DNA Codes and Their Properties

Lila Kari and Kalpana Mahalingam

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

Abstract. One of the main research topics in DNA computing is associated with the design of information encoding single or double stranded DNA strands that are "suitable" for computation. Double stranded or partially double stranded DNA occurs as a result of binding between complementary DNA single strands (A is complementary to T and C is complementary to G). This paper continues the study of the algebraic properties of DNA word sets that ensure that certain undesirable bonds do not occur. We formalize and investigate such properties of sets of sequences, e.g., where no complement of a sequence is a prefix or suffix of another sequence or no complement of a concatenation of n sequences is a subword of the concatenation of $n+1$ sequences. The sets of code words that satisfy the above properties are called θ-prefix, θ-suffix and θ-intercode respectively, where θ is the formalization of the Watson-Crick complementarity. Lastly we develop certain methods of constructing such sets of DNA words with good properties and compute their informational entropy.

1 Introduction

Several attempts have been made to address the problem of encoding information on DNA and many authors have proposed various solutions. A common approach has been to use the Hamming distance [2,7,8,9,25]. Experimental separation of strands with "good" sequences that avoid intermolecular cross hybridization was reported in [5,6]. In [12], 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 [14,16,18]. Based on these ideas and code-theoretic properties, a computer program for generating code words is being developed [13,20]. Another algorithm, based on backtracking, for generating such code words is also developed by Li [22]. In [21] the author used the notion of partial words with holes for the design of DNA strands.

In this paper we continue the study of the algebraic properties of DNA languages suitable for computation. More precisely, 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 (W/C) complementarity of the nucleotides defines a natural involution mapping

C. Mao and T. Yokomori (Eds.): DNA12, LNCS 4287, pp. 127–142, 2006.

θ, $A \mapsto T$ and $G \mapsto C$ which is an anti-morphism 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-prefix, θ-suffix and θ-intercode (i.e.) languages where no Watson-Crick complement of a word is a prefix or suffix of another word, respectively no Watson-Crick complement of a composition of n words is a subword of a composition of $n+1$ words (See Fig 1 for the types of hybridizations that are avoided if a word set satisfies these properties). We start the paper with definitions of coding properties that avoid intermolecular cross hybridizations. The notions of θ-prefix and θ-suffix languages have been defined in [16] under the names of θ-p-compliant and θ-s-compliant respectively. Here we also consider two additional coding properties namely θ-bifix code and θ-intercode. 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.

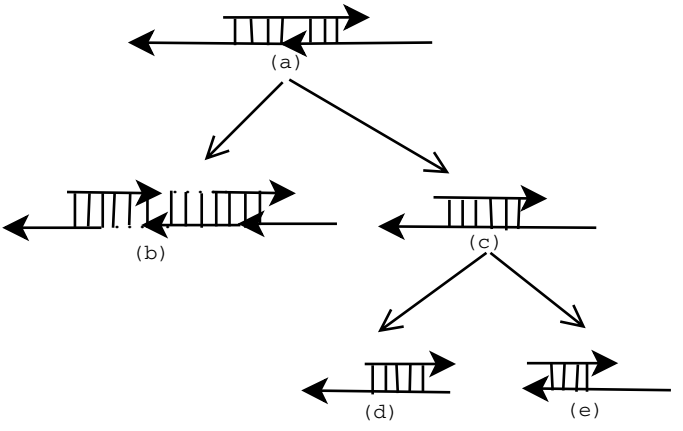

Fig. 1. Various types of intermolecular hybridization that we want to avoid: (a) a code word is the reverse complement of a subword of a concatenation of two other code words: θ-comma-free codes avoid such hybridizations, (b) the catenation of m codewords is the reverse complement of a subword of a concatenation of composition of $m+1$ code words: θ-intercodes (a new notion introduced in this paper) avoid such hybridizations (c) a code word is a reverse complement of a subword of another code word: θ-infix codes avoid such hybridizations (d) a code word is the reverse complement of a suffix of another code word: θ-suffix codes avoid such hybridizations, (e) a code word is the reverse complement of a prefix of another code word: θ-prefix codes avoid such hybiridzations. (The $3'$ end is indicated by an arrow.)

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 closure properties of various types of involution codes. Algebraic properties of θ-intercodes are discussed in Section 4. We introduce and discuss the properties of sets whose n element subsets are θ-intercodes and θ-comma-free codes in Section 5. Section 6

describes several methods to generate involution codes and also calculate their informational entropy. Since it turns out that the entropy of these generated involution codes is greater than $\log 2$, by the coding theorem ([1,23]) it follows that the constructed code words can be used to encode binary strings. We end with a few concluding remarks.

2 Definitions and Properties

An alphabet Σ is a finite non-empty set of symbols. We will denote by Δ the special case when the alphabet is $\{A, G, C, T\}$ representing the DNA nucleotides. 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. We note that with the concatenation operation on words, Σ^* is the free monoid and Σ^+ is the free semigroup generated by Σ. The length of a word $u = a_1 \cdots a_n$ is n and is denoted by $|u|$. For words representing DNA sequences, we will use the following convention. A word u over Δ denotes a DNA strand in its $5' \to 3'$ orientation. The Watson-Crick complement of the word u, also in orientation $5' \to 3'$ is denoted by $\overleftarrow{u}$. For example if $u = AGGC$ then $\overleftarrow{u} = GCCT$.

Throughout the rest of the paper, 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 [4,26]. We will need the following definitions:

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

We define the set of prefixes, suffixes and subwords of a set of words as Pref , Suff and Sub. Similarly, we have $\mathrm{Suff}_k(w) = \mathrm{Suff}(w) \cap \Sigma^k, \mathrm{Pref}_k(w) = \mathrm{Pref}(w) \cap \Sigma^k, \mathrm{Sub}_k(w) = \mathrm{Sub}(w) \cap \Sigma^k$. We follow the definitions initiated in [12] and used in [13]. An involution $\theta : \Sigma \to \Sigma$ of a set Σ is a mapping such that θ^2 equals the identity mapping, $\theta(\theta(x)) = x$, $\forall x \in \Sigma$. The mapping $\nu : \Delta \to \Delta$ defined by $\nu(A) = T$, $\nu(T) = A$, $\nu(C) = G$, $\nu(G) = C$ is an involution on Δ and can be extended to a morphic involution of Δ^*. Since the Watson-Crick complementarity appears in a reverse orientation, we consider another involution $\rho : \Delta^* \to \Delta^*$ defined inductively, $\rho(s) = s$ for $s \in \Delta$ and $\rho(us) = \rho(s)\rho(u) = s\rho(u)$ for all $s \in \Delta$ and $u \in \Delta^*$. This involution is antimorphism such that $\rho(uv) = \rho(v)\rho(u)$. The Watson-Crick complementarity then is the antimorphic involution obtained by the composition $\nu\rho = \rho\nu$. Hence for a DNA strand u we have that $\rho\nu(u) = \nu\rho(u) = \overleftarrow{u}$. The involution ρ reverses the order of the letters in a word and as such is used in the rest of the paper.

The following Definition 1 [14,16] introduces notions meant to formalize a variety of language properties, each of whom guarantees the absence of a certain unwanted hybridization. The notion of θ-infix and θ-comma-free code were

introduced in [12] and was called θ-compliant and θ-free respectively. The definition of θ-intercode and θ-outfix code are new notions introduced here.

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-code *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-code *if $X^2 \cap \Sigma^+\theta(X)\Sigma^+ = \emptyset$.*
3. *The set X is called* θ-strict-code *if $X \cap \theta(X) = \emptyset$.*
4. *The set X is called* θ-prefix-code *if $X \cap \theta(X)\Sigma^+ = \emptyset$.*
5. *The set X is called* θ-suffix-code *if $X \cap \Sigma^+\theta(X) = \emptyset$.*
6. *The set X is called* θ-bifix-code *if X is both θ-prefix and θ-suffix.*
7. *The set X is called a* θ-intercode *if $X^{m+1} \cap \Sigma^+\theta(X^m)\Sigma^+ = \emptyset$, $m \geq 1$. The integer m is called the index of X.*
8. *The set X is called* θ-outfix-code *if for $u, \theta(u_1)x\theta(u_2) \in X$ with $\theta(u) = \theta(u_1)\theta(u_2)$ implies $x = 1$.*

Note that θ-infix languages avoid undesirable hybridization of the type depicted in Fig 1c, θ-comma-free languages avoid undesirable hybridization of the type depicted in Fig 1a, θ-intercodes avoid undesirable hybridization of the type depicted in Fig 1b, θ-suffix languages avoid undesirable hybridization of the type depicted in Fig 1d, θ-prefix languages avoid undesirable hybridization of the type depicted in Fig 1e, θ-outfix languages avoid undesirable hybridization of the type depicted in Fig 2. Note that a θ-intercode of index one is θ-comma-free.

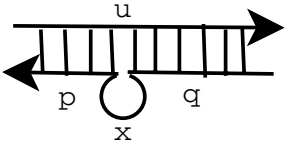

Fig. 2. Another type of intermolecular hybridization that we want to avoid: the reverse complement of a code word is a concatenation of a prefix and a suffix of another code word. A θ-outfix code (a new notion defined in this paper) avoids such hybridizations.

Also note that X is θ-intercode of index m if and only if $\theta(X)$ is θ-intercode of index m. We have defined several properties that are desirable for DNA languages to have. The properties 1 to 4 in Definition 1 have been extensively studied in [12,14,16]. Here we complete this study by proving the relationship between several properties. The following proposition shows the connection between θ-infix and θ-comma-free languages. We use $\theta : \Sigma^* \mapsto \Sigma^*$ to be either morphic or antimorphic involution throughout this paper unless specified. In the following, we list some of the properties and relations between θ-infix and θ-comma-free code.

Proposition 1. *Let $\theta : \Sigma^* \to \Sigma^*$ be a morphic or antimorphic involution and $X \subseteq \Sigma^*$. Then the following are equivalent.*

1. *X is a θ-comma-free code.*
2. *X is θ-infix and $\theta(X) \cap PSuff(X)PPref(X) = \emptyset$.*
3. *X is θ-infix and $X^2 \cap PPref(X)\theta(X)PSuff(X) = \emptyset$.*
4. *X is θ-infix and $X^n \cap (\Sigma^+\theta(X)\Sigma^+X^{n-2}) = \emptyset$.*
5. *X is θ-infix and $X^n \cap (X^{n-2}\Sigma^+\theta(X)\Sigma^+) = \emptyset$.*

Proposition 2. *Let $X \subseteq \Sigma^+$ be a θ-infix code. Then $X^3 \cap \Sigma^+\theta(X^2)\Sigma^+ = \emptyset$ if and only if $\theta(X^2) \cap PSuff(X)XPPref(X) = \emptyset$.*

Corollary 1. *If $X \subseteq \Sigma^+$ is θ-comma-free then $X^2 \cap PSuff(X)\theta(X)PPref(X) = \emptyset$ and $\theta(X^2) \cap PSuff(X)XPPref(X) = \emptyset$.*

3 Closure Properties of Involution Codes

In this section we discuss several properties of θ-prefix, θ-suffix, θ-bifix, θ-outfix codes and θ-strict codes. Besides being generalizations of outfix codes, the motivation behind introducing the notion of θ-outfix codes comes from the fact that a set of DNA words that is a θ-outfix code avoids any undesirable hybridization of the type in Fig 2. Ensuring that no such unwanted hybridization occurs is obviously desirable from an experimental view point. It is interesting to note that certain properties that are not satisfied by θ-prefix and θ-suffix codes are satisfied by θ-bifix codes. In particular we discuss the conditions under which such languages are closed under arbitrary concatenation. From a practical point of view, these results give conditions under which, given a small finite set of "good" codewords, we can construct arbitrarily large sets of good code words by concatenation.

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

1. *If X is θ-infix then X is both θ-prefix and θ-suffix and hence a θ-bifix code.*
2. *For a morphic involution θ, X is θ-prefix(suffix) if and only if $\theta(X)$ is θ-prefix(suffix).*
3. *For an antimorphic involution θ, X is θ-prefix(suffix) if and only if $\theta(X)$ is θ-suffix(prefix).*
4. *X is θ-bifix if and only if $\theta(X)$ is θ-bifix.*

In the next proposition we show that the family of θ-prefix(suffix) codes is closed under concatenation when θ is a morphic involution and hence for a θ-prefix(suffix) code X, any arbitrary power of X is also a θ-prefix(suffix) code when θ is a morphic involution.

Proposition 3. *For a morphic involution θ, the family of θ-prefix (θ-suffix) codes is closed under concatenation.*

Proof. Let X_1 and X_2 be θ-prefix. Suppose X_1X_2 is not θ-prefix then there exists $x_1x_2, y_1y_2 \in X_1X_2$ such that $x_1x_2 = \theta(y_1y_2)b = \theta(y_1)\theta(y_2)b$ for some $b \in \Sigma^+$. Note that neither x_1 is a prefix of $\theta(y_1)$ nor $\theta(y_1)$ is a prefix of x_1 since both X_1 and $\theta(X_1)$ (Please refer 1) are θ-prefix. Hence $x_1 = \theta(y_1)$ which implies that $\theta(y_2)$ is a prefix of x_2 which is a contradiction to our assumption that X_2 is θ-prefix. Similar proof works for θ-suffix. □

Corollary 2. *Let θ be a morphic involution.*

1. *If X is θ-prefix then X^n is θ-prefix for all $n \geq 1$.*
2. *If X is θ-suffix then X^n is θ-suffix for all $n \geq 1$.*

Note that the above proposition does not hold when θ is an antimorphic involution. For example let $X_1 = \{aa, baa\}$ and $X_2 = \{bb, bbb\}$ over the alphabet set $\Sigma = \{a, b\}$ and let θ be antimorphism such that $a \mapsto b$ and $b \mapsto a$. Note that both X_1 and X_2 are θ-prefix but X_1X_2 is not θ-prefix since for $aabb \in \theta(X_1X_2)$, $aabbb \in X_1X_2$. Hence when θ is an antimorphic involution we need an additional restriction on the sets X_1 and X_2 which is shown in the next proposition.

Proposition 4. *For an antimorphic involution θ, if X_1 and X_2 are such that $X_1 \cup X_2$ is θ-bifix, then X_1X_2 and X_2X_1 are θ-bifix.*

Proof. Immediate.

Corollary 3. *Let θ be morphic or antimorphic involution on Σ^*. If X is a θ-bifix code then X^n is a θ-bifix code for all $n \geq 1$.*

In the next proposition we provide with the necessary condition under which for a set X, the Kleene star of the set X is θ-prefix(suffix).

Proposition 5. *If X is such that X is θ-strict-infix code then X^+ is both θ-prefix and θ-suffix.*

Proof. To show that X^+ is θ-prefix (i.e.) to show that $X^+ \cap \theta(X^+)\Sigma^+ = \emptyset$. Suppose X^+ is not θ-prefix code then there exists $x_1x_2...x_n = \theta(y_1...y_m)b$ for $x_i, y_j \in X, i = 1, .., n, j = 1, ..., m$ and $b \in \Sigma^+$. For a morphic θ, $x_1x_2...x_n = \theta(y_1)...\theta(y_m)b$ implies either x_1 is a subword of $\theta(y_1)$ or $x_1 = \theta(y_1)$ or $\theta(y_1)$ is a subword of x_1. All cases contradict our assumption that X is strictly θ-infix. Similarly we can prove that X^+ is θ-suffix code.

The next two propositions gives us conditions under which when a composition of some arbitrary languages satisfy good encoding properties, the right and the left context of such languages also satisfy the same good encoding properties.

Proposition 6. *Let $X \subseteq \Sigma^+$ be such that X is not a θ-strict code.*

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

Proposition 7. *Let X_i, $i = 1, 2, ..., m$ be non empty languages over Σ such that $X_i \cap \theta(X_i) \neq \emptyset$, $i = 1, 2, ..., m$. Let θ be a morphic involution. Then the following are true.*

1. *If $X_1X_2...X_m$ is θ-prefix, then $X_2...X_m$, $X_3...X_m$,..., $X_{m-1}X_m$, X_m are θ-prefix codes.*

2. *If $X_1X_2...X_m$ is θ-suffix, then $X_1...X_{m-1}$, $X_1...X_{m-2}$,..., X_1X_2, X_1 are θ-suffix codes.*

Proof. We prove (i) for the case $m = 2$ and the result follows by induction. Assume X_1X_2 is θ-prefix. If X_2 is not θ-prefix code then there exists $x_2, y_2 \in X_2$ such that $x_2 = \theta(y_2)b$. Since $X_1 \cap \theta(X_1) \neq \emptyset$, there exists $x_1, y_1 \in X$ such that $x_1 = \theta(y_1)$ and hence $x_1x_2 = \theta(y_1)\theta(y_2)b = \theta(y_1y_2)b$ which is a contradiction to our assumption that X_1X_2 is θ-prefix. Similar proof works for (ii). □

In the following propositions we investigate certain properties of θ-outfix codes. We recall the following definition of insertion into a set from [15].

For $X \subseteq \Sigma^+$, let

$$Y = \theta(X) \leftarrow \Sigma^+ = \bigcup_{u \in \theta(X), v \in \Sigma^+} (u \leftarrow v)$$

where, $u \leftarrow v = \{u_1vu_2 : u = u_1u_2, u_1, u_2 \in \Sigma^*\}$.

The next lemma is a direct consequence of the definition of θ-outfix codes.

Lemma 2. *For $X \subseteq \Sigma^+$ let Y be the set obtained above. Then X is a θ-outfix code iff $Y \cap X = \emptyset$.*

Corollary 4. *For a regular X, it is decidable whether X is a θ-outfix code or not.*

Proof. For a regular X, $\theta(X)$ is regular and it has been show in [15] that for regular set $\theta(X)$ and Σ^+, $Y = \theta(X) \leftarrow \Sigma^+$ is also regular. It is decidable whether a regular set is empty or not. Hence it is decidable whether X is a θ-outfix code or not.

It is easy to see that every θ-outfix code is θ-prefix and θ-suffix and hence a θ-bifix code. Also note that X is θ-outfix code if and only if $\theta(X)$ is a θ-outfix code. In the following propositions we investigate the closure properties of θ-outfix codes. In most cases we omit the proof.

Proposition 8. *For a morphic involution θ, the family of θ-outfix codes is closed under concatenation.*

Note that the above proposition does not hold when θ is an anitmorphic involution. For example let $X_1 = \{aa, baa\}$ and $X_2 = \{bb, bbb\}$ over the alphabet set $\Sigma = \{a, b\}$ and let θ be antimorphism such that $a \mapsto b$ and $b \mapsto a$. Note that both X_1 and X_2 are θ-outfix but X_1X_2 is not θ-outfix since for $aabb \in \theta(X_1X_2)$, $aa(b)bb \in X_1X_2$. But the cases become simpler if we just work with one set X. In the next proposition, we show that a θ-outfix code is closed under arbitrary concatenation with itself for both morphic and antimorphic involution.

Proposition 9. *X is a θ-outfix code iff X^+ is a θ-outfix code.*

Proposition 10. *For a morphic involution θ, let $X_1, X_2 \subseteq \Sigma^+$ be such that $X_i \cap \theta(X_i) \neq \emptyset$ for $i = 1, 2$. If X_1X_2 is θ-outfix code then both X_1 and X_2 are θ-outfix codes.*

Proof. Suppose that X_1 is not θ-outfix, then $xy, \theta(x)u\theta(y) \in X_1$ for some $u \in \Sigma^+, x, y \in \Sigma^*$. Consider $z \in X_2 \cap \theta(X_2)$. Then $xyz \in X_1X_2$ and $\theta(x)u\theta(y)z \in X_1X_2$, a contradiction. Hence X_1 must be a θ-outfix code. Similarly, we can show that X_2 is a θ-outfix code. □

In the next proposition we investigate certain properties of θ-strict codes and their relation with other sets of codes.

Lemma 3. *1. If $X_1, X_2 \subseteq \Sigma^+$ are θ-strict, then $X_1 \cup X_2$ is not necessarily θ-strict.*
2. Let X_1, X_2 be θ-strict. Then $X_1 \cap \theta(X_2) = \emptyset$ and $X_2 \cap \theta(X_1) = \emptyset$ if and only if $X_1 \cup X_2$ is θ-strict.
3. If X_1 and X_2 are θ-strict, then $X_1 \cap X_2$ is θ-strict.
4. Let X_1 and X_2 be θ-strict. When θ is morphism, if one of X_1 or X_2 is θ-prefix , then X_1X_2 is θ-strict. When θ is antimorphism, if $X_1 \cup X_2$ is θ-strict-bifix, then X_1X_2 is θ-strict.
5. If X is θ-strict-bifix, then X^+ is θ-strict.
6. X is θ-strict if and only if $\theta(X)$ is θ-strict.

4 Involution Intercodes

We now generalize the concept of θ-comma-free codes to θ-intercodes and study the properties of such codes. Note that if θ is the identity function, a θ-intercode becomes the well known notion of intercode, widely studied in the literature [26]. Besides being generalizations of intercodes, the motivation behind introducing the notion of θ-intercodes comes from the fact that a set of DNA words that is a θ-intercode avoids any undesirable hybridization of the type in Fig 1b. Ensuring that no such unwanted hybridization occurs is obviously desirable from an experimental view point.

Proposition 11. *Let X be a regular language. Then for a given $m \geq 1$, it is decidable whether or not X is a θ-intercode of index m.*

Proof. X is a θ-intercode of index m if and only if $X^{m+1} \cap \Sigma^+\theta(X^m)\Sigma^+ = \emptyset$. Since the family of regular languages is closed under catenation and intersection, X^{m+1} and $\Sigma^+\theta(X^m)\Sigma^+$, $\theta(X^m)$ and $X^{m+1} \cap \Sigma^+\theta(X^m)\Sigma^+$ are regular. It is decidable whether a regular language is empty or not.

Proposition 12. *Let $|\Sigma| \geq 2$. Then for any $m \geq 1$, every θ-intercode of index m is a θ-intercode of index $m + 1$.*

Proof. Given X is a θ-intercode of index m hence $X^{m+1} \cap \Sigma^+\theta(X^m)\Sigma^+ = \emptyset$. Suppose $X^{m+2} \cap \Sigma^+\theta(X^{m+1})\Sigma^+ \neq \emptyset$ then there exists $x_1, x_2, ...x_{m+2}$, y_1, y_2, $...y_{m+1} \in X$ such that $x_1x_2...x_{m+2} = a\theta(y_1...y_{m+1})b$ for some $a, b \in \Sigma^+$. If $|a| \geq |x_1|$ then $x_2...x_{m+2} \in \Sigma^+\theta(y_2)...\theta(y_{m+1})\Sigma^+$. If $|b| \geq |x_{m+2}|$ then $x_1...x_{m+1} \in \Sigma^+\theta(y_1)...\theta(y_m)\Sigma^+$. If $|a| < |x_1|$, $|b| < |x_{m+2}|$ then $a_1x_2...x_{m+1}b_1 = \theta(y_1)...\theta(y_{m+1})$ which implies $y_1...y_{m+1} = a_2\theta(x_2...x_{m+1})b_2$ which is a contradiction.

Both cases contradict our assumption that X is a θ-intercode of index m. □

Proposition 13. *For any involution θ, every θ-intercode X such that $X \cap \theta(X) \neq \emptyset$ is a θ-bifix code.*

Proof. X is a θ-intercode of index m, then by definition $X^{m+1} \cap \Sigma^+\theta(X^m)\Sigma^+ = \emptyset$. Since $X \subseteq \Sigma^+$ and $X^{m+1} \cap \theta(X^{m+1})\Sigma^+ \subseteq X^{m+1} \cap \Sigma^+\theta(X^m)\Sigma^+ = \emptyset$ we have X^{m+1} is a θ-prefix code which implies X is a θ-prefix code by proposition 6. Similarly we can show that X is a θ-suffix code. □

The converse of the above proposition is not true. For example let $X = \{aab, aba\}$ over the alphabet set $\Sigma = \{a, b\}$. Let θ be a morphic involution with $a \mapsto b$ and $b \mapsto a$. Note that X is both θ-prefix and θ-suffix but $aa\theta(aba)a = aababa \in X^2$. Hence X is not θ-intercode of index one. Also it is shown in [12] that every θ-comma-free code is θ-infix. But this is not the case for θ-intercodes of index $m \geq 2$. One example is as follows: Let $X = \{b^2ab^3ab^2, a^3\}$ over the alphabet set $\Sigma = \{a, b\}$ and let θ be an antimorphic involution such that $a \mapsto b$ and $b \mapsto a$. The language X is θ-intercode of index 2 but not a θ-infix code.

Proposition 14. *If X is θ-comma-free code then X is a θ-intercode of index m for all $m \geq 1$.*

Proof. Suppose $X^{m+1} \cap \Sigma^+\theta(X^m)\Sigma^+ \neq \emptyset$, then there exists $x_1, x_2, ...x_{m+1}$, $y_1, y_2, ..., y_m \in X$ and $a, b \in \Sigma^+$ such that $x_1x_2...x_{m+1} = a\theta(y_1y_2...y_m)b$. Then either $\theta(y_i)$ is a subword of x_j which contradicts X being θ-infix and hence θ-comma-free or $\theta(y_i)$ is a subword of x_jx_{j+1} which contradicts X being θ-comma-free. □

Note that the converse of the above proposition is not true. For example let $X = \{cbaa, baad, babb\}$ over the alphabet set $\Sigma = \{a, b, c, d\}$. Let θ be an antimorphic involution with $a \mapsto b$ and $c \mapsto d$. It is easy to check that $X^3 \cap \Sigma^+\theta(X^2)\Sigma^+ = \emptyset$ but X is not θ-comma-free since $cb\theta(babb)ad = cbaabaad \in X^2$.

For any word $u = a_1a_2...a_n \in \Sigma^*$ with $a_i \in \Sigma$ define the reverse of u as $\hat{u} = a_na_{n-1}...a_2a_1$. For $X \subseteq \Sigma^+$, define $\hat{X} = \{\hat{u} : u \in X$. The following characterization of θ-intercodes of index m is an immediate result from the definition of θ-intercodes.

Proposition 15. *Let $X \subseteq \Sigma^+$. The following are equivalent.*

1. *X is a θ-intercode of index m.*
2. *$\hat{X}$ is a θ-intercode of index m.*
3. *For any $u \in X^m$, $x, y \in \Sigma^*$, $x\theta(u)y \in X^{m+1}$ implies $x = 1$ or $y = 1$.*

Proposition 16. *If X is a θ-intercode of index m then $X^k \cap \Sigma^+\theta(X^m)\Sigma^+ = \emptyset$ for all $k \leq m + 1$.*

Proof. Suppose $X^k \cap \Sigma^+\theta(X^m)\Sigma^+ \neq \emptyset$ for some $k < m + 1$, then $\emptyset \neq X^{m+1} \cap X^{m+1-k}\Sigma^+\theta(X^m)\Sigma^+ \subseteq X^{m+1} \cap \Sigma^+\theta(X^m)\Sigma^+$ which is a contradiction to our assumption X is a θ-intercode of index m.

Proposition 17. *If $X \subseteq \Sigma^+$ is a θ-intercode of index m, $m \geq 1$ and X is strictly θ-infix, then $X^n \cap \Sigma^+\theta(X^m)\Sigma^+ = \emptyset$ and $X^m \cap \Sigma\theta(X^n)\Sigma^+ = \emptyset$ for all $n \geq m$.*

Proof. We prove by induction on n. Suppose for $n = m$, $x \in X^m \cap \Sigma^+\theta(X^m)\Sigma^+$ then $x = x_1...x_m = a\theta(y_1)...\theta(y_m)b$. Then atleast one of the $\theta(y_i)$ is a subword of x_j which is a contradiction to our assumption that X is strictly θ-infix. When $n = m+1$, $X^{m+1} \cap \Sigma^+\theta(X^m)\Sigma^+ = \emptyset$ since X is a θ-intercode. Now assume that $X^n \cap \Sigma^+\theta(X^m)\Sigma^+ = \emptyset$ for all $m \leq n \leq k$. Suppose $X^{k+1} \cap \Sigma^+\theta(X^m)\Sigma^+ \neq \emptyset$, then there exists $x_1, x_2, ..., x_{k+1}, y_1, y_2..., y_m \in X$ such that $x_1x_2...x_{k+1} = a\theta(y_1)\theta(y_2)...\theta(y_m)b$. If $|a| = |x_1|$ then $x_2...x_k = \theta(y_1)...\theta(y_m)b$ which implies either x_2 is a subword of $\theta(y_1)$ or $\theta(y_1)$ is a subword of x_2 which contradicts our assumption that X is strictly θ-infix. If $|a| > |x_1|$ then $x_2...x_k = a_1\theta(y_1)...\theta(y_m)b$ which implies that $X^{k-1} \cap \Sigma^+\theta(X^m)\Sigma^+ \neq \emptyset$ which is a contradiction to our induction hypothesis. The cases when $|b| = |x_{k+1}|$ or $|b| > |x_{k+1}|$ are similar to the case when $|a| = |x_1|$ or $|a| > |x_1|$ respectively. If $|a| < |x_1|$ and $|b| < |x_k|$ then atleast one of $\theta(y_i)$ is a subword of x_j which is a contradiction to our assumption that X is strictly θ-infix. □

Proposition 18. *If X is a θ-intercode of index m and X is strictly θ-infix, then X^n is a θ-intercode of index m, for all $n \geq 1$.*

Proof. We need to show that $(X^n)^{m+1} \cap \Sigma^+(\theta(X^n))^m\Sigma^+ = \emptyset$ for all $n > 1$. Then $(X^n)^{m+1} \cap \Sigma^+(\theta(X^n))^m\Sigma^+ = (X^{nm+n}) \cap \Sigma^+\theta(X^{nm})\Sigma^+$. Note that by proposition 12 X is a θ-intercode of index nm and hence $X^{nm+1} \cap \Sigma^+\theta(X^{nm})\Sigma^+ = \emptyset$. Then by proposition 17 $(X^{nm+n}) \cap \Sigma^+\theta(X^{nm})\Sigma^+ = \emptyset$. Hence X^n is a θ-intercode of index m. □

Proposition 19. *If X is a θ-intercode of index m and X is strictly θ-infix then X^+ is a θ-intercode of index m.*

Proof. Suppose X^+ is not θ-intercode of index m, then there exists $x, y \in X^+$ such that $x = x_1...x_{m+1}$ and $y = y_1..y_m$ for all $x_i, y_j \in X^+$ and $x = a\theta(y)b$ for some $a, b \in \Sigma^+$. The case when $x_i, y_j \in X$ for all i, j then $x \in X^{m+1} \cap \Sigma^+\theta(X^m)\Sigma^+$ which is a contradiction. If $x \in X^p$ and $y \in X^q$ for some $p, q \geq m$ then $x \in X^p \cap \Sigma^+\theta(X^q)\Sigma^+$ which is a contradiction by Proposition 12 and 17. Hence X^+ is a θ-intercode of index m. □

Proposition 20. *If $X \cup Y$ is a θ-intercode of index m then XY is a θ-intercode of index m.*

Proof. Suppose $(XY)^{m+1} \cap \Sigma^+\theta((XY)^m)\Sigma^+ \neq \emptyset$ then let $r \in (XY)^{m+1}$ such that $r = x_1y_1x_2y_2...x_{m+1}y_{m+1} = a\theta(p_1q_1...p_mq_m)b$ for $x_1, ..., x_{m+1}$, $p_1, ..., p_m \in X$ and $y_1, ..., y_{m+1}$, $q_1, ..., q_m \in Y$ and $a, b \in \Sigma^+$. But $r \in (X \cup Y)^{2(m+1)} \cap \Sigma^+\theta((X \cup Y)^{2m})\Sigma^+$ which is a contradiction by Proposition 12 and 17. □

5 n-θ-Comma-Free Codes and n-θ-Intercodes

If the alphabet Σ consists of more than one letter, the partial order $\leq_c$ defined on Σ^* by $u \leq_c v$ if and only if $v = xu = ux$ for some $x \in \Sigma^*$ plays an interesting role. That is if $u \leq_c v$, then $u = f^i$ for some primitive word f (f is primitive if $f = a^i$, $a \in \Sigma^+$ for some i implies $i = 1$) and $v = f^{i+j}$ for some $j \geq 0$. Thus if $u, v \in X \subseteq \Sigma^+$ and X is an independent set with respect to $\leq_c$, then $uv \neq vu$, which is equivalent to the fact that the two element set $\{u, v\}$ is a code. Hence a $\leq_c$-independent set is called a 2-code. This notion can be generalized as follows: An n-code is a set X with the property that every n element subset of the set X is a code ([26]). The notion of n-codes, n-comma free codes and hence n-intercodes were defined and studied in [26]. Here we extend these concepts to involution comma-free and involution intercodes as follows. This section investigates these notions and algebraic properties of these codes. An *n-θ-intercode* of index m is a language $X \subseteq \Sigma^+$ such that every subset of X with at most n elements is a θ-intercode of index m. An n-θ-comma-free code is an n-θ-intercode of index one.

Proposition 21. *The class of 2-θ-comma-free codes is not closed under union, catenation, complement, catenation closure.*

Proof. The proof will be done by constructing some examples. We consider two languages $\{ab\}$ and $\{ba\}$ that are 2-θ-comma-free for an antimorphic θ mapping $a \mapsto b$. It is clear that the union $\{ab, ba\} = \{ab, ba\}$, the product $\{ab\}^2$ and the catenation closure $\{(ab)^+\}$ are not 2-θ-comma-free codes. Also, the class of 2-θ-comma-free codes are closed under intersection but not under union or complement.

Proposition 22. *If X is a 2-θ-comma-free code then X is θ-infix.*

Proof. Suppose X is not θ-infix then there exists $x, y \in X$ such that $x = a\theta(y)b$ which implies $\{x, y\}$ is not θ-infix and hence not θ-comma-free which is a contradiction. Hence X is θ-infix. □

Proposition 23. *Let $X \subseteq \Sigma^+$ be such that $X \cap \theta(X) = \emptyset$ and $\theta(\mathrm{PSuff}(X)) \cap \mathrm{PPref}(X) = \emptyset$.*

Then the following are equivalent.

1. *X is a 2-θ-comma-free code.*
2. *X is θ-infix and for $u, v \in \Sigma^+$, if $uv \in \theta(X)$ then $X \cap v\Sigma^* u = \emptyset$.*

Proof. Note that from Proposition 22 X is θ-infix. Let $u, v \in \Sigma^+$ such that $uv \in \theta(X)$. If $X' = \{\theta(uv), vxu\} \subseteq X$ for some $x \in \Sigma^*$, then $(vxu)^2 \in X'^2 \cap \Sigma^+ X' \Sigma^+$. This implies that X is not a 2-θ-comma-free code.

For the converse, let X be θ-infix. Suppose there exists $x, y \in X$ such that $\{x, y\}$ is not θ-comma-free then the either $xy = a\theta(x)b$ or $x^2 = a\theta(x)b$ or $y^2 = a\theta(x)b$ or $yx = a\theta(x)b$ for some $a, b \in \Sigma^+$. Since X is θ-infix, $\theta(x)$ is not a proper subword of x or y. Also note that $\theta(x) \neq x_2x_1$ for $x_1x_2 = x$ or $\theta(x) \neq y_2y_1$ for

$y_1y_2 = y$ since for all $uv \in \theta(X)$, $v\Sigma^*u \cap X = \emptyset$. Suppose $a\theta(x)b = xy$, then $\theta(x) = x_2y_1$ for $x = x_1x_2$ and $y = y_1y_2$. When θ is morphism, $x = \theta(x_2)\theta(y_1)$ and when θ is antimorphism, $x = \theta(y_1)\theta(x_2)$ both cases contradict our assumption that $\theta(\text{PSuff}(X)) \cap \text{PPref}(X) = \emptyset$. Hence X is a 2-θ-comma-free code.

Proposition 24. *X is a 3-θ-comma-free code if and only if X is a θ-comma-free code.*

Proposition 25. *If X is a k-θ-comma-free code then X is a i-θ-comma-free code for all $i \leq k$.*

Proof. Immediate. Note that X being k-θ-comma-free code does not imply X is i-θ-comma-free code for $i \geq k+1$ and $k \leq 2$. For example let $X = \{cbaa, baad, babb\}$ over the alphabet set $\Sigma = \{a, b, c, d\}$. Let θ be an antimorphic involution such that $a \mapsto b$ and $c \mapsto d$. It is easy to check that X is not θ-comma-free but X is 2-θ-comma-free.

Proposition 26. *X is a θ-intercode of index m if and only if X is a $(2m+1)$-θ-intercode of index m.*

Proof. Let X be $(2m+1)$-θ-intercode of index m. Suppose X is not θ-intercode of index m then there exists $x_1, x_2, ..., x_{m+1}, y_1, ..y_m \in X$ such that $x_1x_2...x_{m+1} = a\theta(y_1...y_m)b$ for some $a, b \in \Sigma^+$ which implies that X is not $(2m+1)$-θ-intercode of index m. The converse of the proof is immediate. □

Note that every θ-intercode of index m is an n-θ-intercode of index m for all $n \geq 1$. But for $n \leq 2m$ an n-θ-intercode of index m is not neccesarily a θ-intercode of index m. For example.....

Proposition 27. *If X is a 2-θ-comma-free code , then Xy and yX are 2-θ-comma-free code for all $y \in X$.*

Proof. Suppose Xy is not 2-θ-comma-free then there exists $\{x_1y, x_2y\} \subseteq Xy$ such that atleast one of the following happens:

$x_1yx_2y = a\theta(x_1y)b$ or $x_1yx_2y = a\theta(x_2y)b$ or $x_2yx_1y = a\theta(x_1y)b$ or $x_2yx_1y = a\theta(x_2y)b$ or $x_1yx_1y = a\theta(x_1y)b$ or $x_1yx_1y = a\theta(x_2y)b$ or $x_2yx_2y = a\theta(x_1y)b$ or $x_2yx_2y = a\theta(x_2y)b$.

Note that none of the $\theta(x_1)$ or $\theta(x_2)$ or $\theta(y)$ is a subword of x_1, x_2 or y since X is θ-infix. Also none of the $\theta(x_i)$ or $\theta(y)$ is a subword of x_iy or yx_i since X is a 2-θ-comma-free code. Suppose in $x_1yx_2y = a\theta(x_1y)b$ if $\theta(x_1)$ is a subword of yx_2 then either $\theta(y)$ is a subword of x_1 or x_2y which is a contradiction to the given assumption. Similarly we can show that yX is a 2-θ-comma-free code. □

Note that X being 2-θ-comma-free code does not imply X^n is 2-θ-comma-free code. For example let $X = \{ebb, dae, aac, bcb\}$. Let θ be a morphic involution such that $a \mapsto b$, $c \mapsto d$ and $e \mapsto e$. It is easy to check that X is 2-θ-comma-free code but X^2 is not since $ebbdae, aacbeb \in X^2$ with $ebbdaeaacbeb = e\theta(aacbeb)acbeb$.

6 Methods for Constructing Involution Codes

With the constructions in this section we show several ways to generate involution codes with "good" properties. Many authors have realized that in the design of DNA strands it is helpful to consider three out of the four bases. This was the case with several successful experiments [3,8,24]. It turns out that this, or a variation of this technique, can be generalized in such a way that codes with some of the desired properties can be easily constructed. Methods to construct θ-infix, θ-comma-free, θ-k-code and θ-subword-k-codes were provided in [14]. In this section, we concentrate on providing methods to generate θ-prefix, θ-suffix, θ-bifix, θ-outfix and θ-intercodes X such that X^+ has the same property. Some of these methods (Proposition 28) are in some sense generalizations of the idea of considering only three out of four bases. For each code X, the entropy of X^+ is computed. The entropy measures the information capacity of the codes, i.e., the efficiency of these codes when used to represent information.

Suppose that we have a source alphabet with p symbols each occurring with probability $s_1, s_2, ...s_p$. If $s_1 = 1$, then there is no information since we know what the message must be. If all the probabilities are different then for a symbol with low probability we get more information than for a symbol with high probability. Hence information is somewhat inversely related to the probability of occurrence. Entropy is the average information over the whole alphabet of symbols.

The standard definition of entropy of a code $X \subseteq \Sigma^+$ uses a probability distribution over the symbols of the alphabet of X (see [4]). However, for a p-symbol alphabet, the maximal entropy is obtained when each symbol appears with the same probability $\frac{1}{p}$. In this case the entropy essentially counts the average number of words of a given length as subwords of the code words [19]. From the Coding Theorem ([1]), it follows that $\{0,1\}^+$ can be encoded by X^+ with $\Sigma \mapsto \{0,1\}$ if the entropy of X^+ is at least $\log 2$ (see Theorem 5.2.5 in [23]). The codes for θ-comma-free, strictly θ-comma-free, and θ-k-codes designed in this section have entropy larger than $\log 2$ when the alphabet has $p = 4$ symbols. Hence, such DNA codes can be used for encoding bit-strings.

We start with the entropy definition as defined in [23].

Definition 2. *Let X be a code. The entropy of X^+ is defined by*

$$\hbar(X) = lim_{n\to\infty} \frac{1}{n} \log |\mathrm{Sub}_n(X^+)|.$$

If G is a deterministic automaton or an automaton with a delay (see [23]) that recognizes X^+ and A_G is the adjacency matrix of G, then by Perron-Frobenius theory A_G has a maximal positive eigen value $\bar{\mu}$ and the entropy of X^+ is $\log \bar{\mu}$ (see Chapter 4 of [23]). We use this fact in the following computations of the entropies of the designed codes. In [12], Proposition 16, authors designed a set of DNA code words that is strictly θ-comma-free. The following propositions shows that, in a similar way, we can construct codes with additional "good" properties.

In what follows we assume that Σ is a finite alphabet with $|\Sigma| \geq 3$ and $\theta : \Sigma \to \Sigma$ is an involution which is not identity. We denote by p the number of symbols in Σ.

Proposition 28. *Let $a \in \Sigma$ be such that $\theta(a) \neq a$. Let $X = \bigcup_{i=1}^{\infty} a^n(\Sigma \setminus \theta(a))^i a^n$ for a fixed integer $n \geq 1$. Then X and X^+ are both θ-prefix and θ-suffix. The entropy of X^+ is such that $\log(p-1) < \hbar(X^+) < \log(p)$.*

Proof. By Proposition 5 it is enough to show that X is strict θ-infix. Let $x, y \in X$ such that $u\theta(x)b = v$ for some $u, v \in \Sigma^*$. Then $u\theta(a^n w_1 a^n)v = a^n w_2 a^n$ for some $w_1, w_2 \in (\Sigma \setminus \{\theta(a)\})^i$ which implies $ub^n w_3 b^n v = a^n w_2 a^n$ where $\theta(a) = b$ which is not possible since $a \neq b$ and $b \neq \mathrm{Sub}(w_2)$. Therefore X is θ-strict-infix code and hence X^+ is both θ-prefix and θ-suffix.

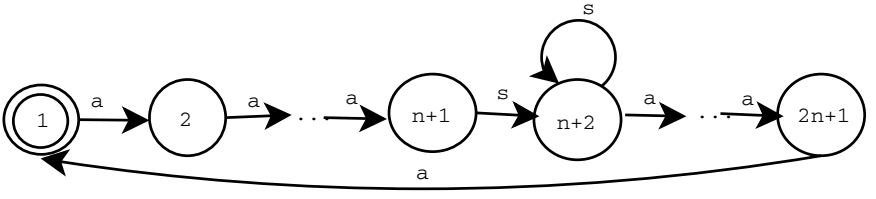

Fig. 3. Finite state automaton $\mathcal{A}$ that recognizes X^+ where $S = \Sigma \setminus \theta(a)$

Let $\mathcal{A} = (\mathcal{V}, \mathcal{E}, \lambda)$ be the automaton that recognizes X^+ where $V = \{1, ..., 2n+1\}$ is the set of vertices, $E \subseteq V \times \Sigma \times V$ and $\lambda : E \to \Sigma$ (with $(i, s, j) \mapsto s$) is the labeling function defined in the following way:

$$\lambda(i,s,j) = \begin{cases} a & \text{for } 1 \leq i \leq n, n+2 \leq i \leq 2n,\ j = i+1, \\ & \text{and } i = 2n+1, j = 1, \\ s & \text{for } i = n+1, n+2,\ j = n+2,\ s \in \Sigma \setminus \{\theta(a)\} \end{cases}$$

Then the adjacency matrix for $\mathcal{A}$ is a $(2n+1)\times(2n+1)$ matrix with ijth entry equal to the number of edges from vertex i to vertex j. Then the characteristic polynomial can be computed to be $\det(A-\mu I) = (-\mu)^{2n}(p-1-\mu)+(-1)^{2n}(p-1)$. The eigen values are solutions of the equation $\mu^{2n}(p-1) - \mu^{2n+1} + p - 1 = 0$ which gives $p - 1 = \mu - \frac{\mu}{\mu^{2n}+1}$. Hence $0 < \frac{\mu}{\mu^{2n}+1} < 1$, i.e., $p - 1 < \mu < p$. □

In the case of the DNA alphabet, $p = 4$ and for $n = 1$ the above characteristic equation becomes $\mu^3 - 3\mu^2 - 3 = 0$. The largest real value of μ is approximately 3.27902 which means that the entropy of X^+ is greater than $\log 2$.

Proposition 29. *Let $a, b \in \Sigma$ be such that for all $\theta(a) \neq \theta(b) \neq a \neq b$. Let $X = \bigcup_{i=1}^{\infty} a^n \Sigma^i b^n$ for a fixed integer $n \geq 1$. Then X and X^+ are θ-bifix and θ-outfix. The entropy of X^+ is such that $\log(p-1) < \hbar(X^+) < \log(p)$.*

In the case of the DNA alphabet, $p = 4$ and for $n = 1$ the above characteristic equation becomes $\mu^3 - 4\mu^2 - 4 = 0$. The largest real value of μ is approximately 4.22417 which means that the entropy of X^+ is greater than $\log 2$.

Proposition 30. *Choose distinct $a, b, c \in \Sigma$ such that $\theta(a) \neq b, c$, $\theta(a) \neq a$. Let $X = \bigcup_{i=1}^{\infty} a^n(\Sigma^{n-1}c)^i b^n$ for some $n \geq 2$. Then X and so X^+ are strictly θ-intercodes of index m for all $m \geq 1$. The entropy of X^+ is such that $\log(p^{\frac{n-1}{n}}) < \hbar(X^+) < \log((p^{n-1}+1)^{\frac{1}{n}})$.*

For the DNA alphabet, $p = 4$, and for $n = 2$, the above characteristic equation becomes $\mu^6 - 4\mu^4 - 4 = 0$. Solving for μ, the largest real value of μ is 2.05528. Hence the entropy of X^+ is greater than $\log 2$.

7 Concluding Remarks

In this paper we investigated theoretical properties of languages that avoided certain type of undesirable Watson-Crick bindings; θ-outfix codes, θ-intercodes, n-θ-intercodes and n-θ-comma-free codes. All these new concepts generalize classical notions of outfix codes, intercodes, n-intercodes and n-comma-free codes respectively. In addition, DNA word sets that are θ-outfix codes or θ-intercodes are of interest in the design of DNA computing experiments since such sets avoid unwanted hybridization Fig 1 and Fig 2. This paper We also developed certain methods to construct such sets of DNA code words with good properties and have calculated their informational entropy.

Acknowledgment. This work has been supported by NSERC and Canada Research Chair Grant for Lila Kari.

References

1. R.L. Adler, D. Coppersmith and M. Hassner, *Algorithms for sliding block codes -an application of symbolic dynamics to information theory*, IEEE Trans. Inform. Theory 29 (1983): 5-22.
2. E.B. Baum, *DNA Sequences useful for computation* unpublished article (1996).
3. R.S.Braich, N.Chelyapov, C.Johnson, P.W.K.Rothemund, L.Adleman, *Solution of a 20-variable 3-SAT problem on a DNA computer Science*, Science 19, Vol 296(5567) (2002) 499-502.
4. J. Berstel, D. Perrin, *Theory of Codes,* Academis Press, Inc. Orlando Florida, 1985.
5. 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.
6. 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) June7-10, (2004) 132-141.
7. 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.
8. 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.
9. 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.

10. T. Head, *Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors*, Bull. Math. Biology 49 (1987) 737-759.
11. 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.
12. 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.
13. N. Jonoska, D. Kephart, K. Mahalingam, *Generating DNA code words* Congressus Numernatium **156** (2002): 99-110.
14. N.Jonoska , K.Mahalingam and J.Chen, *Involution Codes: With Application to DNA Coded Languages*, Natural Computing, Vol 4-2(2005), 141-162.
15. L.Kari, *On insertion and deletion on formal languages*, Doctoral Dissertation in Mathematics, University of Turku, Finland.
16. L. Kari, S. Konstantinidis, E. Losseva and G. Wozniak, *Sticky-free and overhang-free DNA languages*, Acta Informatica 40 (2003): 119-157.
17. L.Kari, S.Konstantinidis and P.Sosik, *Bond-free Languages: Formalizations, Maximality and Construction Methods*, Preliminary Proceedings of DNA 10 June 7-10 (2004):16-25.
18. L.Kari, S.Konstantinidis, P.Sosik, *Preventing Undesirable Bonds between DNA Codewords* Preliminary Proceedings of DNA 10 June 7-10 (2004) 375-384.
19. M.S. Keane, *Ergodic theory an subshifts of finite type* , Ergodic theory, symbolic dynamics and hyperbolic spaces (ed. T.Edford, et.al.) Oxford Univ. Press, Oxford (1991): 35-70.
20. D.Kephart and J.Lefevre, *Codegen: The generation and testing of DNA code words*, Proceedings of IEEE Congress on Evolutionary Computation, June (2004): 1865-1873.
21. P.Leupold, *Partial Words for DNA Coding* Preliminary Proceedings of DNA 10 June 7-10 (2004) 26-35.
22. Z. Li, *Construct DNA code words using backtrack algorithm*, preprint.
23. D. Lind and B. Marcus, *An introduction to Symbolic Dynamics and Coding*, Cambridge University Press, Inc. Cambridge United Kingdom (1999).
24. Q. Liu et al., *DNA computing on surfaces*, Nature 403 (2000) 175-179.
25. 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.
26. H.J.Shyr, *Free Monoids and Languages*, Hon Min Book Company 2001.