Involution Solid and Join Codes

Nataša Jonoska[1], Lila Kari[2], and Kalpana Mahalingam[2]

[1] University of South Florida, Department of Mathematics, Tampa, FL-33620
jonoska@math.usf.edu
[2] University of Western Ontario, Department of Computer Science,
London, Ontario, Canada-N6A 5B7
{lila, kalpana}@csd.uwo.ca

Abstract. In this paper we study a generalization of the classical notions of solid codes and comma-free codes: involution solid codes (θ-solid) and involution join codes (θ-join). These notions are motivated by DNA strand design where Watson-Crick complementarity can be formalized as an antimorphic involution. We investigate closure properties of these codes, as well as necessary conditions for θ-solid codes to be maximal. We show how the concept of θ-join can be utilized such that codes that are not themselves θ-comma free can be split into a union of subcodes that are θ-comma free.

1 Introduction

When using single stranded DNA molecules in DNA nanotechnology and DNA computing it is important to minimize the errors that are due to unwanted cross-hybridization. Such errors occur if two different bits of information are encoded as single stranded DNA molecules that are totally or partially complementary. This complementarity induces unintentional hybridizations and such encodings should be avoided (See Fig. 1).

Several attempts have been made to address this issue and many authors have proposed various solutions. Such approaches to the design of DNA encodings without undesirable bonds and secondary structures were summarized in [20] and [24]. For more details we refer the reader to [1, 3, 4, 5, 6, 8, 23]. One approach to this issue of "good encodings" is theoretical study of the algebraic and code-theoretic properties of DNA encodings through formal language theory. In [11], Kari et al. introduced such theoretical approach to the problem of designing code words. Properties of languages that avoid certain undesirable bonds were discussed in several follow-up papers [13, 18, 19, 21, 22].

This paper follows the approach introduced in [11] and investigate the formal language and coding theoretic notions inspired and motivated by DNA encoded information. In order to model the characteristics of the biologically encoded information, we replace the identity function by a composition of the complement function with the mirror-image function (such a function is a correct mathematical model of the Watson-Crick complement of DNA strands) or, more generally, by an arbitrary involution (a function θ with the property that θ^2 equals identity)

O.H. Ibarra and Z. Dang (Eds.): DLT 2006, LNCS 4036, pp. 192–202, 2006.

[20]. Moreover, as needed, we replace regular homomorphisms by antimorphic functions, to allow for the property of reversal in complementary DNA strands. These lead to natural, as well as theoretically elegant, generalizations of classical notions such as prefix codes, suffix codes, infix codes, bifix codes, intercodes, comma-free codes, etc [13, 18, 19, 22]. These are but some examples of ways in which classical notions and results in formal language theory and algebraic informatics can be meaningfully generalized. This paper continues this line of research by introducing and studying the notions of θ-solid and θ-join codes. If θ is the identity function, the θ-solid codes and θ-join codes respectively become the well known solid and join codes respectively.

Section 2 includes the definitions and we introduce the new concept of θ-solid codes and include some properties of θ-overlap-free codes. Section 3 contains the closure properties of θ-solid codes. Note that the results obtained for θ-solid codes hold true for solid codes when θ is the identity function. We show that the property of being a θ-solid code is decidable for regular languages and provide results about maximal θ-solid codes. In Section 4 we generalize the concept of join codes to θ-join codes and develop a method to extract a sequence of subsets which are θ-comma-free from a set that is not θ-comma-free. Due to space restrictions, some of the proofs are omitted.

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 non-empty words over 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 paper, we concentrate on sets $X \subseteq \Sigma^+$ that are *codes* such that every word in X^+ can be written uniquely as a product of words in X, or equivalently, X^+ is a free semigroup generated by X. For the background on codes we refer the reader to [2, 26]. For a language $X \subseteq \Sigma^*$, let

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

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

An involution $\theta : \Sigma \to \Sigma$ of a set Σ is a mapping such that θ^2 equals the identity mapping.

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

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

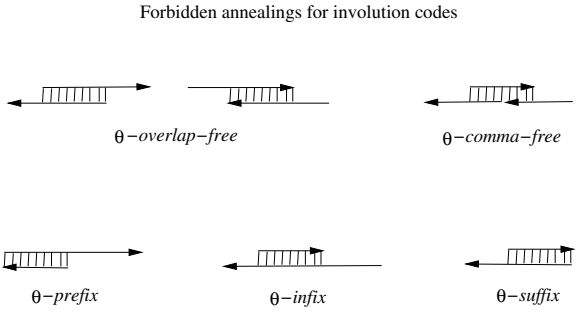


Fig. 1. Schematic representation for forbidden inter molecular DNA hybridizations. The $3'$ ends are indicated with an arrow.

Note that when θ is identity θ-infix code and θ-comma-free code are just infix and comma-free codes [2, 26]. In [12, 13] θ-strict codes are called strictly θ and in [18] they are called θ-non overlapping.

Solid codes were introduced in [25]. Certain combinatorial and closure properties of solid codes were discussed in [15]. We recall the definition of solid codes used in [17] defined by 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 [22]. Note that when the involution map denotes the Watson-Crick complement, the set of involution-solid codes comprises of DNA strands that do not overlap with the complement of any other DNA strand (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 $X \cup \{u\}$ is a θ-solid code.*

Note 1. 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.

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

Proposition 1. *If X is a θ-strict-solid code then X^+ is θ-overlap free.*

Proof. We need to show that $\text{PPref}(X^+) \cap \text{PSuff}(\theta(X^+)) = \emptyset$ and $\text{PSuff}(X^+) \cap \text{PPref}(\theta(X^+)) = \emptyset$. Suppose there exists $x \in \text{PPref}(X^+) \cap \text{PSuff}(\theta(X^+))$ such that $x = x_1x_2..x_ia = \theta(b)\theta(y_1)...\theta(y_j)$ for $x_i, y_j \in X$ for all i, j and $a \in \text{PPref}(X)$ and $\theta(b) \in \text{PSuff}(\theta(X))$. Then x_1 is not a subword of $\theta(b)$ as this contradicts the assumption that X is θ-infix. Also, $\theta(b)$ being in $\text{PPref}(x_1))$ contradicts the assumption that X is θ-solid. Hence $\text{PPref}(X^+) \cap \text{PSuff}(\theta(X^+)) = \emptyset$. Similarly, $\text{PSuff}(X^+) \cap \text{PPref}(\theta(X^+)) = \emptyset$. □

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

Proposition 2. *Let X be a regular language. It is decidable whether or not X is θ-overlap free.*

3 Properties of Involution Solid Codes

In this section we consider the closure 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 3. *The class of θ-solid codes is closed under arbitrary intersection and θ-solid codes is not closed under union, complement, concatenation and homomorphism.*

Example 1. 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. Let $h : \Sigma^* \mapsto \Sigma^*$ be homomorphism such that $h(a) = aba$ and $h(b) = bab$. Note that $\{a\}$ is θ-solid but $h(a) = aba$ is not θ-solid.

Proposition 4. *If X is a θ-solid code then X is a θ-comma-free code.*

Proof. According to proposition 3.5 in [14], X is θ-comma-free if and only if X is θ-infix and $X_{ip}X_{is} \cap \theta(X) = \emptyset$ where $X_{ip} = \text{PPref}(\theta(X)) \cap \text{PSuff}(X)$ and $X_{is} = \text{PSuff}(\theta(X)) \cap \text{PPref}(X)$. Since X is θ-solid, X is θ-infix and $X_{ip} = X_{is} = \emptyset$ and hence $X_{ip}X_{is} \cap \theta(X) = \emptyset$. □

Note that the converse of the above proposition does not hold in general. For example let $X = \{aa, baa\}$ and let θ be an antimorphic involution such that $\theta : a \to b, b \to a$, then $\theta(X) = \{bb, bba\}$. It is easy to check that X is θ-comma-free. But $ba \in \text{PPref}(X) \cap \text{PSuff}(\theta(X))$ which contradicts condition 2 of definition 3.

Proposition 5. *Let $X, Y \subseteq \Sigma^+$ be such that X and Y are θ-strict 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 $x_1y_1 = a\theta(x_2y_2)b$ for some $a, b \in \Sigma^*$ not both empty. When θ is morphic, $x_1y_1 = a\theta(x_2)\theta(y_2)b$. 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 $X \cup Y$ being θ-infix. Similar contradiction arises when θ is antimorphic. From Corollary 1, XY is θ-overlap free and hence XY is θ-solid. □

Corollary 2. *X is a θ-strict-solid code if and only if X^+ is a θ-strict-solid code.*

Proposition 6. *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. From Proposition 2 it is decidable whether X is θ-overlap free or not. Hence for a regular X, one can decide whether X is θ-solid or not.

The following proposition gives a method for constructing θ-strict-solid codes.

Proposition 7. *Let θ be a morphic or antimorphic involution and $X \subseteq \Sigma^+$ be θ-strict-solid code. Then $Y = \{u_1vu_2 : u_1u_2, v \in X, u_1, u_2 \in \Sigma^*\}$ is a θ-solid code.*

The next proposition provides a general method for constructing not just θ-solid codes, but maximal θ-solid codes.

Proposition 8. *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 θ-strict and $A \cap \theta(B) = \emptyset$ and $C \cap \theta(B) = \emptyset$. Then $X = AB^*C$ is a maximal θ-solid code.*

Example 2. Let $\Sigma = \{a, b, c, d\}$ and θ be an antimorphic involution such that θ maps $a \mapsto c$ and $b \mapsto d$. Then $X = \{a\}\{b, d\}^*\{c\}$ is a maximal θ-solid code.

From the above definitions and propositions we can deduce 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 θ-strict 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 θ-strict for $i = 1, 2$, then $\Sigma_1\Sigma_2$ is maximal θ-solid code.*
3. *Let $A \subseteq \Sigma$ such that $A = \theta(A)$ and $X \subseteq A^+$. Then X is maximal θ-solid code over A if and only if $X \cup (\Sigma \setminus A)$ is maximal θ-solid code over Σ.*
4. *Let $B \subseteq \Sigma$ such that $B \cap \theta(B) = \emptyset$. Then $X = B^+$ is θ-solid code.*

The next proposition provides conditions under which the involution solid codes are preserved under a morphic or antimorphic mapping.

Proposition 9. *Let Σ_1 and Σ_2 be finite alphabet sets and let f be an injective morphism or antimorphism from Σ_1 to Σ_2^*. Let $\theta_1 : \Sigma_1^* \mapsto \Sigma_1^*$ and $\theta_2 : \Sigma_2^* \mapsto \Sigma_2^*$ be both morphic or both antimorphic involutions such that $f(\theta_1(x)) = \theta_2(f(x))$. Define $P = Pref(\theta_2(f(X))$, $S = Suff(\theta_2(f(X))$ and $A = \Sigma_2^* \setminus f(\Sigma_1^*)$.*

Suppose $(A^+P \cap SA^+) \cap f(\Sigma_1^+) = \emptyset$ *and* $A^+PA^+ \cap f(\Sigma_1) = \emptyset$. *Then*

1. *If* X *is* θ_1*-strict-infix* (*comma-free*) *then* $f(X)$ *is* θ_2*-strict-infix* (*comma-free*).
2. *If* X *is a* θ_1*-solid code then* $f(X)$ *is a* θ_2*-solid code.*

Proof. The first statement was proved in [14]. We consider the case of θ-solid codes. Let X be θ_1-solid code. Note that $f(X)$ is θ_2-infix by the first part of the proposition. We need to show that $\mathrm{PPref}(f(X)) \cap \mathrm{PSuff}(\theta_2(f(X))) = \emptyset$ and $\mathrm{PSuff}(f(X)) \cap \mathrm{PPref}(\theta_2(f(X))) = \emptyset$. Let θ_1 and θ_2 be morphic involutions and let f be an injective antimorphism. Suppose there exists $a \in \Sigma^+$ such that $a \in \mathrm{PPref}(f(x_1x_2))$ 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 follows similarly. Hence $f(X)$ is θ_2-solid. □

4 Involution Join Codes

In [10], Head, by using the coding properties relative to a language [9], showed how a sequence of subsets which are comma-free ([2, 16, 26]) from a set that is not comma-free can be obtained. The codes of this sequence are called join codes Similarly a sequence of subsets which are θ-comma-free codes from a given set that is not θ-comma-free can be obtained ([11, 12, 13, 18, 14]). The ith element in this sequence of codes θ-join codes is called θ-join code of level i. In this section we have several observations about these codes.

Definition 4. *Let* $X \subseteq \Sigma^*$ *and* $w \in \Sigma^*$. *Then the context of the word* w *in* X *is defined as the set* $C_X(w) = \{(u, v) : uwv \in X, u, v \in \Sigma^*\}$.

The following was defined in [10] and used in [7].

Definition 5. *A word* w *in* X *is a* join for X *if* $(u, v) \in C_{X^*}(w)$ *then both* u *and* v *are in* X^*. *The set of all joins for* X *is denoted* $J(X)$.

Recall that when X is a code, $J(X)$ is comma-free subset of X (see [10]), but $J(X)$ is not necessarily the maximal comma-free subset of X.s

Example 3. Let $X = \{aab, aba, bab\}$ over the alphabet set $\Sigma = \{a, b\}$. Note that $J(X) = \{aab\}$ but $Y = \{aab, bab\} \subseteq X$ is the maximal comma-free subset of X since $\Sigma^+Y\Sigma^+ \cap Y^2 = \emptyset$.

Similar to the definition for $J(X)$, we define $J_\theta(X)$ such that $J_\theta(X)$ is θ-comma-free. The authors in [7] define $J_\theta(X)$ as $J(X) \setminus \theta(J(X))$, but such defined $J_\theta(X)$ is not necessarily θ-comma-free in general. For example, consider $X = \{aab, bab, abbb\}$. For an antimorphic involution θ with $a \to b$ and $b \to a$, $\theta(X) = \{abb, aba, aaab\}$. Note that $\Sigma^+\{aab, abbb\}\Sigma^+ \cap X^2 = \emptyset$ but $\Sigma^+X\Sigma^+ \cap X^2 \neq \emptyset$. Hence $J(X) = \{aab, abbb\}$ is comma-free subset of X and $J_\theta(X) = J(X)$ since $\theta(J(X)) \cap J(X) = \emptyset$. But $J_\theta(X)$ is not θ-comma-free since, $aab\theta(aab)b = aab.abbb \in (J_\theta(X))^2$. We alter the definition for $J_\theta(X)$ such that $J_\theta(X)$ becomes θ-comma-free for all involutions θ.

Definition 6. *A word $w \in X$ is a* θ-join for X *if* $(u, v) \in C_{X^*}(\theta(w))$*, then both* u *and* v *are also in* X^**. The set of all* θ*-joins for* X *is denoted with* $J_\theta(X)$*.*

Lemma 2. *Let* $X \subseteq \Sigma^+$ *and* θ *be a morphic or antimorphic involution. Then the following hold true.*

1. $J_\theta(X) = J(X)$ *when* θ *is identity.*
2. *Let* $X \cup \theta(X)$ *be a code. When* X *is* θ*-comma-free,* $J_\theta(X) = X$*.*
3. $X \cup \theta(X)$ *is* θ*-comma-free if and only if* $X \cup \theta(X)$ *is comma-free and hence* $X \cup \theta(X)$ *is an infix code.*

Note that $X \cup \theta(X)$ being θ-infix does not necessarily imply that $X \cup \theta(X)$ is θ-comma-free. For example let $X = \{aa, aba\}$ with θ being antimorphic involution mapping $a \to b$ and $b \to a$. Then we have $\theta(X) = \{bb, bab\}$. It is easy to check that $X \cup \theta(X)$ is θ-infix but not θ-comma-free since $ababb = a(bab)b = a\theta(aba)b$.

Lemma 3. *If* $X \cup \theta(X)$ *is a code then* $J_\theta(X)$ *is* θ*-comma-free.*

Proof. Suppose $J_\theta(X)$ is not θ-comma-free, then there are $x, y, z \in J_\theta(X)$ such that $a\theta(z)b = xy$ for some $a, b \in \Sigma^*$. Since $z \in J_\theta(X)$ with $a\theta(z)b \in X^*$, for $a, b \in X$, xy has two distinct factorizations in $X \cup \theta(X)$. This contradicts that $X \cup \theta(X)$ is a code. Hence $J_\theta(X)$ is θ-comma-free. □

Corollary 3. *If* $X \cup \theta(X)$ *is a code then* $\Sigma^+\theta(J_\theta(X))\Sigma^+ \cap X^2 = \emptyset$*.*

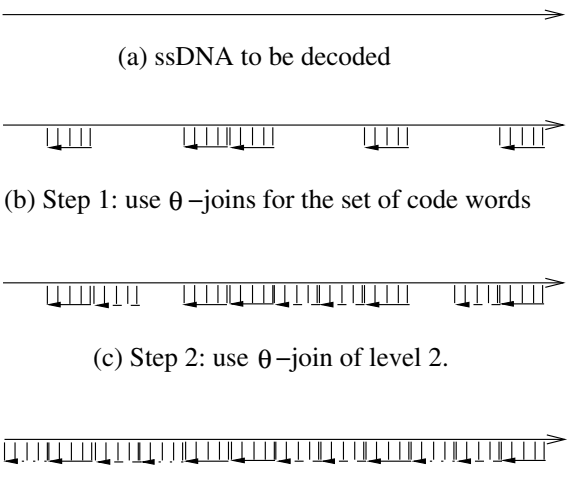
Note that $J_\theta(X)$ is not necessarily the maximal θ-comma-free subset of X.

Example 4. Let $X = \{abb, aab, aba\}$ over the alphabet set $\Sigma = \{a, b\}$ and for an antimorphic θ with $a \to b$ and $b \to a$, $\theta(X) = \{abb, bab, aab\}$. Note that $Y = \{aab, abb\}$ is the maximal θ-comma-free subset of X. But $J_\theta(X) = \{aab\}$.

Lemma 4. *Let* $X \cup \theta(X)$ *be a code and let* Y *be the maximal subset of* X *such that* $\Sigma^+\theta(Y)\Sigma^+ \cap X^2 = \emptyset$*. Then* $Y = J_\theta(X)$*.*

Proof. Since $J_\theta(X)$ is θ-comma-free, $J_\theta(X) \subseteq Y$. Note that $\Sigma^+\theta(Y)\Sigma^+ \cap X^2 = \emptyset$ if and only if $\Sigma^+\theta(Y)\Sigma^+ \cap X^* = \emptyset$. Then for $w \in Y$ and for all $(u, v) \in C_{X^*}(\theta(w))$, $u = v = 1$ which implies $w \in J_\theta(X)$. Hence $J_\theta(X) = Y$.

A possible communication by transferring single stranded DNA molecules can be done in the following way. To derive the meaning represented by a single stranded DNA molecule, by allowing attachments of complementary pieces of single stranded molecules. The meaning conveyed by the message molecule is expressed by the sequence of complementary code word molecules that attach. In such cases, each code word that is part of a θ-comma-free code has a unique place for hybridization, and the whole "message" molecule can be recovered. However, if the set of code words is not θ-comma-free, then we can extract a subset of code words that form a θ-comma-free set, i.e., the θ-joins for the code words. These code words would have unique places for annealing to the "message" DNA (see Fig. 2) leaving positions complementary to other words "empty". From the



(a) ssDNA to be decoded

(b) Step 1: use θ −joins for the set of code words

(c) Step 2: use θ −join of level 2.

(d) Step 3: use θ −join of level 3 to decode.

Fig. 2. Step-by-step recovery of a "message" encoded within a single stranded DNA

remaining code words another θ-comma-free set can be extracted, i.e., the θ-joins for the remaining code words. Now these words can anneal in a unique way to the the empty places that were not occupied by the first set of words. If this process ends, the "message" can be uniquely decoded. This process is schematically represented in Fig. 2.

Formally, let $X = X_0$ and $X_1 = X \setminus J_\theta(X)$. We define X_i, $i \geq 0$, a chain of descending subsets of X such that $X_{i+1} = X_i \setminus J_\theta(X_i)$ where $J_\theta(X_i)$ is a θ-join of X_i (see Fig. 3). We call $J_\theta(X_i)$ the θ-join at level i. When θ is identity the θ-join code at level 1 is precisely $J(X)$.

Definition 7. *The set X is called as* θ-split code *if* $X = \bigcup_{k=0}^{\infty} J_\theta(X_k)$.

If there exists m such that $X_{m+1} = \emptyset$ then X is called as θ-k-split where $k = min\{m : X_{m+1} = \emptyset\}$.

Example 5. Let $X = \{abb, aab, aba\}$ over the alphabet set $\Sigma = \{a, b\}$ and for an antimorphic θ with $a \to b$ and $b \to a$, $\theta(X) = \{abb, bab, aab\}$. Note that $J_\theta(X) = \{aab\}$ and hence $X_1 = X \setminus J_\theta(X) = \{abb, aba\}$. But $J_\theta(X_1) = \emptyset$. Hence X is not a θ-split code.

We assume that for a set X, $X \cup \theta(X)$ is a code throughout the rest of this section.

Proposition 10. *X is a θ-split code if and only if $\theta(X)$ is θ-split code.*

Note that it is possible to find an X such that X is θ-infix but not θ-split. For example let $X = \{aba, bab\}$ and let θ be an antimorphic involution such that θ maps $a \mapsto b$. Then X is θ-infix and $J_\theta(X) = \emptyset$.

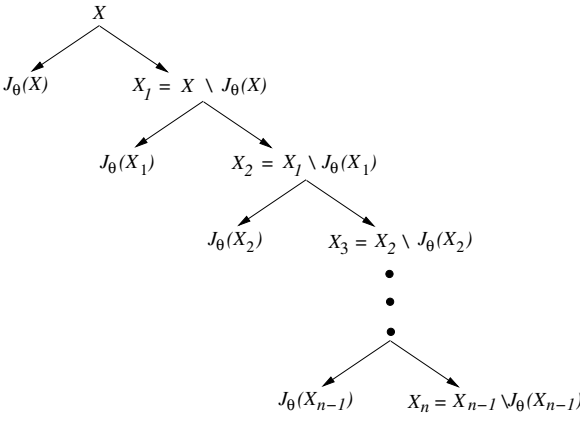


Fig. 3. The construction of θ-split code

Corollary 4. *If X and Y are such that $X \cup Y$ is a θ-split code, then XY is θ-split code.*

Corollary 5. *If X is a θ-split code, than X^n is a θ-split code for all $n \geq 1$.*

5 Concluding Remarks

The theory of codes, born in the context of information theory, has been developed as an independent subject using both combinatorial and algebraic methods. The objective of the theory of codes, from an elementary point of view, is the study of properties concerning factorizations of words into a sequence of words taken from a given set. Solid codes were introduced in [25] in the context of the study of disjunctive domains. Certain combinatorial properties of solid codes have been investigated in [15] and results concerning maximal solid codes of variable length were presented in [17]. In the hierarchy of codes, solid codes lie below the class of comma-free codes. By using code properties relative to a language [9], a sequence of generalizations of the concept of a comma-free code was developed in [10]. In other words, Head showed a way to partition a non-necessarily comma-free code into a sequence of subsets all of which are comma-free. The codes of this sequence are the join codes of various levels. The comma-free codes are precisely the join codes of the first level. The split codes of level k allow the segmentation of messages to be made in a sequence of k steps for which each step has the simplicity of a comma-free segmentation.

In this paper we extended the concepts of solid codes and join codes to incorporate the notion of an involution function replacing the identity function (An involution function θ is such that θ^2 equals identity). An involution code refers to any of the generalization of classical notion of codes ([2, 16, 26]) that replace the identity function with the involution morphic or antimorphic function in a way

explained in Definition 1. Involution codes were introduced in [11] in the process of designing DNA strands suitable for computation. Along these lines properties of θ-comma-free codes, θ-infix codes, θ-prefix codes, θ-outfix codes, θ-intercodes were introduced and studied in [11, 13, 14, 19, 21]. This paper completes this line of research by investigating the notions of θ-solid codes and θ-join codes. Several closure properties of θ-overlap free and θ-solid codes were discussed and we introduced θ-split codes as codes that can be "split", i.e., partition into a sequence of θ-comma-free codes. Properties of a code X that is a θ-split code of finite or infinite level remain to be determined. Note that if θ is the identity function, these notions become the well known notions of solid respectively join codes and the results obtained hold true for solid respectively join codes. Generalizations where θ is substituted with an arbitrary morphism seem as a natural next step.

Acknowledgment. Research supported by NSERC and Canada Research Chair grants for Lila Kari and NSF grants CCF–0523928 and CCF–0432009 for Natasha Jonoska.

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,* Academis Press, Inc. Orlando Florida, 1985.
3. R.Deaton, J.Chen, H.Bi, M.Garzon, H.Rubin, D.F.Wood, *A PCR based protocol for in 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) 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. C. Ferreti and G. Mauri, *Remarks on Relativisations and DNA Encodings* Aspects of Molecular Computing, N.Jonoska, G.Paun, G.Rozenberg editors, Springer LNCS 2950 (2004) 132-138.
8. M. Garzon, R. Deaton, D. Reanult, *Virtual test tubes: a new methodology for computing,* Proc. 7th. Int. Symposium on String Processing and Information retrieval, A Coruňa, Spain. IEEE Computing Society Press (2000) 116-121.
9. T.Head, *Unique decipherability relative to a language*, Tamkang J. Math, 11(1980) 59-66.
10. T. Head, *Relativized code concepts and multi-tube DNA dictionaries*, Finite vs Infinite, C.S. Calude and G. Paun editors, (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), 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 and K. Mahalingam, *Languages od DNA based code words*, Proceedings of the 9th International Meeting on DNA Based Computers, (J.Chen, J.Reif editors), LNCS 2943 (2004) 61-73.
14. N.Jonoska, K.Mahalingam, J.Chen *Involution codes: with application to DNA coded languages*, Natural Computing, Vol 4-2 (2005) 141-162.
15. H.Jürgensen, S.S.Yu, *Solid Codes*, Journal of Information Processing Cybernatics, EIK 26 (1990) 10, 563-574.
16. H.Jürgensen, S.Konstantinidis, *Codes*, Handbook of Formal Languages, Vol 1, (G.Rozenberg and A.Salomaa editors), Springer Verlag (1997) 511-608.
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, S.Konstantinidis, E.Losseva, G.Wozniak, *Sticky-free and overhang-free DNA languages*, Acta Informatica 40 (2003) 119-157.
19. L.Kari and K.Mahalingam, *DNA Codes and their properties*, Submitted.
20. L.Kari, S.Konstantinidis and P.Sosik, *Bond-free languages: formalizations, maximality and construction methods*, Proceedings of the 10th International Meeting on DNA Computing, (C.Ferreti, G.Mauri, C.Zandron editors) LNCS 3384 (2005) 169-181.
21. L.Kari, S.Konstantinidis, P.Sosik, *Preventing undesirable bonds between DNA codewords* Proceedings of the 10th International Meeting on DNA Computing, LNCS 3384 (2005) 182-191.
22. K.Mahalingam, *Involution Codes: With Application to DNA Strand Design* Ph.d. Thesis, University of South Florida, Tampa, FL, 2004.
23. A. Marathe, A.E. Condon, R.M. Corn, *On combinatorial word design*, Preproceedings of the 5th International Meeting on DNA Based Computers, Boston (1999) 75-88.
24. G.Mauri, C.Ferretti, *Word design for molecular computing: a survey.* Proceedings of the 9th International Meeting on DNA Based Computers, (J.Chen, J.H.Reif editors), LNCS 2943 (2004) 37-47.
25. H.J, Shyr, S.S.Yu , *Solid codes and disjunctive domains*, Semigroup Forum 41(1990) 23-37.
26. H.J.Shyr, *Free Monoids and Languages*, Hon Min Book Company 2001.