Codes, Involutions, and DNA Encodings*

Lila Kari, Rob Kitto, and Gabriel Thierrin

Department of Computer Science, Univ. of Western Ontario
London, Ontario, N6A 5B7 Canada
{lila, kitto, gab}@csd.uwo.ca

If we knew what it was we were doing,
it would not be called research, would it?
(Albert Einstein)

1 Introduction

DNA computing as a field started in 1994 when Leonard Adleman solved a hard computational problem entirely by manipulations of DNA molecules in a test tube [1]. The premise behind DNA computing is that DNA is capable of storing information, while various laboratory techniques that operate on and modify DNA strands (called bio-operations in the sequel) can be used to perform computational steps. Most DNA computations consists of three basic stages. The first is encoding the problem using single-stranded or double-stranded DNA. Then the actual computation is performed by employing a succession of bio-operations [14]. Finally, the DNA strands representing the solution to the problem are detected and decoded. Because of the nature of the substrate in which the data is encoded, namely DNA strands, problems can occur at the encoding stage which would not occur in an electronic medium. In order to describe these problems and our attempts at solutions, we now briefly present some basic molecular biology notions and notations.

DNA (deoxyribonucleic acid) is found in every cellular organism as the storage medium for genetic information. It is composed of units called nucleotides, distinguished by the chemical group, or base, attached to them. The four bases, are *adenine, guanine, cytosine* and *thymine*, abbreviated as A, G, C, and T. The names of the bases are also commonly used to refer to the nucleotides that contain them. Single nucleotides are linked together end–to–end to form DNA strands. A short single-stranded polynucleotide chain, usually less than 30 nucleotides long, is called an *oligonucleotide*. The DNA sequence has a *polarity*: a sequence of DNA is distinct from its reverse. The two distinct ends of a DNA sequence are known under the name of the $5'$ end and the $3'$ end, respectively. Taken as pairs, the nucleotides A and T and the nucleotides C and G are said to be complementary. Two complementary single–stranded DNA sequences with

* Research partially supported by Grants R0504A01 and R2824A01 of the Natural Sciences and Engineering Research Council of Canada.

W. Brauer et al. (Eds.): Formal and Natural Computing, LNCS 2300, pp. 376–393, 2002.

opposite polarity are called *Watson/Crick (W/C) complements* and will join together to form a double helix in a process called *base-pairing*, or *hybridization* [14].

In most proposed DNA-based algorithms, the initial DNA solution encoding the input to the problem will contain some DNA strands which represent single *codewords*, and some which represent strings of catenated codewords. Several attempts have been made to address the issue of "good encodings" by trying to find sets of codewords which are unlikely to form undesired bonds with each other by hybridization [3], [9], [8]. For example genetic and evolutionary algorithms have been developed which select for sets of DNA sequences that are less likely to form undesirable bonds [4], [5]. [6] has developed a program to create DNA sequences to meet logical and physical parameters such as uniqueness, melting temperatures and G/C ratio as required by the user. [7] has addressed the issue of finding an optimal word design for DNA computing on surfaces. [12] has designed a software for constraint-based nucleotide selection. [10] has investigated encodings for DNA computing in virtual test tubes. [15] used combinatorial methods to calculate bounds on the size of a set of uniform code words (as a function of codeword length) which are less likely to mishybridize.

In this paper, we only address some of the various possible issues that might arise in DNA encodings, namely undesirable intramolecular and intermolecular hybridizations of the following types. Firstly, it is undesirable for any single DNA strand (representing a codeword or a string of codewords) to form a hairpin structure, which can happen if either end of the strand binds to another section of that same strand. Secondly, it is undesirable for any strand representing a codeword to bind to another one representing a string of one or more codewords. If such undesirable hybridizations occur, they will in practice render the involved DNA strands useless for the subsequent computational steps.

We present an initial investigation into the algebraic structure of *DNA-compliant* languages: languages consisting of codewords (DNA strands) that avoid some or all of the above mentioned undesirable bindings. It is hoped that this will lead to a deeper understanding of what is necessary for encoding information in DNA, and will ultimately assist in solving the difficult problem of "good DNA encodings". The paper is organized as follows. Section 2 introduces some basic definitions and notations. Section 3 formalizes the problems we address and defines the notion of a DNA compliant language: a language of codewords that avoid the above undesirable bindings. Section 4 studies the simpler notions of complement-compliance and mirror-image compliance needed to solve the general case of DNA compliance. Section 5 connects these particular cases to DNA compliance, generalizes the notion of DNA compliance, and investigates languages with this generalized property. Section 6 investigates generalizations of some classical coding theory notions (infix, prefix, suffix codes, density, residues, ideals) inspired by this extension of the notion of DNA compliance.

2 Basic Definitions and Notations

In this section we define the basic algebraic, formal language theory and coding theory notions needed in this paper. For further formal language theory notions the reader is referred to [11] and for further coding theory and algebraic notions to [18], [13], [11].

A mapping $\alpha : S \to S$ of a set S into S is *bijective* if it is both injective and surjective. Every bijective mapping has an inverse α^{-1} that is also bijective and $\alpha\alpha^{-1} = \alpha^{-1}\alpha = \epsilon$ where ϵ is the identity mapping.

Definition 2.1. *An involution $\theta : S \to S$ of S is a mapping such that $\theta^2 = \epsilon$.*

It follows then that an involution θ is bijective and $\theta = \theta^{-1}$.

The identity mapping is a trivial example of involution.

Given an involution, the relation ρ_θ defined by $u\rho_\theta v$ if and only if $u = v$ or $v = \theta(u)$ is an equivalence of S and every class has one or two elements.

The product of involutions is not necessarily an involution. For example take $S = \{a, b, c\}$ and the involutions α and β defined by:

$$\alpha(a) = b, \alpha(b) = a, \alpha(c) = c \ ; \ \beta(a) = c, \beta(b) = b, \beta(c) = a$$

Then:

$$\alpha\beta(a) = c, \alpha\beta(b) = a, \alpha\beta(c) = b.$$

It is clear that the product $\alpha\beta$ is no more an involution as, for example, $(\alpha\beta)^2(a) = b \neq a$.

However if the involutions α, β commute, i.e. if $\alpha\beta = \beta\alpha$, then their product $\alpha\beta$ is an involution because;

$$(\alpha\beta)(\alpha\beta) = (\alpha\beta)(\beta\alpha) = \alpha(\beta^2)\alpha = \alpha^2 = \epsilon.$$

Let X^* be the free monoid generated by the finite alphabet X, let 1 denote the neutral element, i.e. the empty word, and let $X^+ = X^* \setminus \{1\}$. The catenation of two words $u, v \in X^*$ is denoted by uv or by $u.v$ and consists of the juxtaposition of the words. A mapping $\alpha : X^* \to X^*$ is called a *morphism* (*anti-morphism)* of X^* if $\alpha(uv) = \alpha(u)\alpha(v)$ (respectively $\alpha(uv) = \alpha(v)\alpha(u)$) for all $u, v \in X^*$. A bijective morphism (anti-morphism) is called an *isomorphism (anti-isomorphism)* of X^*.

Remark that a morphism or an anti-morphism α of X^* is completely determined by the image $\alpha(X)$ of the alphabet X. If α is a morphism (anti-morphism) of X^* that is also an involution, then $\alpha(X) = X$ since α is an involution and therefore a bijective function, and $\alpha(a) \neq 1$ for all $a \in X$.

Examples.

(1) Let $\mu : X^* \to X^*$ be the mapping $\mu(u) = v$ defined by:

$$u = a_1 a_2 \cdots a_k, \ \ v = a_k \cdots a_2 a_1, \ \ a_i \in X, 1 \leq i \leq k.$$

The word v is called the *mirror image* of u. Since $\mu^2 = \epsilon$, μ is an *involution* of X^*, and will be called the *mirror-involution* or simply the *m-involution* of X^*.

The m-involution of X^* is not a morphism but an *anti-morphism* of X^* because $\mu(uv) = \mu(v)\mu(u)$ for $u, v \in X^*$.

(2) A mapping $\gamma : X \to X$ of X into X can be extended to a morphism of X^* by $\gamma(a_1a_2 \cdots a_k) = \gamma(a_1)\gamma(a_2) \cdots \gamma(a_k)$. If furthermore $\gamma^2 = \epsilon$, then this morphism of X^* is also an involution of X^*.

In the general case, an involution is just a mapping whose square is the identity and as such is not closely related to the algebraic structure of X^*. We give an example of an involution π of X^* that is neither a morphism nor an anti-morphism.

A *primitive word*, [18], w over X^+ is a word with the property that $w = u^p$ for some $p \geq 1$ and $u \in X^+$ implies $p = 1$ and $u = w$.

Let $u \in X^*$. If $u = 1$, then $\pi(u) = 1$. If $u \neq 1$, then $u = p^k$ where p is a primitive word and k a positive integer. If k is odd, then we define $\pi(u) = p^{k+1}$, and if k is even we define $\pi(u) = p^{k-1}$. Then clearly π is an involution which is not a morphism or an anti-morphism. For example take $u = a \in X$ and $v = a^2$. Then $\pi(uv) = \pi(a^3) = a^4$, $\pi(u) = a^2$ and $\pi(v) = a$. Hence $\pi(uv) \neq \pi(u)\pi(v)$ and $\pi(uv) \neq \pi(v)\pi(u)$.

Note. *The involutions considered in this paper being always either* morphisms *or an* anti-morphisms *of X^*, from now on an involution will always be implicitely assumed to be morphic or antimorphic.*

3 DNA-related Types of Compliance

The W/C complement of a single strand of DNA is the strand that would anneal to it in the process of base-pairing. For example, the Watson-Crick complement of $5' - AAACC - 3'$ is $3' - TTTGG - 5'$ which is the same as $5' - GGTTT - 3'$. If we consider the DNA alphabet $\Delta = \{A, C, G, T\}$, by convention, a word $a_1a_2 \ldots a_n \in \Delta^*$, $n \geq 0$, represents the single DNA strand $5' - a_1a_2 \ldots a_n - 3'$ in the 5' to 3' orientation.

Three involutions can be defined on Δ^*. The *m-involution μ (mirror-involution)*, which is an anti-morphism, has been formally defined in Section 2. The mapping $\gamma : \Delta \to \Delta$ defined by:

$$\gamma(A) = T, \gamma(T) = A, \gamma(C) = G, \gamma(G) = C,$$

associates to each nucleotide its complement and can be extended in the usual way to a morphism of Δ^* that is also an involution of Δ^*. This involution γ will be called the *complementary-involution* or simply the *c-involution of Δ^**.

It is easy to see that γ and μ commute, i.e. $\gamma\mu = \mu\gamma$. Hence $\gamma\mu$, denoted by τ is also an involution of Δ^*, called the *W/C involution*. Furthermore τ is an anti-morphism. Note that, for $w \in \Delta^*$, $\tau(w)$ represents the W/C complement of w. Using our notations and convention, for example, $\tau(AAACC) = GGTTT$.

The following notations will also be used:

$$\gamma(u) = \bar{u},\ \mu(u) = \tilde{u},\ \gamma\mu(u) = \tau(u) = \overleftarrow{\bar{u}}.$$

Let us assume that for encoding the input information to a computation, we use a set of single DNA strands, i.e., a finite set $K \subseteq \Delta^*$ of words, called also *codewords*. Due to the W/C complementarity, if the codewords are not carefully designed, they will stick to each other forming undesired double-stranded structures.

Some of the situations that should be avoided when designing the codewords are the following.

Situation 1. A codeword that sticks to itself by intramolecular hybridization: $uv \in K^*$, $u, v \in \Delta^*$ with

(i) $\overleftarrow{v}$ being a subword of u (Figure 1);

(ii) $\overleftarrow{u}$ being a subword of v (Figure 2).

Situation 2. Two codewords that stick to each other by intermolecular hybridization: $u, v \in K$ with $\overleftarrow{v}$ being a subword of u (Figure 3).

Situation 3. A codeword that sticks to the catenation of two other codewords, by intermolecular hybridization: $u_1, u_2, v \in K$ with $\overleftarrow{v}$ being a subword of u_1u_2 (Figure 4).

Remark that *Situation 2* and *Situation 3* are closely related. Note also that if we consider the simplified version of *Situation 1* where $u, v \in K$ instead of $u, v \in \Delta^*$ then *Situation 1* becomes *Situation 2*. In view of these remarks, in this paper we will be dealing mainly with languages avoiding *Situation 2*.

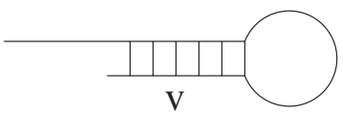

Fig. 1. Intramolecular hybridization leading to unwanted hairpin formations: $uv \in K^*$, $\overleftarrow{v}$ being a subword of u.

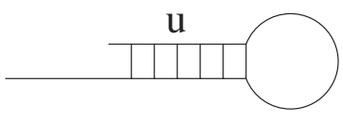

Fig. 2. Intramolecular hybridization leading to unwanted hairpin formations: $uv \in K^*$, $\overleftarrow{u}$ being a subword of v.

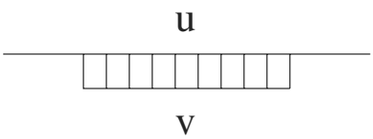

Fig. 3. Intermolecular hybridization between two codewords: $u, v \in K$, $\overleftarrow{v}$ being a subword of u.

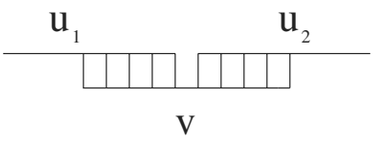

Fig. 4. Intermolecular hybridization between a composite codeword u_1u_2 and a third codeword v: $u_1, u_2, v \in K$, $\overleftarrow{v}$ being a subword of u_1u_2.

A simple (but not necessarily practical) solution to the above problems is the following. Take a language K over Δ such that $K^+ \subseteq \{A, C\}^+$. Then for every $u \in K^+$, $\overleftarrow{u} \notin K^+$. This means that there is no possibility for the occurences of the above *Situations 1, 2* or *3*.

The following definition formalizes a property of a language K which, if satisfied, ensures that the codewords of K avoid Situation 2 (Fig.3).

Definition 3.1. *A language $K \subseteq \Delta^*$ is called DNA compliant (DNA prefix-compliant, DNA suffix-compliant) iff:*

$$x\overleftarrow{v}\,y = u, u, v \in K \;\Rightarrow\; x = y = 1 \;\; (\overleftarrow{v}\,y = u \;\Rightarrow\; y = 1, \;\; y\overleftarrow{v} = u \;\Rightarrow\; y = 1).$$

In the following we shall sometimes refer to DNA compliant, DNA prefix-compliant, DNA suffix-compliant languages shortly as compliant, p-compliant, s-compliant languages.

Note that the preceding definition does not exclude the obviously undesirable case where two codewords in the language are exact W/C complements of each other. To eliminate this situation as well, we introduce the stronger notion of a strictly DNA compliant language.

Definition 3.2. *A language $K \subseteq \Delta^*$ is called strictly DNA compliant if it is DNA compliant and furthermore, for all $u, v \in K$ we have $u \neq \overleftarrow{v}$.*

4 The Complementarity and Mirror Involutions

Before addressing the general problem of DNA compliance that involves the W/C involution $\tau = \gamma\mu$, in this section we address separately the simpler cases involving the complementary involution γ and the mirror involution μ.

Definition 4.1. *A language $K \subseteq \Delta^*$ is called complementarity-compliant or c-compliant (respectively complementarity prefix compliant or cp-compliant) if $u \in K, x\bar{u}y = v \in K$ (resp. $\bar{u}y = v \in K$) implies $x = y = 1$.*

Definition 4.2. *A language $K \subseteq \Delta^*$ is called strictly c-compliant (cp-compliant) if it is c-compliant (cp-compliant) and for all $u, v \in K$ we have that $u \neq \bar{v}$.*

Let $\bar{K}^* = \{\bar{u} | u \in K^*\}$. Clearly $\bar{K}^*$ is a submonoid of Δ^* that is isomorphic to K^*.

Proposition 4.1. *A c-compliant (cp-compliant) language $K \subseteq \Delta^*$ is strictly c-compliant (cp-compliant) if and only if $K^* \cap \bar{K}^* = \{1\}$.*

Proof. Indeed, suppose K is a strictly c-compliant language and $K^* \cap \bar{K}^* \neq \{1\}$. Since $1 \in K^*$ and $1 \in \bar{K}^*$, there must be some $x \in \Delta^+$ such that $x \in K^*$ and $x \in \bar{K}^*$. Furthermore, $x \in \bar{K}^*$ implies $\bar{x} \in K^*$, from the definition of $\bar{K}^*$ and because γ is an involution.

Let $x = uv$, $u \in K$, $v \in K^*$. Let $\bar{x} = wz$, $w \in K$, $z \in K^*$. Suppose $|u| < |w|$. Then $\exists r \in \Delta^+$ such that $ur = \bar{w}$ (in fact, r will be a prefix of v). Since $r \neq 1$, K is not *cp*-compliant, and thus not c-compliant.

If $|w| < |u|$, the argument is the same. If $|w| = |u|$, then $w = \bar{u}$, and although K could be c-compliant, it is not *strictly* c-compliant. Thus $K^* \cap \bar{K}^* = \{1\}$.

Suppose K is a c-compliant language with $K^* \cap \bar{K}^* = \{1\}$ and that K is not strictly c-compliant. Then there is a $v \in \Delta^+$ such that v, $\bar{v} \in K$. But then v, $\bar{v} \in \bar{K}$, and thus $\{v, \bar{v}\} \subseteq K^* \cap \bar{K}^*$ - a contradiction.

The proof is similar for *cp*-compliant languages. ♣

Note that $K^* \cap \bar{K}^* = \{1\}$ does *not* imply that K is c-compliant, as shown by the example $K = \{ACC, TGGC\}$.

Proposition 4.2. *Let $K \subseteq \Delta^*$ be c-compliant language. Then $ux = \bar{v}y$, $u, v \in K$ implies $x = y$ and hence $u = \bar{v}$.*

Proof. If u is a prefix of $\bar{v}$, then $\bar{v} = ur$, $v = \bar{u}\bar{r}$. Since K is a *cp*-compliant language, then $\bar{r} = 1$, i.e. $r = 1$ and $u = \bar{v}$. If $\bar{v}$ is a prefix of u, then $u = \bar{v}s$ and similarly $s = 1$ and $u = \bar{v}$. ♣

Proposition 4.3. *If not empty, the intersection of c-compliant (cp-compliant) languages is a c-compliant (cp-compliant) language.*

Proof. Immediate. ♣

Similar results hold for the case of strictly c-compliant (*cp*-compliant) languages.

During computations, encodings for two different problems might be combined by, for example, catenating codewords from the two languages used for encoding the inputs of the two problems. The following proposition shows that this process preserves the property of c-compliance.

Proposition 4.4. *The catenation $K = K_1K_2$ of (strictly) cp-compliant languages $K_1 \subseteq \Delta^*$ and $K_2 \subseteq \Delta^*$ is a (strictly) cp-compliant language.*

Proof. Let $u \in K$ and $\bar{u}x \in K$. Then $u = u_1u_2$ with $u_1 \in K_1$, $u_2 \in K_2$, $\bar{u} = \bar{u}_1\bar{u}_2$ and $\bar{u}_1\bar{u}_2x = c_1c_2$, $c_1 \in K_1, c_2 \in K_2$.

If $|\bar{u}_1| \leq |c_1|$, then $\bar{u}_1y = c_1$ for some $y \in \Delta^*$. Since K_1 is *cp*-compliant, $y = 1$ and $\bar{u}_1 = c_1$. Therefore $\bar{u}_2x = c_2$ and $x = 1$ since K_2 is also *cp*-compliant.

If $|\bar{u}_1| > |c_1|$, then $\bar{u}_1 = c_1z$ for some $z \in \Delta^*$. This implies $u_{_1} = \bar{\bar{u}}_1 = \bar{c}_1\bar{z}$. Since $u_1, c_1 \in K_1$ and K_1 is *cp*-compliant, we have $\bar{z} = 1$, $z = 1$ and hence $\bar{u}_1 = c_1$, a contradiction.

Therefore $\bar{u}_1\bar{u}_2x = c_1c_2$ implies $x = 1$ and hence K_1K_2 is a *cp*-compliant language.

For the case of strict compliance, we have to show that $\bar{u}_1\bar{u}_2 \neq c_1c_2$. Suppose that $\bar{u}_1\bar{u}_2 = c_1c_2$. Then, as above, either $\bar{u}_1y = c_1$ or $\bar{u}_1 = c_1z$. Since both K_1 and K_2 are strictly compliant, this is impossible. ♣

Proposition 4.5. *Every (strictly) cp-compliant language can be embedded in a maximal (stricly) cp-compliant language.*

Proof. Suppose that $\Gamma = \{K_i \subseteq \Delta^* | i \in I\}$ is a chain of *cp*-compliant languages:

$$\cdots \subseteq K_j \subseteq \cdots \subseteq K_k \subseteq \cdots$$

Let $K = \bigcup_{i\in I} K_i$. It is easy to see that K is also a *cp*-compliant language. The proposition follows then from Zorn's Lemma. The proof is similar for the case of stricly *cp*-compliant languages. ♣

A language $L \subseteq X^*$ is called *dense* (right dense), [18], if for every $u \in X^*$, there exist $x, y \in X^*$ such that $xuy \in L (uy \in L)$. In the following we define an analogous notion, that of complementarity density. Proposition 4.6 then proves that a sufficient condition for a strictly *cp*-compliant language $K \subseteq \Delta^*$ to be maximal is that K^* be right *c*-dense.

Definition 4.3. *A language $K \subseteq \Delta^*$ is said to be right c-dense if for every $u \in \Delta^*$ there is $x \in \Delta^*$ such that $\bar{u}x \in K$.*

Proposition 4.6. *Let $K \subseteq \Delta^*$ be a stricly cp-compliant language. If K^* is right c-dense, then K is a maximal stricly cp-compliant language.*

Proof. Suppose K is not maximal. Then there is $v \notin K$ such that $T = K \cup \{v\}$ is a strictly *cp*-compliant language. The right c-density implies the existence of $x \in \Delta^*$ such that $\bar{v}x \in K^*$, i.e. $\bar{v}x = u_1u_2 \cdots u_k$, $u_i \in K$. This implies either $\bar{v} = u_1y$ or $u_1 = \bar{v}z$, i.e. $\bar{u}_1\bar{y} = v \in T$ or $\bar{v}z = u_1 \in T$. Since T is a *cp*-compliant language, then either $\bar{y} = 1, \bar{u}_1 = v$ or $z = 1, \bar{v} = u_1$, a contradiction because of the strict compliance of T. ♣

The converse of Proposition 4.6 is not true as shown by the following example. Let $K = A\Delta^* \cup C\Delta^*$. This language over Δ is strictly *cp*-compliant and it is easy to see that it is maximal. Clearly the language K^* is not right *c*-dense.

Recall that, [18], a language $L \subseteq X^*$ is called *right-unitary* if $u \in L^*$, $ux \in L^*$ imply $x \in L^*$. The following definition is analogous to that of a right-unitary language and Proposition 4.7 relates the notions of right *c*-unitary and *cp*-compliance. It turns out that, given a right *c*-unitary language K^*, where $K \subseteq \Delta^*$, one can construct a *cp*-compliant language $K' \subseteq \Delta^*$.

Definition 4.4. *A language $K \subseteq \Delta^*$ is called right c-unitary if $u \in K^*$ and $\bar{u}x \in K^*$* imply *$x \in K^*$.*

Example. Let $K = A\Delta^* \cup C\Delta^*$. Then K^+ is right c-unitary. Indeed suppose that $u, \bar{u}x \in K^+$. Every word in K^+ starts either with A or C and hence $\bar{u}x = Ty$ or Gy for some $y \in \Delta^*$, which is impossible. Hence $K = K^+$ is right c-unitary by default.

Proposition 4.7. *Consider $K \subseteq \Delta^*$ and let*

$$K' = K^+ \setminus [\bar{K}^+ K^+]$$

If K^* is right c-unitary then K' is cp-compliant.

Proof. Let $u \in K'$, $\bar{u}x \in K' \subseteq K^*$. As K^* is right c-unitary we have that $x \in K^*$, therefore

$$x = x_1 x_2 \dots x_m, x_i \in K, 1 \leq i \leq m.$$

The word $u \in K' \subseteq K^*$ and therefore $\bar{u} \in \bar{K}^*$. If any of x_i, $1 \leq i \leq m$ would not be the empty word, $\bar{u}x$ would belong to $\bar{K}^+K^+$ which would contradict the assumption about $\bar{u}x$. We can therefore conclude that $x_i = 1$ for all $1 \leq i \leq m$ which implies that K' is cp-compliant. ♣

In a similar manner to the c-involution, one can use the mirror involution to define m-compliant languages. A language $K \subseteq \Delta^*$ is called *m-compliant* (*mp-compliant, ms-compliant*) if $u \in K, x\tilde{u}y = v \in K (\tilde{u}y = v \in K, x\tilde{u} = v \in K)$ imply $x = y = 1$. If we add the condition that for any $u, v \in K$, $\tilde{u} \neq v$, then K will be *strictly m-compliant*. Informally, in an m-compliant language, the mirror image of a codeword cannot be a proper subword of another codeword. In a *strictly* m-compliant language, the mirror image of a codeword cannot be a subword of another codeword.

Example: $K = \{GGGA, CCG, TTAAA\}$ is strictly m-compliant, whereas $K' = \{GGGA, AGG\}$ is not m-compliant, and $K'' = \{GGGA, AGGG\}$ is m-compliant but not *strictly* m-compliant.

Using this definition of m-compliance, properties of m-compliant languages can be explored.

In contrast to the c-compliant case, the catenation $K = K_1K_2$ of m-compliant languages over Δ is not necessarily an m-compliant language. Indeed, let $K_1 = \{CTG, TA\}$ and $K_2 = \{AT, GT\}$. Both K_1 and K_2 are m-compliant (in fact they are strictly m-compliant), but $\{CTGAT, TAGT\} \subset K$, so K is not m-compliant. Note that this example also demonstrates that the catenation of two ms-compliant languages is not necesarily ms-compliant.

A language consisting of a single codeword can fail to be strictly m-compliant. An example is $K = \{TAAT\}$. In contrast, any single-codeword language (except the empty word) will be strictly c-compliant.

A language which is both c-compliant and m-compliant will not necessarily be DNA compliant. Take for example $K = \{CGT, ACGA\}$. Then K is both m- and c- compliant, but is not DNA compliant. However, m- and c- compliance are still *related* to DNA compliance. Indeed in Section 5 we prove results that connect the three different types of compliance determined by two different involutions and their product, respectively. In particular, Corollary 5.1 gives a sufficient condition

(involving the notions of c-compliance and m-compliance) for a language to be DNA compliant.

5 From DNA Compliance to Involution-Compliance

This section addresses the connections between the notions of c-compliance, m-compliance and the DNA compliance which was the original motivator of this study. Recall that the notions of c-compliance, m-compliance and DNA compliance were related respectively to the complementarity involution γ, the mirror involution μ and the Watson/Crick involution τ which is the product between the two, i.e., $\tau = \gamma\mu$. Instead of proving results related to the particular DNA-related involutions, we generalize from the DNA alphabet Δ to an arbitrary finite alphabet X and extend the notions of compliance to refer to arbitrary (morphic or anti-morphic) involutions of X^*. All the results obtained will have as corollaries results pertaining to the DNA-related involutions. In particular, Corollary 5.1 gives sufficient conditions for a c-compliant language to be DNA compliant, and for an m-compliant language to be DNA compliant.

Definition 5.1. *Let θ be an involution of X^*. A language $L \subseteq X^*$ is said to be θ-compliant (θ-p-compliant, θ-s-compliant) if:*

$$u, x\theta(u)y \in L \ \ (\theta(u)y \in L, x\theta(u) \in L) \ \ \Rightarrow \ \ x = y = 1.$$

The above condition prevents the image under θ of a word $u \in L$ to be a proper subword (prefix, suffix) of a word in L.

In order to eliminate also the case of two words $u, v \in L$ with $u = \theta(v)$, we strengthen the definition as follows.

Definition 5.2. *Let θ be an involution of X^* and $L \subseteq X^*$. A language L is said to be strictly θ-compliant (θ-p-compliant, θ-s-compliant) if it is θ-compliant (θ-p-compliant, θ-s-compliant) and, in addition, for each $u, v \in L$ we have that $u \neq \theta(v)$.*

Remark that if $\theta = \epsilon$, the identity involution, then the θ-compliant languages are respectively the infix, prefix and suffix codes. A *prefix code (suffix code)*, [18], [13], is a nonempty language $A \subseteq X^+$ with the property that $A \cap AX^+ = \emptyset$ ($A \cap X^+A = \emptyset$). An *infix code* is a language A with the property that for all $x, y, u \in X^*$ we have that $xuy \in L$ and $u \in L$ together imply $x = y = 1$.

In general θ-compliant languages are not infix codes. Let $L = \{A, C, AC\} \subseteq \Delta^*$. L is not an infix code, but a τ-compliant language relatively to the W/C involution τ.

An infix code is in general not θ-compliant. For example, the language $\{GG, ACCA\} \subseteq \Delta^*$ is an infix code, but $\tau(GG) = CC$ and $ACCA = A\tau(GG)A$, and hence the language is not τ-compliant.

Proposition 5.1. *If L is a θ-compliant language of X^* such that $\theta(L) \subseteq L$, then L is an infix code.*

Proof. Let $xuy \in L$ with $u \in L$. Since the involution θ is bijective, there is $v \in X^*$ such that $\theta(v) = u$ and hence $x\theta(v)y \in L$. Since $\theta(u) = v$ and $\theta(L) \subseteq L$, then $v \in L$. Because L is θ-compliant, $x = y = 1$ and therefore L is an infix code. ♣

The following results make the connection between c-compliance, m-compliance and DNA compliance.

Proposition 5.2. *Let θ_1 and θ_2 be two commuting involutions of X^*. If $L \subseteq X^*$ is a language such that $L \cup \theta_1(L)$ is θ_2-compliant, then L is $\theta_1\theta_2$-compliant.*

Proof. Let $\theta = \theta_1\theta_2$ and suppose that $L \cup \theta_1(L)$ is θ_2-compliant but is not θ-compliant. Then there exist $u, v \in L$ such that $u = x\theta(v)y$ with $xy \neq 1$. Let $T = \{u, v, \theta_1(u), \theta_1(v)\}$. Since $T \subseteq L \cup \theta_1(L)$, then T must be θ_2-compliant. From $u = x\theta(v)y$ follows $\theta_1(u) = \theta_1(x\theta(v)y)$.

If θ_1 is a morphism, then:

$$\theta_1(u) = \theta_1(x)\theta_1(\theta_1\theta_2(v))\theta_1(y) = \theta_1(x)\theta_2(v)\theta_1(y)$$

with $\theta_1(u), v \in T$. Since T is θ_2-compliant, this implies $\theta_1(x) = \theta_1(y) = 1$ and therefore $x = y = 1$, a contradiction with $xy \neq 1$. Hence L is θ-compliant.

If θ_1 is an antimorphism, then, since $\theta_1\theta_2 = \theta_2\theta_1$:

$$\theta_1(u) = \theta_1(y)\theta_1(\theta_1\theta_2(v))\theta_1(x) = \theta_1(y)\theta_2(v)\theta_1(x)$$

with $\theta_1(u), v \in T$. As above, we get a contradiction, hence L is θ-compliant. ♣

Corollary 5.1. *Let $L \subseteq \Delta^*$. If $L \cup \gamma(L)$ is μ-compliant or if $L \cup \mu(L)$ is γ-compliant, then L is DNA compliant, i.e. $\gamma\mu$-compliant.*

Furthermore, $K = \{AGT, CCCG\}$ is a DNA-compliant language for which $K \cup \tilde{K}$ is c-compliant and $K \cup \bar{K}$ is m-compliant, which means the sets of languages $\{K | K \cup \tilde{K}$ is c-compliant $\}$ and $\{K | K \cup \bar{K}$ is m-compliant $\}$ are non-empty subsets of the set of DNA-compliant languages.

Proposition 5.3. *Let θ be an involution of X^*. Then $L \subseteq X^*$ is θ-compliant iff $\theta(L)$ is θ-compliant.*

Proof. If $u, x\theta(u)y \in \theta(L)$, then $\theta(u), \theta(x)u\theta(y) \in L$ if θ is a morphism or $\theta(u)$,$\theta(y)u\theta(x) \in L$ if θ is an antimorphism. In both cases, since L is θ-compliant, we have $\theta(x) = \theta(y) = 1$ and hence $x = y = 1$. Therefore $\theta(L)$ is θ-compliant.

Conversely, if $\theta(L)$ is θ-compliant, then, from the first part of the proof, it follows that $L = \theta(\theta(L))$ is also θ-compliant. ♣

The union of θ-compliant languages is not in general a θ-compliant language. For example let Δ be the DNA alphabet and τ be the W/C involution. Then $L_1 = \{A, AC\}$ and $L_2 = \{T, TG\}$ are both τ-compliant. However their union $L = \{A, T, AC, TG\}$ is not τ-compliant, because $A \in L$, $TG = \theta(A)G \in L$ with $G \neq 1$.

If L is θ-compliant, then any subset T of L is also θ-compliant. Indeed $u, x\theta(u)y \in T$ implies $u, x\theta(u)y \in L$ and hence $x = y = 1$.

The following generalizes Proposition 4.4.

Proposition 5.4. *If the involution θ of X^* is a morphism, then the catenation $L = L_1L_2$ of θ-compliant (strictly θ-compliant) languages $L_1 \subseteq X^*$ and $L_2 \subseteq X^*$ is a θ-compliant (strictly θ-compliant) language.*

Proof. Let $u \in L$ and $x\theta(u)y \in L$. Then $u = u_1u_2$ with $u_1 \in L_1$, $u_2 \in L_2$.

Since θ is a morphism, $\theta(u) = \theta(u_1)\theta(u_2)$ and $x\theta(u_1)\theta(u_2)y = c_1c_2$, $c_1 \in L_1, c_2 \in L_2$.

If $|x\theta(u_1)| \leq |c_1|$, then $x\theta(u_1)z = c_1$ for some $z \in X^*$. Since L_1 is θ-compliant, then $x = z = 1$ and $\theta(u_1) = c_1$. Therefore $\theta(u_2)y = c_2$ and $y = 1$ since L_2 is θ-compliant.

If $|x\theta(u_1)| > |c_1|$, then $|\theta(u_2)y| < |c_2|$ and $z\theta(u_2)y = c_2$. Using a similar argument as above, we get $z = y = 1$.

Hence $u \in L$ and $x\theta(u)y \in L$ implies $x = y = 1$, i.e. L is θ-compliant.

For the case of strict compliance, we have to show that $\theta(u_1u_2) = \theta(u_1)\theta(u_2) \neq c_1c_2$. Suppose that $\theta(u_1)\theta(u_2) = c_1c_2$. Then, as above either $\theta(u_1)z = c_1$ or $z\theta(u_2) = c_2$ which is impossible because of the strict compliance of both L_1 and L_2. ♣

If the involution θ is an anti-morphism of X^*, then the catenation of θ-compliant languages is not in general a θ-compliant language. Indeed, let τ be the W/C involution of Δ^*. The languages $\{CTG, AT\}$ and $\{AT, CA\}$ are both τ-compliant. However $L = \{CTGAT, ATCA\}$ which is included in their catenation, is not τ-compliant.

If θ is an involution of X^*, then a submonoid $S \subseteq X^*$is said to be *right θ-unitary* if $u, \theta(u)x \in S$ implies $x \in S$.

The following proposition generalizes Proposition 4.7.

Proposition 5.5. *Let $L \subseteq X^+$ and let θ be an involution of X^*. Let*

$$T = L^+ \backslash (\theta(L)^+ L^+)$$

If L^* is right θ-unitary, then T is a θ-p-compliant language.

Proof. Let $u, \theta(u)x \in T \subseteq L^+$. Since L^* is right θ-unitary, $x \in L^*$. Suppose $x \neq 1$. Then:

$$x = x_1x_2 \cdots x_m,\ x_i \in L, 1 \leq i \leq m.$$

The word $u \in L^+$and therefore $\theta(u) \in \theta(L^+)$. If any of x_i, $1 \leq i \leq m$, would not be the empty word, $\theta(u)x$ would belong to $\theta(L)^+L^+$ which would contradict the assumption about $\theta(u)x$. We can therefore conclude that $x_i = 1$ for all i, $1 \leq i \leq m$, which implies that T is θ-p-compliant. ♣

Proposition 5.6. *Let θ be an involution and a morphism of X^* and let $L \subseteq X^*$ be a nonempty language. If L is a θ-p-compliant language, then L^* is a right θ-unitary submonoid of X^*.*

Proof. Suppose that $u, \theta(u)x \in L^*$. If θ is a morphism of X^*, then:

$$u = u_1u_2\cdots u_k,\ u_i \in L, \theta(u)x = \theta(u_1)\theta(u_2)\cdots\theta(u_k)x = v_1v_2\cdots v_r, v_j \in L$$

Either $|\theta(u_1)| \leq |v_1|$ or $|\theta(u_1)| > |v_1|$.

In the first case, we have $v_1 = \theta(u_1)z$ for some $z \in X^*$. Since L is θ-p-compliant and $u_1, v_1 \in L$, then $z = 1$ and $\theta(u_1) = v_1$.

In the second case, we have $\theta(u_1) = v_1z$ with $z \neq 1$. But, as above, $u_1, v_1 \in L$ and the compliance condition implies $z = 1$, and $\theta(u_1) = v_1$.

Therefore $\theta(u_1) = v_1$ and by cancellation:

$$\theta(u_2)\cdots\theta(u_k)x = v_2\cdots v_r$$

This reduction can be extended until we get $x = 1$ or $x = v_{k+1}\cdots v_r$ and hence $x \in L^*$, i.e. L^* is right θ-unitary. ♣

The following generalizes Proposition 4.1.

Proposition 5.7. *Let θ be a morphic involution. A θ-compliant (θ-p-compliant) language $L \subseteq X^*$ is strictly θ-compliant (strictly θ-p-compliant) if and only if $L^* \cap \theta(L)^* = \{1\}$.*

Proof. The proof given for the complementarity involution can be extended to this general case in the following way.

Suppose L is strictly θ-compliant and let $u \in L^* \cap \theta(L)^*$, $u \neq 1$. Then $u = u_1x_1 = \theta(u_2x_2) = \theta(u_2)\theta(x_2)$ with $u_1, u_2 \in L$, $u_1, u_2 \neq 1$, $x_1, x_2 \in L^*$.

If $|u_1| < |\theta(u_2)|$, then $\theta(u_2) = u_1x$, $u_2 = \theta(u_1)\theta(x)$ with $x \neq 1$ and hence $\theta(x) \neq 1$, a contradiction since L is θ-compliant.

If $|u_1| \geq |\theta(u_2)|$, then $u_1 = \theta(u_2)x$. This implies $x = 1$ and $u_1 = \theta(u_2)$. Hence $u_2, \theta(u_2) \in L$, a contradiction with the strictness of L.

Therefore $L^* \cap \theta(L)^* = \{1\}$.

Suppose now L to be a θ-compliant language with $L^* \cap \theta(L)^* = \{1\}$. If L is not strictly θ-compliant, then there is a word $u \in X^+$ such that $u, \theta(u) \in L$. This implies $u, \theta(u) \in \theta(L)$. Thus, $\{u, \theta(u)\} \subseteq L^* \cap \theta(L)^*$, a contradiction. ♣

6 Maximality, Density, Residue, and Ideals

We have seen that in the particular case when θ is the indentity involution, θ-compliant (θ-*p*-compliant, θ-*s*-compliant) languages coincide with the classical infix codes (prefix codes, suffix codes). This suggests that other well-known notions from coding theory, like density, residue and ideals can be generalized from the particular case of the identity involution to the case of an arbitrary involution. This section investigates such generalizations and their properties.

The following proposition generalizes Proposition 4.5.

Proposition 6.1. *Let θ be an involution of X^*. Every θ-compliant (θ-p-compliant) language L can be embedded in a maximal θ-compliant (θ-p-compliant) language.*

Proof. The proof follows by showing first that the union of languages of any chain of θ-compliant (θ-p-compliant) languages containing L has the same property. Then, by Zorn's Lemma, it follows that the family of θ-compliant (θ-p-compliant) languages in X^* containing L has a maximal element. ♣

Proposition 6.2. *Let θ be an involution of X^*. Every strictly θ-compliant (θ-p-compliant) language L can be embedded in a strictly maximal θ-compliant (θ-p-compliant) language.*

Proof. Similar to the proof of the previous proposition. ♣

If $L \subseteq X^*$, let $Lg(L) = max\{|u| \mid u \in L\}$. It is known (see [18]) that, for every finite prefix code P with $Lg(P) = n$, there exists a maximal prefix code P' such that $P \subseteq P'$ and $Lg(P') = n$.

This result which is true for ϵ-p-compliant languages, cannot in general be extended to every θ-involution as shown by the following example.

Example. Let $X = \{a, b\}$ and the morphic involution θ of X^* defined by $\theta(a) = b$, $\theta(b) = a$. The language $L = \{a, a^2, ab\}$ is a stricly θ-p-compliant language. Let T be a maximal strictly θ-p-compliant language containing L. Remark first that $bX^* \cap T = \emptyset$. If not, let $by \in T$. Since $b = \theta(a)$ and $a \in T$, we have $\theta(a)y \in T$. Hence $y = 1$ and $a, \theta(a) \in T$, in contradiction with the strictness of T. If $u \in T$, then $u = ax$ and $\theta(u) = b\theta(x)$. Hence $\theta(T) \subseteq bX^*$.

We will show that T is infinite by showing that $a^+ \subseteq T$. We have $a, a^2 \in T$. Suppose that $a^k \in T$ and that $a^{k+1} \notin T$. Let $T' = T \cup a^{k+1}$. Then T' is not a strictly θ-p-compliant language. That means that one or both of the following situations hold:

(1) There is a word u such that $u, \theta(u) \in T'$ with $u = \theta(u)$. Since T is strictly θ-p-compliant, $u = a^{k+1}$ or $\theta(u) = a^{k+1}$. In the first case, $\theta(u) = b^{k+1} \notin T'$, a contradiction. In the second case, $u \in T$ and hence $\theta(u) \in bX^*$, a contradiction.

(2) T' is not θ-p-compliant, i.e. there are words $u, v \in T'$ such that $\theta(u)x = v$ with $x \neq 1$ and either $u \notin T$ or $v \notin T$.

If $v \notin T$, then $v = a^{k+1}$. From $\theta(T) \subseteq bX^*$ follows that $\theta(T') \subseteq bX^*$ and hence $\theta(u)x = bz = a^{k+1}$, a contradiction.

If $v \in T$, then $u \in T' \setminus T$, i.e. $u = a^{k+1}$ and

$$\theta(a^{k+1})x = b^{k+1}x = v \in bX^*$$

a contradiction since $bX^* \cap T = \emptyset$.

Therefore $a^{k+1} \in T$.

It follows then that every maximal θ-p-compliant language containing the finite language L is infinite, showing that the result for finite prefix codes cannot be extended to finite θ-p-compliant languages for every involution θ.

We now connect the notion of maximality to the notions of density, residue, and ideals generalized to arbitrary involutions.

The following definition is a further generalization of the notion of a dense language, from the particular case of the identity involution to that of an arbitrary involution.

Definition 6.1. *Let θ be an involution of X^*. A language $L \subseteq \Delta^*$ is said to be (right,left) θ-dense if for every $u \in \Delta^*$, there are $x, y \in \Delta^*$ such that $x\theta(u)y \in L$ ($\theta(u)y \in L$, $x\theta(u) \in L$).*

Based on the preceding definition, the following result now generalizes Proposition 4.6 in Section 4 dealing with the particular case of the complementary involution.

Proposition 6.3. *Let θ be an involution of X^* that is a morphism and let $L \subseteq X^*$ be a stricly θ-p-compliant language. If L^* is right θ-dense, then L is a maximal stricly θ-p-compliant language.*

Proof. Suppose L is not maximal. Then there is $v \notin L$ such that $T = L \cup \{v\}$ is a strictly θ-p-compliant language. The right θ-density of L^* implies the existence of $x \in \Delta^*$ such that $\theta(v)x \in L^*$, i.e. $\theta(v)x = u_1u_2 \cdots u_k$, $u_i \in L$. This implies either $\theta(v) = u_1y$ or $u_1 = \theta(v)z$, i.e. $\theta(u_1)\theta(y) = v \in T$ or $\theta(v)z = u_1 \in T$. Since T is a θ-p-compliant language, either $\theta(y) = 1$ and $\theta(u_1) = v$, or $z = 1$ and $\theta(v) = u_1$, a contradiction because of the strict compliance of T. ♣

Recall that, [18], the residue $R(L)$ of a language $L \subseteq X^*$ is defined as

$$R(L) = \{u|\ u \in X^*, X^*uX^* \cap L = \emptyset\}.$$

We can now generalize this notion from the identity involution to an arbitrary involution as follows.

Definition 6.2. *Let $L \subseteq X^*$ and let θ be an involution of X^*. The θ-residue $R_\theta(L)$ of L is defined by:*

$$R_\theta(L) = \{u|u \in X^*,\ \ X^*\theta(u)X^* \cap L = \emptyset\}$$

The right and the left θ-residue of L are defined similarly.

It is immediate that a language L is θ-dense if and only if its θ-residue is empty. Recall that, [18], a language $L \subseteq X^*$ is called a (right, left) ideal of X^* if $u \in L$, $x, y \in X^*$ imply $xuy \in L$ ($uy \in L$, $xu \in L$). The following definition generalizes the notion of ideal.

Definition 6.3. *Let θ be an involution of X^*. A language $L \subseteq X^*$ is called a (right, left) θ-ideal of X^* if $u \in L$, $x, y \in X^*$, implies $x\theta(u)y \in L$ ($\theta(u)y \in L$, $x\theta(u) \in L$).*

If θ is the identity involution, then the above notions correspond to the usual notions of residue and ideal.

Proposition 6.4. *Let θ be an involution of X^* and let L be a (right, left) θ -ideal of X^*. Then $\theta(L) = L$ and L is a (right, left) ideal of X^*.*

Proof. Let $u \in L$. Then $\theta(u) = 1.\theta(u).1 \in L$ and therefore $\theta(L) \subseteq L$. This implies $L \subseteq \theta(\theta(L)) \subseteq \theta(L)$ and hence $\theta(L) = L$. Since $u \in L$, there exists then $v \in L$ such that $\theta(v) = u$. Therefore $xuy = x\theta(v)y \in L$ and therefore L is an ideal. ♣

The previous proposition shows that every θ-ideal is an ideal, but the converse is not true in general. For example, the language $L = \Delta^* C \Delta^*$ is an ideal of Δ^*. However $C \in L$, but $\tau(C) = \overleftarrow{C} = G \notin L$. This implies $\tau(L) \neq L$ and hence L cannot be a τ-ideal.

Proposition 6.5. *Let θ be an involution of X^*. If not empty, the (right, left) θ-residue $R_\theta(L)$ of a language $L \subseteq X^*$ is a (right, left) θ-ideal of X^*.*

Proof. Let $u \in R_\theta(L)$ and suppose that, for some $x, y \in X^*$, $x\theta(u)y \notin R_\theta(L)$. This implies $X^* x\theta(u)y X^* \cap L \neq \emptyset$ and hence $X^*\theta(u)X^* \cap L \neq \emptyset$, a contradiction.♣

Proposition 6.6. *Let θ be an involution of X^*. A language $L \subseteq X^*$ is an ideal if and only if $\theta(L)$ is an ideal of X^*.*

Proof. ($\Rightarrow$) Let $u \in \theta(L)$ and $x, y \in X^*$. Since θ is bijective, there exist $v \in L$, $x', y' \in X^*$ such that $\theta(v) = u$, $\theta(x') = x$ and $\theta(y') = y$. Since L is an ideal, $x'vy' \in L$ and $y'vx' \in L$.

If θ is a morphism, then:

$$xuy = \theta(x')\theta(v)\theta(y') = \theta(x'vy') \in \theta(L).$$

If θ is an anti-morphism, then:

$$xuy = \theta(x')\theta(v)\theta(y') = \theta(y'vx') \in \theta(L).$$

($\Leftarrow$) Let $u \in L$, $x, y \in X^*$ and let x', y' such that $\theta(x') = x$, $\theta(y') = y$. Then $x'\theta(u)y' \in \theta(L)$ and $y'\theta(u)x' \in \theta(L)$. If θ is a morphism, we have:

$$xuy = \theta(x')\theta^2(u)\theta(y') = \theta(x'\theta(u)y') \in \theta(\theta(L)) = L$$

and if it is an anti-morphism, we have:

$$xuy = \theta(x')\theta^2(u)\theta(y') = \theta(y'\theta(u)x') \in \theta(\theta(L)) = L.$$

Hence L is an ideal. ♣

We now consider relations between density and maximality. The next result shows that if θ is an involution of X^*, then right density of a language L is equivalent to right θ-density of L.

Proposition 6.7. *Let θ be an involution of X^*. A language $L \subseteq X^*$ is right θ-dense (θ-dense) if and only if it is right dense (dense).*

Proof. Suppose L is right θ-dense and let $u \in X^*$. Since an involution is a bijective mapping, there exists $v \in X^*$ such that $u = \theta(v)$. The right θ-density of L implies the existence of $x \in X^*$ such that $\theta(v)x \in L$ and hence $ux \in L$.

Suppose L is right dense and let $u \in X^*$. Since $\theta(u) \in X^*$, then there is $x \in X^*$ such that $\theta(u)x \in L$ and thus L is right θ-dense.

The proof for θ-density is similar to the proof for right θ-density. ♣

If the involution θ is the identity, then a θ-p-compliant language L is a prefix code and we obtain the known result that if L^* is right dense, the prefix code L is maximal. The converse is true for prefix codes, but not in general as shown in the following example.

Example. Let $X = \{a, b\}$ and let θ be defined as $\theta(a) = b$, $\theta(b) = a$. This involution of X^* is also a morphism of X^*.

Let $L = aX^*$. This language L is a maximal strictly θ-p-compliant language. Indeed, suppose L is not maximal. Then L is contained in a strictly $\theta - p$-compliant language T with $L \subset T$. Let $u \in T \setminus L$. Then $u = by$ for some $y \in X^*$ and:

$$u = \theta(a)\theta(y) \in T,$$

a contradiction with the fact that T is strictly θ-p-compliant. It follows then that $L = aX^*$ is a maximal strictly θ-p-compliant language.

The language L^* is not right θ-dense. This follows from the fact that $(aX^*)^* \cap bX^* = \emptyset$. This example shows that, contrary to the case of maximal prefix codes, a maximal strictly θ-p-compliant language is not necessarily right θ-dense.

7 Conclusions

This paper is a preliminary study in the formalization and algebraic treatment of problems that arise when encoding data on a DNA substrate, where each word corresponds to a single DNA strand. We define and study the algebraic structure of DNA compliant languages, i.e. languages displaying several "good encoding properties". Moreover, viewing words and languages from this biological perspective leads to generalizations of several well-known notions such as infix code, prefix code, suffix code, density, residue and ideals.

References

1. L. Adleman. Molecular Computation of Solutions to Combinatorial Problems. *Science*, 266, 1994, 1021-1024.
2. E.B. Baum. DNA Sequences Useful for Computation. Proceedings of *DNA-based Computers II*, Princeton. In AMS DIMACS Series, vol.44, L.F.Landweber, E.Baum Eds., 1998, 235-241.
3. R. Deaton, R. Murphy, M. Garzon, D.R. Franceschetti, S.E. Stevens. Good Encodings for DNA-based Solutions to Combinatorial Problems. Proceedings of *DNA-based Computers II*, Princeton. In AMS DIMACS Series, vol.44, L.F.Landweber, E.Baum Eds., 1998, 247-258.

4. R. Deaton, M. Garzon, R. Murphy, D.R. Franceschetti, S.E. Stevens. Genetic Search of Reliable Encodings for DNA Based Computation, *First Conference on Genetic Programming GP-96*, Stanford U., 1996, 9-15.
5. R. Deaton, R.E. Murphy, J.A. Rose, M. Garzon, D.R. Franceschetti, S.E. Stevens Jr. A DNA Based Implementation of an Evolutionary Search for Good Encodings for DNA Computation. Proc. *IEEE Conference on Evolutionary Computation ICEC-97*, 267-271.
6. U. Feldkamp, S. Saghafi, H.Rauhe. DNASequenceGenerator - A Program for the Construction of DNA Sequences. In [16], 179-189.
7. A.G. Frutos, Q. Liu, A.J. Thiel, A.M.W. Sanner, A.E. Condon, L.M. Smith, R.M. Corn. Demonstration of a Word Design Strategy for DNA Computing on Surfaces. *Nucleic Acids Research*, 25(23), 1997, 4748-4757.
8. M. Garzon, P.Neathery, R. Deaton, R.C. Murphy, D.R. Franceschetti, S.E. Stevens Jr., A New Metric for DNA Computing. In J.R. Koza, K.Deb, M. Dorigo, D.B. Vogel, M. Garzon, H.Iba, R.L. Riolo, Eds., Proc. *2nd Annual Genetic Programming Conference*, Stanford, CA, 1997, Morgan-Kaufmann, 472-478.
9. M. Garzon, R. Deaton, L.F. Nino, S.E. Stevens Jr., M. Wittner. Genome Encoding for DNA Computing *Proc. 3rd Genetic Programming Conference*, Madison, WI, 1998, Morgan Kaufmann, 684-690.
10. M. Garzon, C. Oehmen. Biomolecular Computation in Virtual Test Tubes. In [16], 75-83.
11. Handbook of Formal Languages. G. Rozenberg, A. Salomaa Eds., Springer Verlag, Berlin, 1997.
12. A.J. Hartemink, D.K. Gifford, J. Khodor. Automatic Constraint-Based Nucleotide Sequence Selection for DNA Computations. Proceedings of *DNA-based Computers IV*, Philadelphia. In *Biosystems* vol.52, nr.1-3, L.Kari, H.Rubin, D.H.Wood Guest Eds., 1999, 227-235.
13. H. Jürgensen, S. Konstantinidis. Codes. In [11] vol.3, 511-600.
14. L. Kari. DNA Computing: Arrival of Biological Mathematics. *The Mathematical Intelligencer*, vol.19, nr.2, Spring 1997, 9–22.
15. A. Marathe, A. Condon, R. Corn. On Combinatorial DNA Word Design. Proceeedings of *DNA-based Computers V*, June 14-15, 1999, E. Winfree, D. Gifford Eds., 75-89.
16. *Pre-Proceedings of DNA-based Computers VII*, Tampa, Florida, June 10-13, 2001, N. Jonoska, N.C. Seeman Eds.
17. J.H. Reif, T.H. LaBean, M. Pirrung, V.S. Rana, B. Guo, C. Kingsford, G.S. Wickham. Experimental Construction of Very Large Scale DNA Databases with Associative Search Capability. In [16], 241-250.
18. H.J.Shyr, *Free Monoids and Languages*, Hon Min Book Company, Taichung, Taiwan, R.O.C., 1991.