k-involution codes and related sets

Lila Kari *

Kalpana Mahalingam †

University of Western Ontario

Department of Computer Science

London, ON N6A5B7

Canada

Abstract

This study was motivated by the problem of optimally encoding information on DNA for biocomputational purposes. Our formalization of intermolecular hybridization (binding) with bulges led to the notion, interesting in its own right, of k-involution codes. An involution code refers to any of the generalizations of the classical notion of codes in which the identity function is replaced by an involution function. (An involution function θ is such that θ^2 equals the identity. An antimorphic involution is the natural formalization of the notion of DNA complementarity.) We namely define and study the notions of k-θ-prefix, k-θ-suffix and k-θ-bifix codes. We also extend the notion of k-insertion set and k-deletion set of a language to incorporate the notion of an involution function. Thus, to an involution map θ and a language L, we associate a set k-θ-ins(L) (k-θ-del(L)) with the property that its k-insertion (k-deletion) into any word of L yields words which belongs to $\theta(L)$. We study the properties of these languages and their connection to involution codes.

Keywords and phrases : Codes, Waston-Crick involution, DNA computing, insertion, deletion

1. Introduction

An essential step of any DNA computation is encoding the input data on single or double DNA strands. Due to the biochemical properties of DNA, complementary single strands can bind to one another forming double stranded DNA. In practical biocomputation experiments, data-encoding DNA strands can potentially interact in undesirable ways result-

**E-mail*: `lila@csd.uwo.ca`

†*E-mail*: `kalpana@csd.uwo`

Journal of Discrete Mathematical Sciences & Cryptography
Vol. 10 (2007), No. 4, pp. 485–503

ing in certain non-specific hybridizations and posing potential problems for the results of the biocomputation. Several attempts have been made to address the problem of encoding information on DNA for biocomputing and various solutions have been proposed. A common approach has been to use the Hamming distance [4-6, 17]. Experimental separation of strands with "good" sequences that avoid intermolecular cross hybridization was reported in [2, 3]. In [7], Kari *et al.* introduced a theoretical approach to the problem of designing code words. Theoretical properties of languages that avoid certain undesirable hybridizations were discussed in [8, 12, 14-16].

One such unwanted hybridization is that of the hairpin structure. The notion of a hairpin structure was formalized and its coding properties and relations between hairpin-free codes and other type of codes have been discussed in [8].

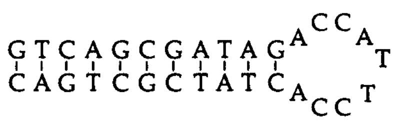

Figure 1

A simple hairpin loop with a bulge

Certain algebraic properties were discussed in [10]. Note that a simple hairpin structure is formed when a portion of a DNA strand binds with its complement within the same strand (see Figure 1) forming a "bulge" within the strand. Hence a hairpin structure is a particular case of a bulge: an intramolecular hybridization with a bulge. Similar situations can happen with intermolecular hybridizations as seen in Figure 2.

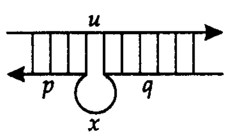

Figure 2

An intermolecular hybridization with a bulge between the DNA single strands u and pxq. Note that u is Watson-Crick complementary to pq while x represents the bulge

In this paper we extend the concept of hairpins (bulges in intramolecular hybridizations) to intermolecular hybridizations with bulges similar to the one in Figure 2. This extends the notion of θ-outfix-code defined in [12], where θ represents a morphic or antimorphic involution. (An involution function θ is such that θ^2 equals the identity. An antimorphic involution is the natural formalization of the notion of DNA complementarity.) The formalization of the notion of languages free of certain intermolecular hybridization with bulges leads to the concept, interesting in its own right, of k-involution codes. It turns out that these k-involution codes are generalizations of the k-prefix codes defined in [9] and moreover they can be studied using the operations of k-insertion and k-deletion of languages [11].

This paper defines and investigates k-involution codes and related sets. The paper is organized as follows: In Section 2 we formalize the notion of certain unwanted intermolecular hybridization with bulges. In Section 3 we extend the concept of k-prefix and k-suffix codes to involution k-prefix and involution k-suffix codes and show that these codes are just a special case of the codes defined in Section 2. We also study several properties of such codes. In particular we show that for a morphic involution θ, the class of k-θ-prefix and k-θ-suffix codes is closed under concatenation.

In Section 4 we define for a language L and an involution θ, the k-θ-insertion set of a language L denoted by k-θ-ins(L) as the language consisting of the words with the property that their k-insertion into any word of L yields a word in $\theta(L)$. We study the connection between this set and the involution codes and study some of its properties. In Section 5, the k-θ-deletion set of a language L denoted by k-θ-del(L) is defined as the language consisting of the words with the property that their k-deletion from any word of $\theta(L)$ yields a word in L. We construct this set using the dual operation of dipolar k-deletion.

2. Involution codes with bulges

In this paper we use the following notations. By Σ we denote the finite nonempty alphabet set and by Σ^* the free monoid generated by Σ under the catenation operation. Any word over Σ is a finite sequence of letters from Σ and by 1 we denote the empty word. The length of a word $u \in \Sigma^*$ is the number of letters in u and is denoted by $|u|$. Throughout the rest of the paper, we focus on sets $L \subseteq \Sigma^+$ that are codes meaning that every word in L^+ can be written uniquely as a product of words in L (i.e., L^+ is a free semigroup generated by L). For the background on codes we refer the reader to [1, 18]. An involution $\theta : \Sigma \mapsto \Sigma$ is a function such that

$\theta^2 = I$ where I is the identity function and can be extended to a morphic involution on Σ^* if for all $u, v \in \Sigma^*$, $\theta(uv) = \theta(u)\theta(v)$ or an antimorphic involution if $\theta(uv) = \theta(v)\theta(u)$. For more on involution codes we refer the reader to [7, 12, 8].

The antimorphic involution θ defined on the DNA alphabet $\{A, G, C, T\}$ as $\theta(A) = T, \theta(C) = G$ has recently been of particular interest, since it succinctly formalizes the notion of DNA single strand Watson-Crick complementarity. This DNA involution has been extensively used for theoretical studies of DNA languages (languages over the DNA alphabet) and properties that make them suitable for biocomputations. Notions such as θ-infix-code, θ-comma-free-code and θ-strict-code have been thus defined and studied in [7, 8,12,14-16].

We follow this theoretical approach and introduce the concept of intermolecular hybridization with a bulge. This concept is a formalization of DNA secondary structures that are known to form in practical wet lab experiments. We formalize the concept of a DNA language L that avoids such mismatched pairings. We namely focus on generalizing the θ-infix and θ-comma-free codes. Similar generalizations of other type of codes (for example, θ-sticky-free, θ-k-codes, θ-solid etc.) defined in [12, 8, 14, 15] can also be defined.

With this purpose in mind, we recall the following definitions [9, 11]. For $u \in \Sigma^+$, we define:

$$\begin{aligned}
\mathrm{Ins}(u) &= \{u_1 v u_2 : v \in \Sigma^*, u_1, u_2 \in \Sigma^*, u = u_1 u_2\}, \\
\mathrm{Del}(u) &= \{u_1 u_3 : u_1, u_2, u_3 \in \Sigma^*, u = u_1 u_2 u_3\}, \\
\mathrm{Subs}(u) &= \{u_1 v u_3 : v \in \Sigma^*, u_1, u_2, u_3 \in \Sigma^*, u = u_1 u_2 u_3, |u_2| = |v|\}.
\end{aligned}$$

We extend the above definitions to a language $L \subseteq \Sigma^+$ in the natural way:

$$\begin{aligned}
\mathrm{Ins}(L) &= \bigcup_{u \in L} \mathrm{Ins}(u), \\
\mathrm{Del}(L) &= \bigcup_{u \in L} \mathrm{Del}(u), \\
\mathrm{Subs}(L) &= \bigcup_{u \in L} \mathrm{Subs}(u).
\end{aligned}$$

The following definition generalizes the concepts of θ-infix, θ-comma-free and θ-strict codes to include hybridizations with bulges.

Definition 1. Let Υ = Ins or Subs or Del and let $L \subseteq \Sigma^+$. Let θ be either a morphic or an antimorphic involution. Then

1. L is θ-Υ-infix if and only if $L \cap (\Sigma^+ \Upsilon(\theta(L))\Sigma^* \cup \Sigma^* \Upsilon(\theta(L))\Sigma^+) = \emptyset$
2. L is θ-Υ-comma-free if and only if $L^2 \cap \Sigma^+ \Upsilon(\theta(L))\Sigma^+ = \emptyset$.
3. L is θ-Υ-strict iff $L \cap \Upsilon(\theta(L)) = \emptyset$.

Example 1. Let $\Sigma = \{a, b\}$ and θ be a morphic involution on Σ^* such that $\theta(a) = b$ and $\theta(b) = a$. Take $L = \{a^n b^n : n \geq 1\}$. Then $\theta(L) = \{b^n a^n : n \geq 1\}$. It is easy to see that L is θ-ins-infix.

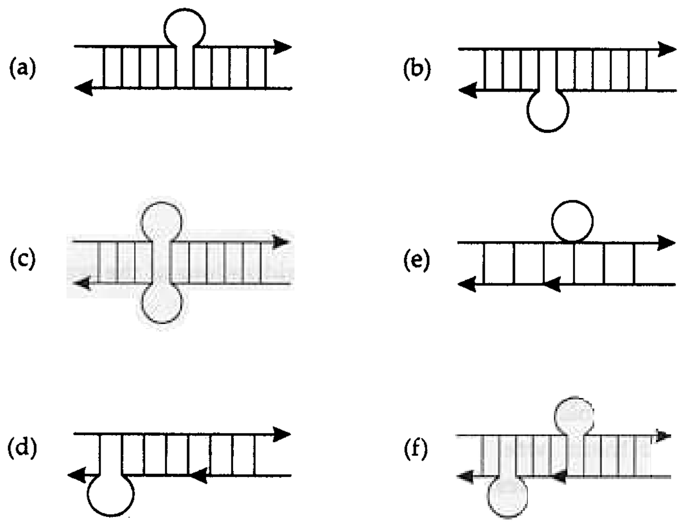

Figure 3

θ-Υ-infix codes avoid unwanted hybridizations of the type: (a) If Υ = Ins, (b) If Υ = Del and (c) If Υ = Subs. θ-Υ-comma-free codes avoid hybridizations of the type in, (d) If Υ-Ins, (e) If Υ = Del, and (f) If Υ = Subs

Figure 3 and 4 illustrate the type of unwanted hybridizations avoided by DNA languages possessing one of the properties defined in Definition 1 (θ in this case is the DNA involution). Note that θ-ins-infix codes avoid bindings of the type in Figure 3(a), θ-del-infix codes avoid hybridizations of the type in Figure 3(b) and θ-subs-infix codes avoid hybridizations of the type in Figure 3(c). Similarly, θ-ins-comma-free codes avoid bindings of the type in Figure 3(d), θ-del-comma-free codes avoid hybridizations of the type in Figure 3(e) and θ-subs-comma-free codes avoid hybridizations of the type in Figure 3(f), while θ-ins-strict codes avoid bindings of the type in Figure 4(a), θ-del-strict codes avoid hybridizations of the type in Figure 4(b) and θ-subs-strict codes avoid hybridizations of the type in Figure 4(c). Note that since $\theta(L) \subseteq \Upsilon(\theta(L))$, we have that if L is θ-Υ-infix then L is θ-infix. If L is θ-Υ-comma-free then L is θ-comma-free. Also L is θ-Υ-infix(comma-free) iff $\theta(L)$ is θ-Υ-infix(comma-free).

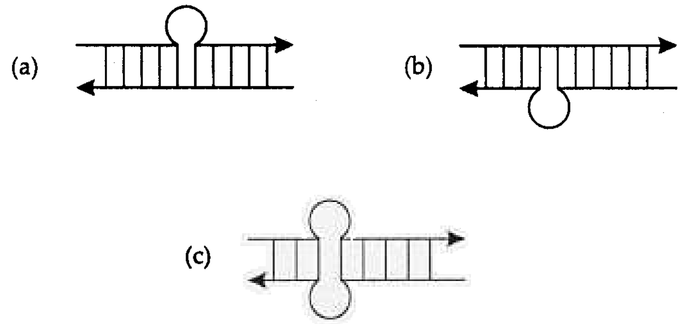

Figure 4

θ-Υ-strict codes avoid unwanted hybridizations of the type (a) If Υ = Ins, (b) If Υ = Del, and (c) If Υ = Subs

Most of the results obtained for θ-infix codes (θ-comma-free codes) (see [8, 12, 7]) hold also for θ-Υ-infix(comma-free) codes hence we do not include them here. We only list a few closure properties of θ-Υ-infix(comma-free) codes. For example Proposition 1 can be proved using techniques that are very similar to those used for θ-comma-free and θ-infix codes.

Proposition 1. *The following are equivalent*

1. *L is θ-Υ-comma-free.*
2. *L^+ is θ-Υ-infix.*
3. *L^+ is θ-Υ-comma-free.*

Since the proof techniques and the results about θ-Υ-infix (θ-Υ-comma-free) codes are very similar to the ones that already exist for θ-infix and θ-comma-free codes, we focus herein on a special case. In the next section we namely investigate a special case of θ-Υ-strict codes that puts some restrictions on the length of words involved. It turns out that these codes can be defined using the k-insertion and k-deletion operation.

3. k-involution codes

k-involution codes can be defined using the operation of k-insertion as detailed in the following. Given two words $u, v \in \Sigma^*$, the insertion of v in to u is defined as $u \leftarrow v = \{u_1 v u_2 : u = u_1 u_2\}$. The notion of k-insertion was introduced in [9] under the name of k-catenation. The operation of k-insertion restricts the generality of insertion by allowing words to be inserted in at most $k+1$ positions. For a given $k \geq 1$, the left and the right k-insertions of v into u (the right and the left k-catenation of v in to u) are defined as follows:

$$u \leftarrow_r^k v = \{u_1 v u_2 : u = u_1 u_2, |u_2| \leq k, u_1, u_2, v \in \Sigma^*\},$$
$$u \leftarrow_l^k v = \{u_1 v u_2 : u = u_1 u_2, |u_1| \leq k, u_1, u_2, v \in \Sigma^*\}.$$

The left and the right insertion of a language L_2 in to L_1 can be defined in a natural fashion.

The concept of k-prefix code was introduced and studied in [9] and the concept of k-suffix code was introduced and studied in [13]. We recall the following definitions:

Definition 2. Let $S \subseteq \Sigma^*$ be a nonempty language

1. S is a k-prefix code if $u \in S$ and $u \leftarrow_r^k x \cap S \neq \emptyset$ then $x = 1$
2. S is a k-suffix code if $u \in S$ and $u \leftarrow_l^k x \cap S \neq \emptyset$ then $x = 1$

In this section we generalize the class of k-prefix and k-suffix codes to involution k-prefix(k-θ-prefix) and involution k-suffix (k-θ-suffix) codes. An involution code refers to any of the generalizations of the classical notion of codes that replace the identity function with the involution function as explained in [7, 12, 8]. Note that when θ is identity a k-θ-prefix (suffix) code is nothing but a k-prefix(suffix) code. Also it is rather easy to see that k-θ-prefix and k-θ-suffix-codes (see Figure 5) are a special case of θ-Υ-strict codes (Figure 4) when Υ = Ins.

Definition 3. Let u, v be words over the alphabet Σ and let θ be a morphic or antimorphic involution.

1. A k-θ-prefix code is a non empty language $P \subseteq \Sigma^+$ such that $u \in P$ and $\theta(u) \leftarrow_r^k v \cap P \neq \emptyset$ implies $v = 1$.
2. A k-θ-suffix code is a non empty language $S \subseteq \Sigma^+$ such that $u \in S$ and $\theta(u) \leftarrow_l^k v \cap S \neq \emptyset$ implies $v = 1$.

3. A set L is called a k-θ-bifix code iff L is both a k-θ-prefix and a k-θ-suffix code.

Note that a k-θ-prefix code (k-θ-suffix code) avoids hybridizations of the type illustrated in Figure 5 where $|q| \leq k$ ($|p| \leq k$). Thus, a k-θ-prefix code (k-θ-suffix code) is a special case of a θ-Υ-strict code (Figure 4(a)). Indeed in the latter, no restriction is placed on the lengths of the words involved.

In the following we investigate certain closure properties of k-θ-prefix, k-θ-suffix and k-θ-bifix codes.

Lemma 1. *Let* $L \subseteq \Sigma^+$.

1. *For a morphic involution* θ, *L is k-θ-prefix code (suffix) iff $\theta(L)$ is k-θ-prefix code (suffix).*
2. *For an antimorphic involution* θ, *L is k-θ-prefix code (suffix) iff $\theta(L)$ is k-θ-suffix code (prefix).*
3. *L is k-θ-bifix code iff $\theta(L)$ is k-θ-bifix code.*

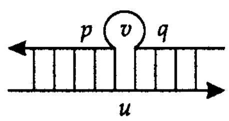

Figure 5
k-θ-prefix codes avoid hybridization between words u and pvq, where $\theta(u) = pq$ with $|q| \leq k$, while k-θ-suffix codes avoid such hybridizations where $\theta(u) = pq$ with $|p| \leq k$

Proof. Let θ be morphic involution and L be a k-θ-prefix code. Suppose there exists $\theta(u) \in \theta(L)$ such that $\theta(\theta(u)) = u_1u_2$ with $|u_2| \leq k$ and $u_1vu_2 \in \theta(L)$ for some $v \in \Sigma^*$. We need to show that $v = 1$. Note that $u_1vu_2 \in \theta(L)$ iff $\theta(u_1vu_2) \in L$ iff $\theta(u_1)\theta(v)\theta(u_2) \in L$ which implies $\theta(v) = 1$ since L is k-θ-prefix. Similarly we can prove the other direction and also the other statements. □

Remark that a k-θ-prefix(suffix) code is also an m-θ-prefix(suffix) code for $m \leq k$. Note that if $k = 0$, then θ-prefix(suffix) codes become θ-prefix (suffix) codes. Recall that, $L \subseteq \Sigma^+$ is called a θ-prefix (θ-suffix) code if $L \cap \theta(L)\Sigma^+ = \emptyset (L \cap \Sigma^+\theta(L) = \emptyset)$. In the next proposition we show that the class of all k-θ-prefix (suffix) codes is closed under arbitrary concatenation when θ is a morphic involution.

Proposition 2. *When θ is morphic involution the class of k-θ-prefix(suffix) codes is closed under concatenation.*

Proof. We prove the proposition for k-θ-prefix-codes. Let P, Q be two k-θ-prefix codes. Let $a \in P$ and $b \in Q$ such that $\theta(ab)[k]_r v \in PQ$. We need to show that $v = 1$. We have the two following cases:

(i) $\theta(a_1)v\theta(a_2)\theta(b) \in PQ$ with $|\theta(a_2)\theta(b)| \leq k$ and $\theta(a) = \theta(a_1 a_2)$.
(ii) $\theta(a)\theta(b_1)v\theta(b_2) \in PQ$ with $|\theta(b_2)| \leq k$ and $\theta(b) = \theta(b_1 b_2)$.

Consider *Case* (i). Let $xy = \theta(a_1)v\theta(a_2)\theta(b) \in PQ$ such that $x \in P$ and $y \in Q$. Then

1. $x = \theta(a_1')$ and $y = \theta(a_1'')v\theta(a_2)\theta(b)$ with $b, y \in Q$ and $|\theta(a_2)\theta(b)| \leq k$. Since $y \in Q$ and Q is k-θ-prefix, we have $|\theta(b)| \leq k$ and $\theta(a_1'')v\theta(a_2) = 1$.
2. $x = \theta(a_1)v_1$ and $y = v_2\theta(a_2)\theta(b)$ with $b, y \in Q$ and $|\theta(b)| \leq k$. Since $y \in Q$ and Q is k-θ-prefix, we have $v_2\theta(a_2) = 1$ which implies $v = v_1$ and $\theta(a) = \theta(a_1)$. Since $x, a \in P$ and P is k-θ-prefix with $|1| \leq k$, we have $v_1 = v = 1$.
3. $x = \theta(a_1)v\theta(a_2')$ and $y = \theta(a_2'')\theta(b)$ with $y, b \in Q$ and $|\theta(b)| \leq k$. Since Q is k-θ-prefix, we have $\theta(a_2'') = 1$ and since $|\theta(a_2')| \leq k$ with $x, a_1 a_2' \in P$, we have $v = 1$.
4. $x = \theta(a_1)v\theta(a_2)\theta(b_1)$ and $y = \theta(b_2)$ with $b, y \in Q$ and $|\theta(b_2)| \leq k$ (i.e.) we have $\theta(b_2), b_1 b_2 \in Q$ which implies $b_1\theta(\theta(b_2)) \in Q$ and hence $b_1 = 1$ since $x = \theta(a_1)v\theta(a_2)$ and P is k-θ-prefix with $|\theta(a_2)| \leq k$ we have $v = 1$.

A similar proof works for *Case* (ii). Hence PQ is a k-θ-prefix code.

The above proposition does not hold when θ is an antimorphic involution and L is either a k-θ-prefix code or a k-θ-suffix code. For example consider the DNA alphabet $\Delta = \{A, G, C, T\}$ and let $X_1 = \{AGC, G\}$ and $X_2 = \{GCT, T, C\}$. Then for an antimorphic involution θ that maps $A \mapsto T$ and $C \mapsto G$ and viceversa, both X_1 and X_2 are k-θ-suffix codes for $k = 1$. Note that $X_1X_2 = \{AGCGCT, AGCT, AGCC, GGCT, GT, GC\}$ and $AGCT \in \theta(X_1X_2)$ while $AGCT \leftarrow_l^k GC \in X_1X_2$ for $k = 1$, i.e., $AGCGCT \in X_1X_2$ and hence X_1X_2 is not a k-θ-suffix code. □

In the next proposition we show that for an antimorphic involution and for a k-θ-bifix code L, any power of L is also a k-θ-bifix code.

Proposition 3. *When θ is antimorphic involution, if L is a k-θ-bifix code, then L^n is a k-θ-bifix code for all $n \geq 1$.*

Proof. By induction on n. □

Base case. Let L be a k-θ-bifix code. For $n = 1$, L^n is a k-θ-bifix code. We show for $n = 2$. Suppose L^2 is not a k-θ-bifix code, then there exists $x_1, x_2 \in L$ such that $\theta(x_1x_2)[k]_r v \in L^2$. Then either $\theta(x_2)\theta(x_{12})v\theta(x_{11}) \in L^2$ or $\theta(x_{22})v\theta(x_{21})\theta(x_{11}) \in L^2$ or $\theta(x_2)v\theta(x_1) \in L^2$. We only show for the first case. Let $\alpha\beta = \theta(x_2)\theta(x_{12})v\theta(x_{11}) \in L^2$ with $|\theta(x_2)\theta(x_{12})| \leq k$. Then we have the following cases:

- $\alpha = \theta(x_2'')$ and $\beta = \theta(x_2')\theta(x_{12})v\theta(x_{11})$ with $|\theta(x_2')\theta(x_{12})| \leq k$.
- $\alpha = \theta(x_2)\theta(x_{12}')$ and $\beta = \theta(x_{12}'')v\theta(x_{11})$ with $|\theta(x_{12}'')| \leq k$.
- $\alpha = \theta(x_2)\theta(x_{12})v_1$ and $\beta = v_2\theta(x_{11})$.
- $\alpha = \theta(x_2)\theta(x_{12})v\theta(x_{11}'')$ and $\beta = \theta(x_{12}')$.

All cases contradict our assumption that L is a k-θ-bifix code. Hence L^2 is a k-θ-bifix code.

Induction step. Assume that L^m is a k-θ-bifix code for some $m \geq 1$. Let $a = a_1 \ldots a_{m+1} \in L^{m+1}$ such that $a_i \in L$ for all $1 \leq i \leq m+1$ and $\theta(a)[k]_r v \in L^{m+1}$. We need to show that $v = 1$. We have the following $m+1$ cases.

Case (1). We have $\theta(a_{m+1,1})v\theta(a_{m+1,2})\theta(a_m)\ldots\theta(a_1) \in L^{m+1}$ such that $|\theta(a_{m+1,2})\theta(a_m)\ldots\theta(a_1)| \leq k$.

Cases (i). $(2 \leq i \leq m)$: $\theta(a_{m+1})\ldots\theta(a_{i,1})v\theta(a_{i,2})\theta(a_{i-1})\ldots\theta(a_1) \in L^{m+1}$ with $|\theta(a_{i,2})\theta(a_{i-1})\ldots\theta(a_1)| \leq k$ for $2 \leq i \leq m$.

Case $(m+1)$: $\theta(a_{m+1})\theta(a_m)\ldots(a_{1,1})v\theta(a_{1,2}) \in L^{m+1}$ with $|\theta(a_{1,2})| \leq k$.

Consider *Case* (i).

Let $xy = \theta(a_{m+1,1})v\theta(a_{m+1,2})\theta(a_m)\ldots\theta(a_1)$ such that $xy \in L^{m+1}$ $x \in L$ and $y \in L^m$ with $|\theta(a_{m+1,2})\theta(a_m)\ldots\theta(a_1)| \leq k$. Then we have,

1. $x = \theta(a_{m+1,1}')$ and $y = \theta(a_{m+1,1}'')v\theta(a_{m+1,2})\theta(a_m)\ldots(a_1)$ with $\theta(a_{m+1,1}) = \theta(a_{m+1,1}')\theta(a_{m+1,1}'')$ which implies $v = 1$ since L^m is k-θ-bifix.
2. $x = \theta(a_{m+1,1})v_1$ and $y = v_2\theta(a_{m+1,2})\theta(a_m)\ldots\theta(a_1)$ with $v = v_1v_2$ and $|\theta(a_m)\ldots\theta(a_1)| \leq k$. Since L^m is k-θ-bifix, we have

$v_2\theta(a_{m+1,2}) = 1$ and hence $\theta(a_{m+1}) = \theta(a_{m+1,1})$ and $v = v_1$. Since L is k-θ-bifix, we have $v = 1$.

3. $x = \theta(a_{m+1,1})v\theta(a'_{m+1,2})$ and $y = \theta(a''_{m+1,2})\theta(a_m)\dots\theta(a_1)$. Since L^m is k-θ-bifix we have $\theta(a''_{m+1,2}) = 1$ and hence $v = 1$ since L is k-θ-bifix.
4. $x = \theta(a_{m+1,1})v\theta(a_{m+1,2})\theta(a''_m)$ and $y = \theta(a'_m)\dots\theta(a_1)$. Since $y = \theta(a'_m)\dots\theta(a_1)$ which belongs to L^m, we have $\theta(a_1\dots a'_m) \in L^m$ and hence $\theta(\theta((a_1\dots a'_m))a''_m \in L^m$ which implies $a''_m = 1$ since L^m is k-θ-bifix. Hence $x = \theta(a_{m+1,1})v\theta(a_{m+1,2})$ with $|\theta(a_{m+1,2})| \leq k$ and since L is k-θ-bifix we have $v = 1$.

The other cases can be proved in a similar fashion and hence L^{m+1} is a k-θ-prefix code. We can similarly show that if L is a k-θ-suffix code then L^{m+1} is a k-θ-suffix code. □

Lemma 2. *Let θ be a morphic involution and let L_1 and L_2 be non empty languages over Σ^+ such that $L_i \cap \theta(L_i) \neq \emptyset$ for $i = 1, 2$. Then the following are true.*

1. *If L_1L_2 is k-θ-prefix code, then L_2 is a k-θ-prefix code.*
2. *L_1L_2 is k-θ-suffix code, then L_1 is a k-θ-suffix code.*

Proof. Let L_1L_2 be k-θ-prefix code. Let $u \in L_2$ such that $u = u_1u_2$ and $\theta(u_1)v\theta(u_2) \in L_2$ with $|\theta(u_2)| \leq k$. We need to show that $v = 1$. Choose $x \in L_1$ such that $x \in L_1 \cap \theta(L_1)$. Then $x\theta(u_1)v\theta(u_2) \in L_1L_2$ with $x\theta(u_1)\theta(u_2) \in \theta(L_1L_2)$. Since L_1L_2 is k-θ-prefix, we have $v = 1$. Hence L_2 is k-θ-prefix code. Similarly we can show that L_1 is k-θ-suffix codes, when L_1L_2 is a k-θ-suffix code. □

Corollary 1. *Let θ be a morphic involution and let L_i, $i = 1, 2, \dots, m$ be non empty languages over Σ such that $L_i \cap \theta(L_i) \neq \emptyset$ for all $i = 1, 2, \dots, m$. Then the following are true.*

1. *If $L_1L_2\dots L_m$ is k-θ-prefix code, then $L_2L_3\dots L_m$, $L_3\dots L_m$, $L_{m-1}L_m$ and L_m are k-θ-prefix codes.*
2. *If $L_1L_2\dots L_m$ is k-θ-suffix code, then $L_1L_2\dots L_{m-1}$, $L_1\dots L_{m-2}$, L_1L_2 and L_1 are k-θ-suffix codes.*

Proposition 4. *Let $L \subseteq \Sigma^+$ be such that $L \cap \theta(L) \neq \emptyset$. Then,*

1. *If L^m is k-θ-prefix code for $m \geq 1$, then L is k-θ-prefix code.*
2. *If L^m is k-θ-suffix code for $m \geq 1$, then L is k-θ-suffix code.*
3. *If L^m is k-θ-bifix code for $m \geq 1$, then L is k-θ-bifix code.*

Proof. Assume that L^m is a k-θ-prefix code for some $m \geq 1$. Suppose there exists a $u \in L$ such that $\theta(u) \leftarrow_r^k v \cap L \neq \emptyset$ for some $v \in \Sigma^*$. Then we need to show that $v = 1$. The case when θ is a morphic involution is a special case of Corollary 1 when $L_i = L$ for all i. When θ is antimorphism, let $u = u_1 u_2$ then $\theta(u) = \theta(u_2)\theta(u_1)$ and $\theta(u_2)v\theta(u_1) \in L$ with $|\theta(u_1)| \leq k$. Let $z_1, z_2, \ldots, z_{m-1} \in L \cap \theta(L)$ then $z_1 \ldots z_{m-1}\theta(u_2)v\theta(u_1) \in L^m$ which implies $v = 1$ since L^m is k-θ-prefix code. Similar proof works when L^m is k-θ-suffix code. □

4. The k-θ-insertion set of languages

Section 3 studied the notion of k-θ-prefix and k-θ-suffix codes using the operation of k-insertion. This section continues the theoretical investigation of k-insertion by extending the notion of k-insertion set of languages to k-θ-insertion set of languages. We also explore the relation between k-θ-insertion set of languages and the notion of k-θ-prefix and k-θ-suffix codes (see Lemma 4).

Let $L \subseteq \Sigma^+$. To the language L, a set k-ins(L) can be associated consisting of all the words with the following property: their k-insertion into any word of L yields a word belonging to L [11]. Formally k-ins(L) was defined by:

$$k\text{-ins}(L) = \{x \in \Sigma^* : \forall\, u \in L, u = u_1u_2, |u_2| \leq k \Rightarrow u_1xu_2 \in L\},$$

and various properties of k-ins(L) have been investigated in [11]. In a similar fashion, for a morphic or antimorphic involution θ. we associate two sets, left-k-θ-ins(L) and right-k-θ-ins(L) consisting of all words with the following property: their left(respectively right)-k-insertion into any word of L yields a word belonging to $\theta(L)$. Formally, the right-k-θ-insertion set of L (right-k-θ-ins(L)) and the left-k-θ-insertion set of L (left-k-θ-ins(L)) are defined by:

$$\text{right-}k\text{-}\theta\text{-ins}(L) = \{x \in \Sigma^* : \forall\, u \in L, u = u_1u_2, u_1, u_2 \in \Sigma^*, |u_2| \leq k \Rightarrow u_1xu_2 \in \theta(L)\}$$

left-k-θ-ins(L)

$$= \{x \in \Sigma^* : \forall\, u \in L, u = u_1u_2, u_1, u_2 \in \Sigma^*, |u_1| \leq k \Rightarrow u_1xu_2 \in \theta(L)\}.$$

Note that throughout the rest of this paper $\star$ is used to denote either left or right.

Lemma 3. *Let $L \subseteq \Sigma^+$. If θ is a morphic involution then $\theta(\star$-k-θ-ins$(L)) = \star$-k-θ-ins$(\theta(L))$. If θ is an antimorphic involution then $\theta($right-k-θ-ins$(L))$ = left-k-θ-ins(L) and $\theta($left-k-θ-ins$(L))$ = right-k-θ-ins(L).*

Lemma 4. *For a language $L \subseteq \Sigma^+$ we have:*

L is k-θ-prefix code iff right-k-θ-ins$(L) = \{1\}$.

L is k-θ-suffix code iff left-k-θ-ins$(L) = \{1\}$.

Recall that a language $L \subseteq \Sigma^+$ is commutative if the following condition holds: $xuvy \in L$ iff $xvuy \in L$.

Lemma 5. *$L \subseteq \Sigma^+$ is a commutative language iff $\theta(L)$ is a commutative language.*

Proof. Let L be a commutative language. Let $xuvy \in \theta(L)$, then $\theta(xuvy) \in L$. When θ is morphic involution we have $\theta(x)\theta(u)\theta(v)\theta(y) \in L$ and since L is commutative we have $\theta(x)\theta(v)\theta(u)\theta(y) \in L$ and hence $xvuy \in \theta(L)$ and similar proof works for an antimorphic involution θ. Thus $\theta(L)$ is a commutative language. The converse can be proved similarly. □

Proposition 5. *If L is a commutative language, then $\star$-k-θ-ins(L) is also a commutative language.*

Proof. It is sufficient to show that $xuvy \in \star$-k-θ-ins(L) implies $xvuy \in \star$-k-θ-ins(L). If $w \in L$, such that $w = w_1w_2$, $|w_2| \leq k$, then $w_1xuvyw_2 \in \theta(L)$, hence $w_1xvuyw_2 \in \theta(L)$. (Note that L is commutative iff $\theta(L)$ is commutative.) Therefore $xvuy \in \star$-k-θ-ins(L). □

In order to construct, for a given language L, the set $\star$-k-θ-ins(L), we need to introduce the operation of dipolar k-deletion.

Definition 4 ([11]). For u, v words over the alphabet set Σ, the right and the left dipolar k-deletion is defined respectively by:

$$u \rightleftharpoons_r^k v = \{x \in \Sigma^* : u = v_1xv_2, v = v_1v_2, |v_2| \leq k, v_1, v_2 \in \Sigma^*\}$$

and

$$u \rightleftharpoons_l^k v = \{x \in \Sigma^* : u = v_1xv_2, v = v_1v_2, |v_1| \leq k, v_1, v_2 \in \Sigma^*\}$$

In [9], the operation $u \rightleftharpoons_r^k v$ has been introduced under the name of k-deletion and was later called dipolar k-deletion in [11]. The right(left) dipolar-k-deletion erases from u a prefix(suffix) v_1 of any length and a suffix(prefix) v_2 of length $\leq k$ whose catenation $v_1v_2(v_2v_1)$ equals v. The operation can be extended to languages in the natural fashion. If L_1 and L_2 are languages over the alphabet Σ, then the $\star$-dipolar k-deletion of L_2 into L_1 is the language

$$L_1 \rightleftharpoons_\star^k L_2 = \bigcup_{\in L_1, v \in L_2} u \rightleftharpoons_\star^k v.$$

Lemma 6. *For a morphic involution $\theta, \theta(u \rightleftharpoons_\star^k v) = \theta(u) \rightleftharpoons_\star^k \theta(v)$. For an antimorphic involution θ, we have $\theta(u \rightleftharpoons_r^k v) = \theta(u) \rightleftharpoons_l^k \theta(v)$ and $\theta(u \rightleftharpoons_l^k v) = \theta(u) \rightleftharpoons_r^k (v)$.*

Now we are able to construct the set $\star$-k-θ-ins(L) using the $\star$-dipolar k-deletion.

Proposition 6. $\star$-k-θ-*ins*$(L) = ((\theta(L))^c \rightleftharpoons_\star^k L)^c$

Proof. Take $x \in$ right-k-θ-ins(L). Suppose, $x \in ((\theta(L))^c \rightleftharpoons_r^k L)$ then there exists $u_1xu_2 \in (\theta(L))^c$, $u_1u_2 \in L$, $|u_2| \leq k$ such that $x \in u_1xu_2 \rightleftharpoons_r^k u_1u_2$ which is a contradiction as $x \in$ right-k-θ-ins(L) and $u_1u_2 \in L, |u_2| \leq k$, but the right-$k$-$\theta$-insertion of x into u_1u_2 belongs to $(\theta(L))^c$. Conversely, let $x \in ((\theta(L))^c \rightleftharpoons_r^k L)^c$. If $x \notin$ right-k-θ-ins(L), then there exists $u_1u_2 \in L, |u_2| \leq k$ such that $u_1xu_2 \notin \theta(L)$ which implies $u_1xu_2 \in (\theta(L))^c$ and hence $x \in ((\theta(L))^c \rightleftharpoons L)$ which is a contradiction. □

Corollary 2. *If L is regular, then $\star$-k-θ-ins(L) is regular and can be effectively constructed.*

Proof. It has been proven in [9] that if a language L is regular, then $L \rightleftharpoons_\star^k R$ is regular. Since L is regular, $\theta(L)$ is regular and hence $(\theta(L))^c$ is regular which implies $((\theta(L))^c \rightleftharpoons_\star^k L)$ is regular and hence $((\theta(L))^c \rightleftharpoons L)^c$ is regular. Since the right-k-dipolar deletion of two regular languages can be effectively constructed (see [11]), it follows that $\star$-k-θ-ins(L) can be effectively constructed for a regular language L.

The last part of this section introduces the notion of a $\star$-k-θ-ins-closed languages that naturally derives from the $\star$-k-insertion set of a language L. □

Recall that for a language $L \subseteq \Sigma^+$, L is $\star$-k-θ-ins-closed iff $L \subseteq \star$-k-θ-ins(L).

Proposition 7. *L is $\star$-k-θ-ins-closed iff $L \leftarrow_\star^k L \subseteq \theta(L)$.*

Proof. Let L be right-k-θ-ins-closed. Take $x \in L$ and let $u = u_1u_2 \in L$ such that $|u_2| \leq k$. Then as $x \in L \subseteq$ right-k-θ-ins(L), $u_1xu_2 \in \theta(L)$ which implies $L \leftarrow_r^k L \subseteq \theta(L)$. Conversely, let $L \leftarrow_r^k L \subseteq \theta(L)$ and let $x \in L$. To show that $x \in$ right-k-θ-ins(L). Let $u_1u_2 \in L$, $|u_2| \leq k$. Then $L \leftarrow_r^k L \subseteq \theta(L)$ implies that $u_1xu_2 \in \theta(L)$ which implies $x \in$ right-k-θ-ins(L).

Lemma 7. *For a language $L \subseteq \Sigma^+$ we have:*

1. *When θ is morphic involution, L is $\star$-k-θ-ins-closed iff $\theta(L)$ is $\star$-k-θ ins-closed.*
2. *When θ is antimorphic involution, L is left(right)-k-θ-ins-closed iff $\theta(L$ is right(left)-k-θ-ins-closed.*
3. *For $k = 0$, if L is $\star$-k-θ-ins-closed then $L^n, n \geq 1$ is $\star$-k-θ-ins-closed.*

A $\star$-k-θ-ins-closed language L is said to be minimal if $L' \subseteq L$ with L' a $\star$-k-θ-ins-closed language, implies $L = L'$. The next result shows that a $\star$-k-θ-ins-closed language in Σ^+ cannot be minimal.

Proposition 8. *There is no minimal $\star$-k-θ-ins-closed language in Σ^+.*

Proof. Suppose that $L \subseteq \Sigma^+$ is a *minimal* $\star$-k-θ-ins-closed language. Let $w \in L$ with minimal length $m = |w|$ and let $L' = L \setminus \{w\}$. The language L' is not $\star$-k-θ-ins-closed. Therefore there exists $u = u_1u_2 \in L'$, $v \in L'(|u_2| \leq k$ if $\star =$ right and $|u_1| \leq k$ if $\star =$ left) such that $u_1vu_2 \notin \theta(L')$. However since $L' \subseteq L$ and L is $\star$-k-θ-ins-closed we have that $u_1vu_2 \in \theta(L)$. Therefore $u_1vu_2 = \theta(w)$ which implies that $|w| > |u|$ a contradiction.

Recall that a language $L \subseteq \Sigma^*$ is called right-m-dense if for any $w \in \Sigma^*$, there exists $x \in \Sigma^*$, $|x| \leq m$ such that $wx \in L$. A right-m-dense and $\star$-k-θ-ins-closed language L is said to be minimal if it does not properly contain any right-m-dense and $\star$-k-θ-ins-closed language. It has been shown in [11] that every right-m-dense and k-ins-closed language L contains a minimal right-m-dense and k-ins-closed language. The result also holds true for $\star$-k-θ-ins-closed languages. □

Proposition 9. *Every right-m-dense and ⋆-k-θ-closed language, L contains a minimal right-m-dense and ⋆-k-θ-ins-closed language.*

The proof of the above proposition is similar to the one proved in [11] and hence we omit it here.

5. The k-θ-deletion set of languages

Given two words $u, v \in \Sigma^*$, the deletion of v in to u is defined as $u \rightarrow v = \{u_1 u_2 : u = u_1 v u_2\}$. The notion of k-deletion was introduced in [9] under the name of k-quotient. The operation of k-deletion restricts the generality of deletion by allowing words to be deleted only from at most $k+1$ positions. The right and left k-deletions of v from u are defined (See [11]) respectively by:

$$u \rightarrow_r^k v = \{u_1 u_2 : u = u_1 v u_2, |u_2| \leq k, u_1, v, u_2 \in \Sigma^*\},$$
$$u \rightarrow_l^k v = \{u_1 u_2 : u = u_1 v u_2, |u_1| \leq k, u_1, v, u_2 \in \Sigma^*\}.$$

If $k = 0$, the right-k-deletion and the left-k-deletion become the well known right and left quotient respectively. The left and the right-k-deletion of a language L_2 from L_1 can be defined in a natural fashion. The right-k-deletion was initially called k-deletion and several of its properties were studied in [9]. Similar results can be obtained for the left-k-deletion operation and we omit them here.

We use instead both of these concepts and the notion of an involution function to define the left-k-θ-deletion set and right-k-θ-deletion set of a given language.

Let $L \subseteq \Sigma^*$ and let right-k-Sub$(L) = \{u \in \Sigma^* : xuy \in L, |y| \leq k\}$ and left-k-Sub$(L) = \{u \in \Sigma^* : xuy \in L, |x| \leq k\}$. The elements of left(right)-k-Sub(L) are called the left(right)-k-subwords. To the language L, one can associate a language $\star$-k-θ-del(L) consisting of all the words with the following property: x is a $\star$-k-subword of at least one of the word of $\theta(L)$, and the $\star$-k-deletion of x from any word of $\theta(L)$ containing x as a $\star$-k-subword yields a word belonging to L. Formally the $\star$-k-deletion set of a language L is defined as,

$$\star\text{-}k\text{-}\theta\text{-del}(L)$$
$$= \{x \in \star\text{-}k\text{-Sub}(\theta(L)) : \forall\, u \in \theta(L), u = u_1 x u_2, |u_i| \leq k, u_1 u_2 \in L\}.$$

Note that when $\star$ = right, $i = 2$ and when $\star$ = left, $i = 1$.

The next results show how the k-θ-deletion set of a language can be constructed by using the k-dipolar deletion.

Proposition 10. $\star$-k-θ-$del(L) = (\theta(L) \rightleftharpoons^k_\star L^c)^c \cap \star$-$k$-$Sub(\theta(L))$.

Proof. Take $x \in \star$-k-θ-del(L). Then $x \in \star$-k-Sub(L) which implies for every $u \in \theta(L), u = u_1xu_2, u_1u_2 \in L$. Suppose, $x \in (\theta(L) \rightleftharpoons^k_\star L^c)$, then there exists $u \in \theta(L)$ such that $u = u_1xu_2$ with $u_1u_2 \in L^c$ which is a contradiction. Conversely let $x \in \star$-k-Sub$(\theta(L)) \cap (\theta(L) \rightleftharpoons L^c)^c$. Suppose $x \notin \star$-k-θ-del(L) then there exists $u \in \theta(L)$ such that $u = u_1xu_2 \in \theta(L)$ and $u_1u_2 \notin L$ which implies $u_1u_2 \in L^c$ and hence $x \in \theta(L) \rightleftharpoons^k_\star L^c$ which is a contradiction. Therefore $x \in \star$-k-θ-del(L). □

Corollary 3. *If L is regular, then* $\star$-k-θ-$del(L)$ *is regular and can be effectively constructed.*

A language L is called $\star$-k-θ-del-closed if $v \in L$, $u_1vu_2 \in \theta(L)$ then $u_1u_2 \in L$. (Note that when $\star$ = left, then $|u_2| \leq k$ and when $\star$ = right, $|u_1| \leq k$).

Lemma 8. *Let* $L \subseteq \Sigma^\star$.

1. *When θ is morphic involution, then L is $\star$-k-θ-del-closed iff $\theta(L)$ is $\star$ k-θ-del-closed.*
2. *When θ is antimorphic involution, then L is left(right)-k-θ-del-closed iff $\theta(L)$ is right(left)-k-θ-del-closed.*

The following result provides a relation between k-θ-insertion closed and k-θ-deletion closed languages.

Proposition 11. *Let L be such that L is $\star$-k-θ-ins-closed. Then L is $\star$-k-θ-del-closed iff* $L = (\theta(L) \rightarrow^k_\star L)$.

Proof. Let L be $\star$-k-θ-del-closed. Let $x \in (\theta(L) \rightarrow^k_\star L)$. To show that $u \in L$. Since $u \in (\theta(L) \rightarrow^k_\star L)$, $u = u_1u_2$ such that $u_1xu_2 \in \theta(L)$ with $x \in L$. Since L is $\star$-k-θ-del-closed, $u_1u_2 \in L$ which implies $(\theta(L) \rightarrow^k_\star L) \subseteq L$. To prove the other inclusion, let $u \in L$ and since L is $\star$-k-θ-ins-closed, $u \in L \subseteq \star$-k-θ-ins(L) and $u = u_1u_2$ such that $u_1xu_2 \in \theta(L)$ which implies $u \in \theta(L) \rightarrow^k_\star L)$. Hence $L \subseteq (\theta(L) \rightarrow^k_\star L)$. Therefore $L = (\theta(L) \rightarrow^k_\star L)$. Conversely, let $L = \theta(1) \rightarrow^k_\star L)$. Let $v \in L$ with $u_1vu_2 \in \theta(L)$, then $u_1u_2 \in (\theta(L) \rightarrow^k_\star L) = L$ which implies $u_1u_2 \in L$ and hence L is $\star$-k-θ-del-closed. □

6. Conclusion

Formalizing the notion of DNA languages free of molecular hybridization with bulges led to the notion of k-involution prefix and k-involution suffix codes. We have investigated the theoretical properties of these codes in Section 3. We have also extended the notion of k-insertion set and k-deletion set of a language to incorporate the notion of an involution function. In Section 4 we have explored the connections between k-involution codes and k-insertion sets and have constructed these sets using the dual operation of dipolar k-deletion. As future work, we would like to investigate the algebraic characterizations of these involution codes through their syntactic monoid. The role of such codes in the design of DNA strands with certain properties (see [7, 8, 12]) also needs to be further investigated.

Acknowledgements. Research supported by NSERC and Canada Research Chair grants for Lila Kari.

References

[1] J. Borstel and D. Perrin, *Theory of Codes*, Academic Press, Inc., Orlando Florida, 1985.

[2] R. Deaton, J. Chen, H. Bi, M. Garzon, H. Rubin and D. F. Wood, A PCR based protocol for in vitro selection of non-crosshybridizing oligonucieotides, in *Proceedings of the 8th International Meeting on DNA Based Computers*, M. Hagiya and A. Ohuchi (editors) Springer, *LNCS*, Vol. 2568 (2003), pp. 196–204.

[3] R. Deaton, J. Chen, M. Garzon, J. Kim, D. Wood, H. Bi, D. Carpenter and Y. Wang, Characterization of non-crosshybridizing DNA oligonucleotides manufactured in vitro, in *Proceedings of the 10th International Meeting on DNA Computing*, C. Ferretti, G. Mauri and C. Zandron (editors), Springer, *LNCS*, Vol. 3384 (2004), pp. 50–61.

[4] R. Deaton *et al.*, A DNA based implementation of an evolutionary search for good encodings for DNA computation, in *Proceedings of the IEEE Conference on Evolutionary Computation* ICEC-97, (1997), pp. 267–271.

[5] D. Faulhammcr, A. R. Cukras, R. ,T. Lipton and L. F. Landweber, Molecular computation: RNA solutions to chess problems, in *Proceedings of the National Academy of Sciences*, USA, Vol. 97-4 (2000), pp. 1385–1389.

[6] M. Garzon, R. Deaton and D. Reanult, Virtual test tubes: a new methodology for computing, in *Proceedings 7th International Symposium on String Processing and Information Retrieval*, A. Corufia (editor), Spain, *IEEE Computing Society Press* (2000), pp. 116–121.

[7] S. Hussini, L. Kari and S. Konstantinidis, Coding properties of DNA languages, in *Proceedings of the 7th International Meeting on DNA Based Computers*, N. Jonoska and N.C. Seeman (editors), Springer, *LNCS*, Vol. 2340 (2002), pp. 57–69.

[8] N. Jonoska and K. Mahalingam, Involution codes: with application to DNA coded languages, *Natural Computing*, Vol. 4 (2) (2005), pp. 141-162.

[9] L. Kari and G. Thierrin, K-catenation and applications: k-prefix codes, *Journal of Information and Optimization Sciences*, Vol. 16 (2) (1995), pp. 263-276.

[10] L. Kari, S. Konstantinidis, E. Losseva, P. Sosik and G. Thierrin, Hairpin structures in DNA words, in *Proceedings of the 11th International Meeting on DNA Computing*, A. Carbonc, M. Dalcy, L. Kari, I. McQuilan and N. Pierce (editors), *LNCS*, Vol. 3892 (2005), pp. 158–170.

[11] L. Kari and G. Thierrin, K-insertion and K-deletion closure of languages, *Soochow Journal of Mathematics*, Vol. 21 (4) (1995), pp. 479-495.

[12] L. Kari and K. Mahalingam, DNA codes and their properties, in *Pre-proceedings of the 12th International Meeting on DNA Computing*, (C. Mao, T.Yokomori and B.T. Zhang (editors), (June 2006), pp. 238–249.

[13] L. Kari and K. Mahalingam, Insertion and deletion for involution codes, in *Proceedings of the 1st International Conference on Algebraic Informatics*, Thcssaloniki, Greece, October 20-23, (2005), pp. 207–218.

[14] L. Kari, S. Konstantinidis, E. Losseva and G. Wozniak, Sticky-free and overhang-free DNA languages, *Acta Informatica*, Vol. 40 (2003), pp. 119–157.

[15] L. Kari, S. Konstantinidis and P. Sosik, Bond-free languages: formalizations, maximality and construction methods, *International Journal of Foundations of Computer Science*, Vol. 16 (5) (2005), pp. 1039–1070.

[16] L. Kari, S. Konstantinidis and P. Sosik, Preventing Undesirable Bonds between DNA codewords, in *Proceedings of the 10th International Meeting on DNA Based Computers*, C. Ferretti, G. Mauri and C. Zandron (editors), *LNCS*, Vol. 3384 (2005), pp. 182–191.

[17] A. Marathe, A. E. Condon, R. M. Corn, On combinatorial word design *Journal of Computational Biology*, Vol. 8 (3) (2001), pp. 201-219.

[18] H. J. Shyr, *Free Monoids and Languages*, Hon-Min Book Company, Taichung, Taiwan, (1991).

Received September, 2006