata, Languages and Combinatorics* **9** (2004) 2/3 293–309.
10. L. Kari, S. Konstantinidis and P. Sosík, "Bond-free languages: formalizations, maximality and construction methods," *International Journal of Foundations of Computer Science* **16** (2005) 1039–1070.
11. L. Robbins, "Modelling cryptographic systems," Ph. D. Thesis, Department of Computer Science, University of Western Ontario, London, Canada, 1999.
12. K.H. Rosen, J.G. Michaels, J.L. Gross, J.W. Grossman, and D.R. Shier (eds), *Handbook of Discrete and Combinatorial Mathematics* (CRC Press LLC, Boca Raton, Florida, 2000).
13. G. Rozenberg and A. Salomaa (eds), *Handbook of Formal Languages* (Springer-Verlag, Berlin, 1997).
14. F. Tanaka, M. Nakatsugawa, M. Yamamoto, T. Shiba, and A. Ohuchi, "Developing support system for sequence design in DNA computing," in [7], pp. 129–137.
15. S. Yu, "Regular Languages," in [13], Vol I, pp. 41–110.