Involution Solid Codes

Lila Kari and Kalpana Mahalingam

University of Western Ontario, Department of Computer Science,
London, ON N6A5B7, Canada
lila,kalpana@csd.uwo.ca

1 Introduction

DNA sequences consist of four nucleotide bases A, G, C, T (adenine, guanine, cytosine and thymine) and are joined together by phosphodiester bonds. A single strand of DNA, i.e. a chain of nucleotides, also has a "beginning" (usually denoted by $5'$) and an "end" (denoted by $3'$), and so the molecule is oriented. By the well-known Watson–Crick complementarity, A is complementary to T and C is complementary to G. The double-helix DNA strands are formed by a sequence and its complement binding together. The complementary strand is obtained by replacing the base nucleotide with its complement and reversing its direction.

Besides these "perfect" bonds, in practice certain strands can bind to others which are not their exact complements, hence rendering them useless for subsequent computation. Several attempts have been made to address this issue and many authors have proposed various solutions. A common approach has been to use the Hamming distance [1, 5, 6, 7, 26]. Experimental separation of strands with "good" sequences that avoid intermolecular cross-hybridization was reported in [3, 4].

In [11], Kari et al. introduce a theoretical approach to the problem of designing code words. Theoretical properties of languages that avoid certain undesirable hybridizations were discussed in [13, 18, 19, 25]. Based on these ideas and code-theoretic properties, a computer program for generating code words is being developed [12, 22]. Another algorithm based on backtracking, for generating such code words has also been developed by Li [24]. In [21] the authors have introduced a property of a language and showed that the properties discussed in [13, 19, 25] are its special cases. In [23] the author used the notion of partial words with holes for the design of DNA strands. In this chapter we follow the approach introduced in [11].

Every biomolecular protocol involving DNA or RNA generates molecules whose sequences of nucleotides form a language over the four-letter alphabet

$\Delta = \{A, G, C, T\}$. The Watson–Crick complementarity of the nucleotides defines a natural involution mapping θ, $A \mapsto T$ and $G \mapsto C$ which is an antimorphism of Δ^*. Undesirable Watson–Crick bonds (undesirable hybridizations) can be avoided if the language satisfies certain coding properties. In this paper we concentrate on θ-overlap free and θ-solid codes.

We start the chapter with definitions of coding properties that avoid intermolecular cross-hybridizations. The notions of θ-prefix and θ-suffix languages have been defined in [19] under the names of θ-p-compliant and θ-s-compliant, respectively. Here we consider sets of code words where the Watson–Crick complement of a word does not overlap with any other word. Hence, we have two additional coding properties that leads to the notion of θ-overlap-free code and θ-solid code. We make several observations about the closure properties of such languages. In particular, we concentrate on properties of languages that are preserved by union and concatenation. Also, we show that if a set of DNA strands has "good" coding properties that are preserved under concatenation, then the same properties will be preserved under arbitrary ligation of the strands. Section 3 investigates properties of θ-overlap-free codes. Section 4 investigates the properties of θ-solid codes.

2 Definitions

An alphabet Σ is a finite non-empty set of symbols. A word u over Σ is a finite sequence of symbols in Σ. We denote by Σ^* the set of all words over Σ, including the empty word 1 and, by Σ^+, the set of all nonempty words over Σ. We note that with the concatenation operation on words, Σ^* is the free monoid and Σ^+ is the free semigroup generated by Σ. For a word $w \in \Sigma^*$, the length of w is the number of non empty symbols in w and is denoted by $|w|$. Throughout the rest of the chapter, we concentrate on finite sets $X \subseteq \Sigma^*$ that are *codes*, i.e., every word in X^+ can be written uniquely as a product of words in X. For the background on codes we refer the reader to [2, 16, 27]. For a language $X \subseteq \Sigma^*$, let

$$\begin{aligned}
\mathrm{PPref}(X) &= \{u \mid \exists v \in \Sigma^+, uv \in X\} \\
\mathrm{PSuff}(X) &= \{u \mid \exists v \in \Sigma^+, vu \in X\} \\
\mathrm{PSub}(X) &= \{u \mid \exists v_1, v_2 \in \Sigma^*, v_1 v_2 \neq 1, v_1 u v_2 \in X\}.
\end{aligned}$$

We recall the definitions initiated in [11, 19] and used in [12, 18].

An involution $\theta : \Sigma \to \Sigma$ of a set Σ is a mapping such that θ^2 equals the identity mapping, $\theta(\theta(x)) = x$, $\forall x \in \Sigma$.

Definition 1. *Let $\theta : \Sigma^* \to \Sigma^*$ be a morphic or antimorphic involution and $X \subseteq \Sigma^+$.*

1. *The set X is called* θ-infix *if $\Sigma^*\theta(X)\Sigma^+ \cap X = \emptyset$ and $\Sigma^+\theta(X)\Sigma^* \cap X = \emptyset$.*
2. *The set X is called* θ-comma-free *if $X^2 \cap \Sigma^+\theta(X)\Sigma^+ = \emptyset$.*

3. *The set* X *is called* θ-prefix *if* $X \cap \theta(X)\Sigma^+ = \emptyset$.
4. *The set* X *is called* θ-suffix *if* $X \cap \Sigma^+\theta(X) = \emptyset$.
5. *The set* X *is called* θ-sticky-free *if for all* $w \in \Sigma^+$, $x, y \in \Sigma^*$, $wx, y\theta(w) \in X$ *then* $xy = 1$.
6. *The set* X *is called* θ-overhang-free *if for all* $w \in \Sigma^+$, $x, y \in \Sigma^*$, $wx, \theta(w)y \in X$ *or* $xw, y\theta(w) \in X$ *then* $xy = 1$.
7. *The set* X *is called* strictly θ *if* $X \cap \theta(X) = \emptyset$.

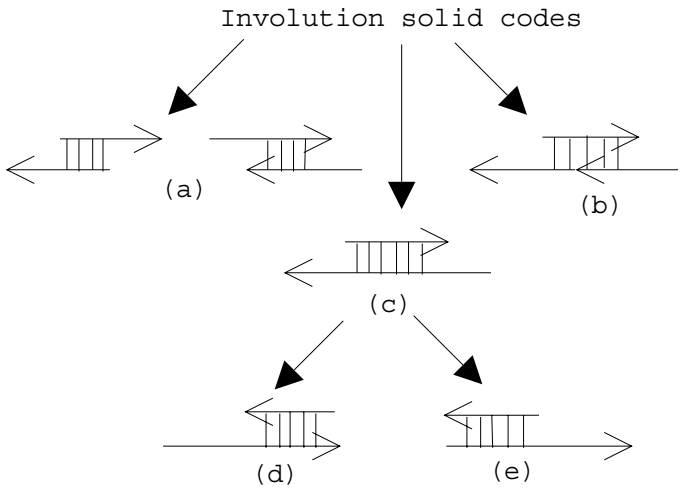

Fig. 1. Various types of intermolecular hybridizations. (**a**) (θ-overlap-free) The prefix or suffix of a code word is a suffix or prefix respectively of a complement of another code word; (**b**) (θ-comma-free) a code word is a reverse complement of a subword of a concatenation of two other code words; (**c**) (θ-infix) one code word is a reverse complement of a subword of another code word; (**d**) (θ-suffix) one code word is a reverse complement of a suffix of another code word; (**e**) (θ-prefix) one code word is a reverse complement of a prefix of another code word. The $3'$ end is indicated with an arrow.

Solid codes were introduced in [28] in the context of the study of disjunctive domains. Certain combinatorial and closure properties of solid codes were discussed in [15]. Properties of maximal solid codes were discussed in [17]. We now recall the definition of solid codes used in [17] which was defined using a characterization given in [15].

Definition 2. *A set* $X \subseteq \Sigma^+$ *is a* solid code *if*

1. X *is an infix code*
2. $\mathrm{PPref}(X) \cap \mathrm{PSuff}(X) = \emptyset$.

The notion of solid codes was extended to involution solid codes in [25]. Note that when the involution map denotes the Watson–Crick complement, the set of involution solid codes comprises of DNA strands that are overlap-free (see Fig. 1).

Definition 3. *Let $X \subseteq \Sigma^+$.*

1. *The set X is called* θ-overlap-free *if* $\mathrm{PPref}(X) \cap \mathrm{PSuff}(\theta(X)) = \emptyset$ *and* $\mathrm{PSuff}(X) \cap \mathrm{PPref}(\theta(X)) = \emptyset$.
2. *X is a* θ-solid code *if X is θ-infix and θ-overlap free.*
3. *X is a* maximal θ-solid code *iff for no word $u \in \Sigma^+ \setminus X$, the language $X \cup \{x\}$ is a θ-solid code.*

Throughout the rest of the chapter we use θ to be either a morphic or antimorphic involution unless specified. Note that X is θ-overlap free (θ-solid) iff $\theta(X)$ is θ-overlap free (θ-solid).

3 Properties of Involution Overlap-Free Codes

In this section we discuss the properties of the class of involution overlap-free codes. We also discuss the relation between the overlap-free codes and some of the previously defined codes (see Definition 1).

Proposition 1. *Let θ be an antimorphic involution. If X is θ-overhang-free then X is θ-overlap-free.*

Proof. Let X be θ-overhang-free. To show that X is θ-overlap free, let us suppose there exists $xw \in X$ and $wy \in \theta(X)$. Then $\theta(y)\theta(w) \in X$ which is a contradiction to our assumption that X is θ-overhang-free. The case when $wx \in X$ and $yw \in \theta(X)$ also result in a contradiction.

Proposition 2. *If X is a strictly θ-solid code then X^+ is θ-overlap free.*

Proof. We need to show that $\mathrm{PPref}(X^+) \cap \mathrm{PSuff}(\theta(X^+)) = \emptyset$ and $\mathrm{PSuff}(X^+) \cap \mathrm{PPref}(\theta(X^+)) = \emptyset$. Suppose X^+ is not θ-overlap-free. Then there exists $x \in \mathrm{PPref}(X^+) \cap \mathrm{PSuff}(\theta(X^+))$ such that $x = x_1x_2 \dots x_ia_1 = \theta(a_2)\theta(y_1)...\theta(y_j)$ for $x_i, y_j \in X$ for all i, j. Then either x_i is a subword of $\theta(y_j)$ which is a contradiction to our assumption that X is θ-infix, or $a_1 \in \mathrm{PSuff}(\theta(y_j))$ which is again a contradiction. Hence $\mathrm{PPref}(X^+) \cap \mathrm{PSuff}(\theta(X^+)) = \emptyset$. Similarly we can show that $\mathrm{PSuff}(X^+) \cap \mathrm{PPref}(\theta(X^+)) = \emptyset$.

Corollary 1. *Let $X, Y \subseteq \Sigma^+$ be such that $X \cup Y$ is strictly θ-solid. Then XY is θ-overlap-free.*

Proposition 3. *Let X be such that X^n is θ-overlap-free for some $n \geq 1$. Then X^i, $1 \leq i \leq n$, is also θ-overlap-free.*

Proof. Suppose not. Then there exists $ax \in X^i$, $ya \in \theta(X^i)$ for some $1 \leq i \leq n$. Let $r \in X^j$ such that $i + j = n$. Then $axr \in X^n$ and $\theta(r)ya \in \theta(X^n)$ which implies $a \in \mathrm{PPref}(X^n) \cap \mathrm{PSuff}(\theta(X^n))$ which is a contradiction. Similarly we can show $\mathrm{PPref}(\theta(X^n)) \cap \mathrm{PSuff}(X^n) = \emptyset$.

4 Properties of Involution Solid Codes

In this section we discuss the properties of the class of involution solid codes. It turns out that involution solid codes are closed under a restricted kind of product, arbitrary intersections and catenation closure while not closed under union, complement, product and homomorphisms. The first two properties are immediate consequences of the definitions.

Proposition 4. *The class of θ-solid codes is closed under arbitrary intersection.*

Proposition 5. *The class of θ-solid codes is not closed under union, complement, product and homomorphism.*

Proof. Consider the θ-solid codes $\{a\}$ and $\{ab\}$ over the alphabet set $\Sigma = \{a, b\}$ and with θ being an antimorphic involution that maps $a \mapsto b$ and $b \mapsto a$. The sets $\{a, ab\} = \{a\} \cup \{ab\}$ and $\{aba\} = \{ab\}\{a\}$ are not θ-solid. This proves the statement for union and concatenation. Let $h : \Sigma^* \mapsto \Sigma^*$ be a homomorphism such that $h(a) = aba$ and $h(b) = bab$. Note that $\{a\}$ is θ-solid but $h(a) = aba$ is not θ-solid.

Note that for $X \subseteq \Sigma^+$ and θ a morphic or antimorphic involution, X is θ-solid code iff $\theta(X)$ is θ-solid code.

Proposition 6. *If X is a θ-solid code then X is strictly θ-comma-free.*

Proof. Note that since $\mathrm{PPref}(X) \cap \mathrm{PSuff}(\theta(X)) = \emptyset$, X is strictly θ. Suppose X is not θ-comma-free. Then there are $x, y, z \in X$ such that $xy = a\theta(z)b$, $a, b \in \Sigma^+$. Then either $\theta(z)$ is a subword of x or a subword of y which contradicts that X is θ-infix, or $\theta(z) = z_1 z_2$ such that $az_1 = x$ and $z_2 b = y$ which implies $z_1 \in \mathrm{PPref}(\theta(X)) \cap \mathrm{PSuff}(X)$ and $z_2 \in \mathrm{PPref}(X) \cap \mathrm{PSuff}(\theta(X))$ which contradicts our assumption that X is θ-overlap-free and hence X is a θ-solid code.

Note that the converse of the above proposition holds when θ is the identity (see [10]) but not for any general θ. For example let $X = \{aa, baa\}$ and for an antimorphic $\theta : a \to b, b \to a$, $\theta(X) = \{bb, bba\}$. It is easy to check that X is θ-comma-free. But $ba \in \mathrm{PPref}(X) \cap \mathrm{PSuff}(\theta(X))$ which contradicts condition 2 of Definition 3.

Proposition 7. *Let $X, Y \subseteq \Sigma^+$ be such that X and Y are strictly θ and $X \cap \theta(Y) = \emptyset$. If $X \cup Y$ is θ-solid then XY is θ-solid.*

Proof. Suppose XY is not θ-infix. Then there exists $x_1, x_2 \in X$ and $y_1, y_2 \in Y$ such that $x_1 y_1 = a\theta(x_2 y_2)b$ for some $a, b \in \Sigma^*$ not both empty. When θ is morphic, $x_1 y_1 = a\theta(x_2)\theta(y_2)b$. Then either $\theta(x_2)$ is a subword of x_1 or $\theta(y_2)$ is a subword of y_1 which is a contradiction with $X \cup Y$ is θ-infix. A similar contradiction arises when θ is antimorphic. Suppose $\mathrm{PPref}(XY) \cap \mathrm{PSuff}(\theta(XY)) \neq \emptyset$. Then there exists $p \in \mathrm{PPref}(XY)$ and $\theta(q) \in \mathrm{PSuff}(\theta(XY))$ such that $p = \theta(q)$. Then the following cases arise:

1. $p \in \text{PPref}(X)$ and $\theta(q) \in \text{PSuff}(\theta(Y))$ or $\theta(q) \in \text{PSuff}(\theta(X))$
2. $p \in \text{PPref}(X)$ and $\theta(q) \in \text{PSuff}(\theta(XY))$
3. $p \in X$ and $\theta(q) \in \theta(X)$ or $\theta(q) \in \theta(Y)$.

The first two cases contradict our assumption that $X \cup Y$ is θ-solid and the third case contradicts our assumption that X is strictly θ or $X \cap \theta(Y) = \emptyset$. Similarly we can show that $\text{PSuff}(XY) \cap \text{PPref}(\theta(XY)) = \emptyset$.

Corollary 2. *If X is a strictly θ-solid code then X^n is a θ-solid code.*

Proposition 8. *The code X is a strictly θ-solid code iff X^+ is a strictly θ-solid code.*

Proof. X is a θ-solid code and hence X is strictly θ-comma-free which implies X^+ is θ-infix (see Proposition 3.3 in [14]). From Proposition 2, X^+ is θ-overlap-free and hence X^+ is θ-solid. The converse is immediate.

Proposition 9. *Let X be a regular language. It is decidable whether or not X is a θ-solid code.*

Proof. It has been proved in [11] that it is decidable whether X is θ-infix or not. The sets $\text{PPref}(X)$ and $\text{PSuff}(X)$ are known to be regular for regular X and also $\theta(X)$ is also regular when X is regular. Hence $\text{PPref}(\theta(X))$ and $\text{PSuff}(\theta(X))$ are also regular. In order to decide whether X is θ-solid one needs to decide whether the intersection $\text{PPref}(\theta(X)) \cap \text{PSuff}(X) = \emptyset$ and $\text{PSuff}(\theta(X)) \cap \text{PPref}(X) = \emptyset$ which is decidable for regular X.

Proposition 10. *Let θ be a morphic or antimorphic involution and $X \subseteq \Sigma^+$ be a strictly θ-solid code. Then $Y = \{u_1 v u_2 : u_1 u_2, v \in X, u_1, u_2 \in \Sigma^*\}$ is a θ-solid code.*

Proof. Given X is θ-solid, we need to show that Y is θ-infix and $\text{PPref}(Y) \cap \text{PSuff}(\theta(Y)) = \emptyset$ and $\text{PPref}(\theta(Y)) \cap \text{PSuff}(Y) = \emptyset$. Suppose Y is not θ-infix, then there exists $p, q \in Y$ such that $p = u_1 x_1 v_1$, $q = u_2 x_2 v_2$ and $p = a\theta(q)b$ for some $a, b \in \Sigma^*$, $u_1 v_1, u_2 v_2, x_1, x_2 \in X$. Hence when θ is a morphic involution we have $u_1 x_1 v_1 = a\theta(u_2)\theta(x_2)\theta(v_2)b$. Then either $\theta(x_2)$ is a subword of u_1 or v_1, or $\theta(x_2)$ is a subword of $u_1 x_1$ or $x_1 v_1$. Both cases contradict our assumption that X is θ-solid. Similarly we can prove when θ is antimorphic involution. Suppose there exists $a \in \text{PPref}(Y) \cap \text{PSuff}(\theta(Y))$. Then $a \in \text{PPref}(u_1 x_1 v_1)$ and $a \in \text{PSuff}(\theta(u_2 x_2 v_2))$ for some $u_1 x_1 v_1, u_2 x_2 v_2 \in Y$. There are several cases that we need to consider which eventually boil down to one of three below. We show when θ is an antimorphic involution and the case when θ is morphism can be proved similarly. We have $a \in \text{PPref}(u_1 x_1 v_1)$ and $a \in \text{PSuff}(\theta(v_2)\theta(x_2)\theta(v_2))$. Then either $a \in \text{PPref}(u_1)$ and $\text{PSuff}(\theta(u_2))$ or $a \in \text{PPref}(u_1)$ and $a \in \text{PSuff}(\theta(x_2)\theta(u_2))$ or $a \in \text{PPref}(u_1)$ and $a \in \text{PSuff}(\theta(v_2)\theta(x_2)\theta(u_2))$. The first two cases contradict $\text{PPref}(X) \cap \text{PSuff}(\theta(X)) = \emptyset$ and the third case contradicts X being θ-infix. Similarly we can show $\text{PSuff}(Y) \cap \text{PPref}(\theta(Y)) = \emptyset$. Therefore Y is θ-solid code.

The next proposition provides a general method for constructing certain maximal θ-solid codes.

Proposition 11. *Let θ be an antimorphic involution. Let $\Sigma = A \cup B \cup C$ such that A, B, C are disjoint sets such that A and C are strictly θ and $A \cap \theta(B) = \emptyset$ and $C \cap \theta(B) = \emptyset$. Then $X = AB^*C$ is a maximal θ-solid code.*

Proof. First we show that X is θ-solid. Suppose X is not θ-solid. Then either X is not θ-infix or X is not θ-overlap-free. Suppose X is not θ-infix. Then there exists $a_1b_1...b_nc_1, a_2d_1...d_kc_2 \in AB^*C$ with $a_1, a_2 \in A$, $c_1, c_2 \in C$ and $b_1...b_n, d_1...d_k \in B^*$ such that $a_1b_1...b_nc_1 = p\theta(a_2d_1...d_kc_2)q$ and hence $a_1b_1...b_nc_1 = p\theta(c_2)\theta(d_k)...\theta(d_1)\theta(a_2)q$ for some $p, q \in \Sigma^*$ not both empty. If $q \in \Sigma^+$ then $\theta(a_2) = b_i$ for some i and if $p \in \Sigma^+$ then $\theta(c_2) = b_j$ for some j. Both cases contradict our assumption that $A \cap \theta(B) = \emptyset$ and $C \cap \theta(B) = \emptyset$. Suppose $x \in \mathrm{PPref}(X) \cap \mathrm{PSuff}(\theta(X))$. Then either $x = a_1b_1 \dots b_i = \theta(d_i) \dots \theta(d_1)\theta(a_2)$ for some $a_1, a_2 \in A$ and $b_1 \dots b_i, d_1 \dots d_i \in B$ which contradicts our assumption that $A \cap \theta(B) = \emptyset$ or $x = a_1 = \theta(a_2)$ which contradicts our assumption that A is strictly θ. Hence $\mathrm{PPref}(X) \cap \mathrm{PSuff}(\theta(X)) = \emptyset$. Similarly we can show that $\mathrm{PPref}(\theta(X)) \cap \mathrm{PSuff}(X) = \emptyset$. Hence X is a θ-solid code.

To show that $X = AB^*C$ is maximal. Consider a word $w \in \Sigma^*$ such that $w \notin X$ where $w = x_1x_2...x_n$ with $x_i \in \Sigma$ for $i = 1, \dots, n$. We show that $X \cup \{w\}$ is not θ-solid. Assume there is an index i with $x_i \in \theta(C)$ and in fact let i be the minimal with this property. If $i = n$ then $x_i \in \mathrm{PSuff}(w) \cap \mathrm{PPref}(\theta(v))$ for some $v \in X$. Hence $X \cup \{w\}$ is not θ-solid. If $i < n$ then $x_{i+1}, ..., x_n \in \theta(B) \cup \theta(A)$. If $x_{i+1} \in \theta(A)$ then $x_ix_{i+1} \in \theta(X) \cap \mathrm{Sub}(w)$ and $X \cup \{w\}$ is not θ-solid. Therefore assume that $x_{i+1} \in \theta(B)$. If $x_{i+1}...x_n \in \theta(B^+)$ then $x_ix_{i+1}...x_n \in \mathrm{PSuff}(w) \cap \mathrm{PPref}(\theta(v))$ for some $v \in X$ and $X \cup \{w\}$ is not θ-solid. Thus, there is an index j with $i + 1 < j \leq n$ and $x_j \in \theta(A)$. Choose j minimal with these properties. Then $x_ix_{i+1}...x_j \in \theta(X) \cap \mathrm{Sub}(w)$, hence $X \cup \{w\}$ is not θ-solid. So far we have proved that w cannot contain a symbol from $\theta(C)$ if $X \cup \{w\}$ is to be θ-solid. Similarly we can show that w cannot contain a symbol from $\theta(A)$. Hence $w \in \theta(B^*)$ (i.e.) $w \in \mathrm{Sub}(\theta(v))$ for some $v \in X$ which again contradicts our assumption that $X \cup \{w\}$ is θ-solid. Hence X is a maximal θ-solid code.

From the above definitions and propositions we have deduced the following.

Lemma 1. *Let θ be an antimorphic involution.*

1. *Let $\Sigma_1, ..., \Sigma_n$ be a partition of Σ such that Σ_i is strictly θ for all i. Then every language $\Sigma_i\Sigma_j$ is θ-solid.*
2. *If Σ_1, Σ_2 is a partition of Σ such that Σ_i is strictly θ for $i = 1, 2$, then $\Sigma_1\Sigma_2$ is maximal θ-solid code.*
3. *Let $A \subseteq \Sigma$ be such that $A = \theta(A)$ and $X \subseteq A^+$. Then X is a maximal θ-solid code over A if and only if $X \cup (\Sigma \setminus A)$ is a maximal θ-solid code over Σ.*

4. Let $B \subseteq \Sigma$ such that $B \cap \theta(B) = \emptyset$. Then $X = B^+$ is a θ-solid code.

The next proposition provides us with conditions so that the involution solid codes are preserved under a morphic or antimorphic mapping.

Proposition 12. *Let Σ_1 and Σ_2 be finite alphabet sets and let f be an injective morphism or antimorphism from $\Sigma_1 \mapsto \Sigma_2^*$. Let X be a code over Σ_1^*. Then $f(X)$ is a code over Σ_2^*. Let $\theta_1 : \Sigma_1^* \mapsto \Sigma_1^*$ and $\theta_2 : \Sigma_2^* \mapsto \Sigma_2^*$ be both morphic or antimorphic involutions such that $f(\theta_1(x)) = \theta_2(f(x))$ for all $x \in X$. Let $P = Pref(\theta_2(f(X)))$ and $S = Suff(\theta_2(f(X)))$. Let $(A^+P \cap SA^+) \cap f(\Sigma_1^+) = \emptyset$ and $A^+PA^+ \cap f(\Sigma_1) = \emptyset$ where $A = \Sigma_2^* \setminus f(\Sigma_1^*)$. If X is θ_1-solid then $f(X)$ is θ_2-solid.*

Proof. Let X be a θ_1-solid code. Note that $f(X)$ is θ_2-infix [14]. We need to show that $\mathrm{PPref}(f(X)) \cap \mathrm{PSuff}(\theta_2(f(X))) = \emptyset$ as well as $\mathrm{PSuff}(f(X)) \cap \mathrm{PPref}(\theta_2(f(X))) = \emptyset$ hold. Let θ_1 and θ_2 be morphic involutions and let f be an injective antimorphism. Suppose there exists $a \in \mathrm{PPref}(f(x_1x_2))$ and $a \in \mathrm{PSuff}(\theta_2(f(y_1y_2))$ for some $x_1x_2, y_1y_2 \in X$. Note that $f(x_1x_2) = f(x_2)f(x_1)$ and $\theta_2(f(y_1y_2)) = f(\theta_1(y_1y_2)) = f(\theta_1(y_1)\theta_1(y_2)) = f(\theta_1(y_2))f(\theta_1(y_1))$. Hence if $a = f(x_2) = f(\theta_1(y_1))$ then $x_2 = \theta_1(y_1)$ since f is injective which is a contradiction to $\mathrm{PPref}(X) \cap \mathrm{PSuff}(\theta_1(X)) = \emptyset$. The other case can be proved similarly. Hence $f(X)$ is θ_2-solid.

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

References

1. E.B. Baum, DNA Sequences useful for computation, unpublished article, available at: http://www.neci.nj.nec.com/homepages/eric/seq.ps (1996).
2. J. Berstel, D. Perrin, Theory of Codes, Academic Press, Inc., Orlando, Florida, 1985.
3. R. Deaton, J. Chen, H. Bi, M. Garzon, H. Rubin, D.H. Wood, A PCR based protocol for n vitro selection of non-crosshybridizing oligonucleotides, *DNA Computing: Proceedings of the 8th International Meeting on DNA Based Computers* (M. Hagiya, A. Ohuchi editors), Springer LNCS **2568** (2003): 196–204.
4. R. Deaton, J. Chen, M. Garzon, J. Kim, D. Wood, H. Bi, D. Carpenter, Y. Wang, Characterization of non-crosshybridizing DNA oligonucleotides manufactured in vitro, *DNA computing: Preliminary Proceedings of the 10th International Meeting on DNA Based Computers* (C. Ferretti, G. Mauri, C. Zandron editors) Springer LNCS **3384**, (2005): 50–61.
5. R. Deaton et al, A DNA based implementation of an evolutionary search for good encodings for DNA computation, *Proc. IEEE Conference on Evolutionary Computation* ICEC-97, (1997): 267–271.

6. D. Faulhammer, A.R. Cukras, R.J. Lipton, L.F.Landweber, Molecular Computation: RNA solutions to chess problems, *Proceedings of the National Academy of Sciences*, USA, **97** 4 (2000): 1385–1389.
7. M. Garzon, R. Deaton, D. Renault, Virtual test tubes: a new methodology for computing, *Proc. 7th. Int. Symposium on String Processing and Information retrieval*, A Coruña, Spain. IEEE Computing Society Press (2000): 116–121.
8. T. Head, Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors, *Bull. Math. Biology* **49** (1987): 737–759.
9. T. Head, Gh. Paun, D. Pixton, Language theory and molecular genetics, in *Handbook of Formal Languages, Vol.II* (G. Rozenberg, A. Salomaa editors) Springer-Verlag (1997): 295–358.
10. T. Head, Relativized code concepts and multi-tube DNA dictionaries, in *Finite vs Infinite*, C.S. Calude and Gh. Paun editors, Springer-Verlag (2000): 175–186.
11. S. Hussini, L. Kari, S. Konstantinidis, *Coding properties of DNA languages*, *DNA Computing: Proceedings of the 7th International Meeting on DNA Based Computers* (N. Jonoska, N.C. Seeman editors), Springer LNCS **2340** (2002): 57–69.
12. N. Jonoska, D. Kephart, K. Mahalingam, Generating DNA code words, *Congressus Numerantium* **156** (2002): 99–110.
13. N. Jonoska, K. Mahalingam, Languages of DNA based code words *Proceedings of the 9th International Meeting on DNA Based Computers*, J. Chen, J. Reif editors, Springer LNCS **2943** (2004): 61–73.
14. N. Jonoska, K. Mahalingam, J. Chen, Involution codes: with application to DNA coded languages, *Natural Computing* **4** 2 (2005): 141–162.
15. H. Jürgensen, S.S. Yu, Solid codes, *Journal of Information Processing and Cybernetics*, **EIK 26** (10) (1990): 563–574.
16. H. Jürgensen, S. Konstantinidis, Codes, *Handbook of Formal Languages*, Vol 1, Chapter 8, G. Rozenberg and A. Salomaa editors, Springer-Verlag (1997).
17. H. Jürgensen, M. Katsura and S. Konstantinidis, Maximal solid codes, *Journal of Automata, Languages and Combinatorics* **6**(1) (2001): 25–50.
18. L. Kari, K. Mahalingam, More on involution codes, Preprint.
19. L. Kari, S. Konstantinidis, E. Losseva, G. Wozniak, Sticky-free and overhang-free DNA languages, *Acta Informatica* **40** (2003): 119-157.
20. L. Kari, S. Konstantinidis, P. Sosik, Bond-free languages: formalizations, maximality and construction methods, in *DNA Computng*, Ferretti et al. (eds.), Springer LNCS **3384** (2004): 169-181.
21. L. Kari, S. Konstantinidis, P. Sosik, Preventing undesirable bonds between DNA codewords, in *DNA Computng*, Ferretti et al. (eds.), Springer LNCS **3384** (2004): 182-191.
22. D. Kephart, J. Lefevre, Codegen: The generation and testing of DNA code words, *Proceedings of IEEE Congress on Evolutionary Computation*, (2004): 1865–1873.
23. P. Leupold, *Partial words for DNA coding*, in *DNA Computng*, Ferretti et al. (eds.), Springer LNCS **3384** (2005): 224–234.
24. Z. Li, *Construct DNA code words using backtrack algorithm*, preprint.
25. K. Mahalingam, *Involution Codes: With Application to DNA Strand Design* Ph.d. Thesis, University of South Florida, Tampa, FL, 2004.
26. A. Marathe, A.E. Condon, R.M. Corn, On combinatorial DNA word design, *DNA Based Computers V*, E. Winfree, D.K. Gifford eds., Providence, RI, DIMACS, American Mathematical Society, (1999): 75–90.

27. H.J. Shyr, *Free Monoids and Languages*, Hon Min Book Company 2001.
28. H.J. Shyr, S.S. Yu, Solid codes and disjunctive domains, *Semigroup Forum*, **41** (1990): 23–37.