Coding properties of DNA languages[**]

Salah Hussini[*], Lila Kari[†], Stavros Konstantinidis[*]

Abstract

The computation language of a DNA-based system consists of all the words (DNA strands) that can appear in any computation step of the system. In this work we define properties of languages which ensure that the words of such languages will not form undesirable bonds when used in DNA computations. We give several characterizations of the desired properties and provide methods for obtaining languages with such properties. The decidability of these properties is addressed as well. As an application we consider splicing systems whose computation language is free of certain undesirable bonds and is generated by nearly optimal comma-free codes.

1 Introduction

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 opposite polarity are called *Watson/Crick complements* and will join together to form a double helix in a process called *base-pairing*, or *hybridization* [9].

In most DNA computations, there are three basic stages which have to be performed. 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* [9]. Finally, the solutions, i.e. the result of the computation are sequenced and decoded.

[**] All correspondence to Lila Kari. Research partially supported by Grants R2824A01 and R22O259 of the Natural Sciences and Engineering Research Council of Canada.

[*]Department of Mathematics and Computing Science, Saint Mary's University, Halifax, Nova Scotia, B3H 3C3, Canada, s.konstantinidis@stmarys.ca

[†]Department of Computer Science, University of Western Ontario, London, Ontario, N6A 5B7 Canada, lila@csd.uwo.ca, http://www.csd.uwo.ca/~ lila

In many proposed DNA-based algorithms, the initial DNA solution will contain some oligonucleotides which represent single "codewords", and some oligonucleotides which are strings of catenated codewords. This leads to two main types of possible undesirable hybridizations. First, it is undesirable for any DNA strand representing a codeword to form a hairpin structure, which can happen if either end of the strand binds to another section of that same strand. Second, it is undesirable for any DNA strand representing a codeword to bind to either another codeword strand or to the catenation of two codeword strands. If such undesirable hybridizations occur, they will in practice render the involved DNA strands useless for the subsequent computations.

To formalize these problems, let us introduce some notations. An alphabet X is a finite non-empty set of symbols (letters). We denote by $\Delta = \{A, C, G, T\}$ the DNA alphabet. A word u over the alphabet X is a sequence of letters and $|u|$ will denote the length of u, that is, the number of letters in it. By 1 we denote the empty word consisting of zero letters. By convention, a word u over the DNA alphabet Δ will denote the corresponding DNA strand in the $5'$ to $3'$ orientation. We denote by $\overleftarrow{u}$ the Watson-Crick complement of the sequence u, i.e., its reverse mirror image. For example, if $u = 5' - AAAAGG - 3'$ then $\overleftarrow{u} = 5' - CCTTTT - 3'$ will be the Watson-Crick complement that would base-pair to u. By X^* we denote the set of all words over X, and by X^+ the set of all non-empty words over X. A language (over X) is any subset of X^*. For a language K, we denote by K^* the set of words that are obtained by concatenating zero or more words of K; then, $1 \in K^*$. We write K^+ for the subset of all non-empty words of K^*.

Let now $K \subseteq \Delta^*$ be a set of DNA codewords. Several types of undesirable situations can occur with these DNA strands encoding initial data.

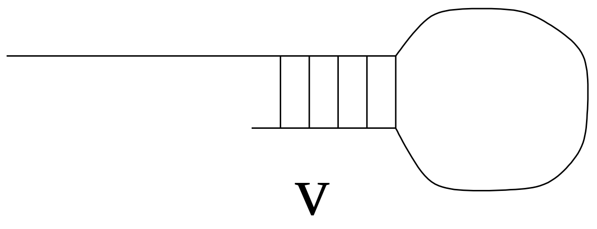

Figure 1. Intramolecular hybridization I: $uv \in K$, $\overleftarrow{v}$ being a subword of u.

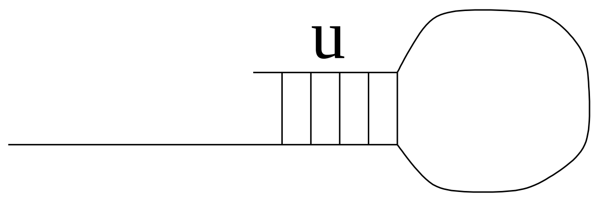

Figure 2. Intramolecular hybridization II: $uv \in K$, $\overleftarrow{u}$ being a subword of v.

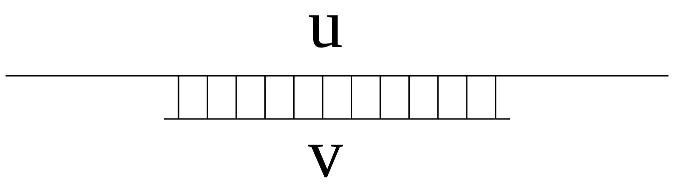

Figure 3. Intermolecular hybridization I: $u, v \in K$, $\overleftarrow{\overline{v}}$ being a subword of u.

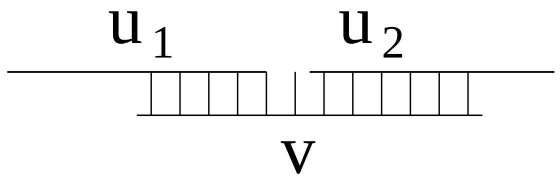

Figure 4. Intermolecular hybridization II: $u_1, u_2, v \in K$, $\overleftarrow{\overline{v}}$ being a subword of u_1u_2.

Several attempts have been made to address this issue by trying to find sets of codewords which are unlikely to form undesired bonds in this manner [2], [5], [4]. For example genetic algorithms have been developed which select for sets of DNA sequences that are less likely to form undesirable bonds [3], and combinatorial methods have been used to calculate bounds on the size of a set of uniform codewords (as a function of codeword length) which are less likely to mishybridize [11].

In this paper, we continue the approach in [10] where the notion of θ-compliance has been defined, where θ is an arbitrary morphic or antimorphic involution. In the particular case when θ is the Watson-Crick complement, a language is strictly θ-compliant (respectively strictly prefix θ-compliant, suffix θ-compliant) if situations like the ones depicted in Figure 3 (respectively Figures 1, 2) cannot occur. [10] studied properties of such θ-compliant languages, i.e. languages consisting of words with good coding properties. Here we define the notion of θ-freedom: if θ is the Watson-Crick complement, a language is θ-free iff situations like the ones depicted in Figure 4 cannot occur.

Section 3 studies θ-freedom and its relation to θ-compliance. It turns out that, under some conditions, languages that are θ-free are also θ-compliant.

Section 4 studies the question of whether or not, given a set of codewords, we can constructively decide if the set is θ-compliant or θ-free, i.e., if it has desirable coding properties.

Section 5 studies under what conditions, if we start with an initial set of "good" codewords, θ-compliance and θ-freedom are preserved by all intermediate strands that occur during computations.

With these studies, we attempt to gain a deeper insight into encoding information in DNA, which will hopefully assist solving this otherwise difficult problem.

2 Definitions and notations

For a set S, we denote by $|S|$ the cardinality of S, that is, the number of elements in S. The set of non-negative integers is denoted by $\mathbf{N}$. Let X^* be the free monoid generated by the finite alphabet X. 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^*.

An involution $\theta : S \to S$ of S is a mapping such that θ^2 equals the identity mapping, i.e., $\theta(\theta(x)) = x$ for all $x \in S$. It follows then that an involution θ is bijective and $\theta = \theta^{-1}$. The identity mapping is a trivial example of involution.

If Δ^* is the free monoid generated by the DNA-alphabet Δ then two involutions can be defined on Δ^*: the mirror involution μ which is an anti-morphism, and the complement involution γ which is a morphism ([10]).

Indeed, the mapping $\gamma : \Delta \to \Delta$ defined by $\gamma(A) = T, \gamma(T) = A, \gamma(C) = G, \gamma(G) = C$ can be extended in the usual way to a morphism of Δ^* that is also an involution of Δ^*. Obviously, γ is an involution as the complement of the complement of a sequence equals the sequence itself. This involution γ will be called the *complement involution* or simply the *c-involution of* Δ^*.

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

$$u = a_1 a_2 \ldots a_k, \quad v = a_k \ldots a_2 a_1, a_i \in \Delta, 1 \leq 1 \leq k.$$

The word v is called the mirror image of u. Since μ^2 is the identity mapping, μ is an involution of Δ^* which will be called the *mirror involution* or simply the *m-involution* of Δ^*. The m-involution is an anti-morphism as $\mu(uv) = \mu(v)\mu(u)$ for all $u, v \in \Delta^*$.

It is easy to see that γ and μ commute, i.e. $\gamma\mu = \mu\gamma$, and hence $\gamma\mu$ which will denoted by τ is also an involution of Δ^*. Furthermore τ is an anti-morphism which corresponds to the notion of Watson-Crick complement of a DNA sequence and will therefore sometimes be called the DNA involution.

Instead of γ, μ, τ we sometimes use the alternative notation

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

Following [10], if $\theta : X^* \to X^*$ is a morphic or antimorphic involution, a language $L \subseteq X^*$ is said to be *θ-compliant*, (prefix θ-compliant, suffix θ-compliant) iff, for any words $x, y, u \in X^*$,

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

If, in addition, $L \cap \theta(L) = \emptyset$ then the language L is called strictly θ-compliant (strictly prefix θ-compliant, strictly suffix θ-compliant, respectively).

Note that if the involution θ is the DNA involution, i.e., it represents the Watson-Crick complement, then a language L being strictly θ-compliant (strictly prefix θ-compliant, strictly suffix θ-compliant) amounts to the fact that situations of the type depicted in Figure 3 (respectively Figure 1, Figure 2) do not occur. Such languages with good coding properties have been studied in [10].

We close the section with some terminology on codes [1]. A code K is a subset of X^+ satisfying the property that, for every word w in K^+, there is a unique sequence $(v_1, v_2, \ldots, v_n)$ of words in K such that $w = v_1 v_2 \cdots v_n$. A bifix code K is a prefix and suffix code; that is, $K \cap KX^+ = K \cap X^+K = \emptyset$.

Every bifix code is indeed a code. An infix code, K, has the property that no word of K is properly contained in another word of K, that is, $K \cap (X^+KX^* \cup X^*KX^+) = \emptyset$. Every infix code is a bifix code. A comma-free code K is a language with the property $K^2 \cap X^+KX^+ = \emptyset$. Every comma-free code is a bifix code.

3 Involution-freedom and involution-compliance

In this section we define the notion of an involution-free language which formalizes the situation depicted in Figure 4. Moreover, this notion extends the concept of comma-free code, [13], in the same fashion that involution-compliance extends the concept of infix code.

We define the notion of a θ-free language (Definition 1) and establish relations between θ-freedom and θ-compliance (Lemma 1). We also establish some properties of θ-free languages. For example, Proposition 1 states that if a set of codewords K is strictly θ-free then the set consisting of arbitrary catenations of codewords from K will be strictly θ-free as well. Proposition 2 states that if a language is dense (it contains all possible sequences in X^* as subsequences of some of its strands), then it cannot be strictly θ-compliant or θ-free.

To aid the intuition, Propositions 3, 6, 7, 8 are proved for the particular case of the complement involution even though the results hold for the more general case of an arbitrary morphic involution.

After some examples of complement-free languages, Propositions 3, 6 bring together the notions of complement-compliance and complement-freedom. As it turns out, under some conditions, languages that avoid undesirable bindings of the type in Figures 1–3 avoid also situations of the type in Figure 4, and vice-versa.

Definitions 3 and 4 introduce the notions of complement-reflective respectively anti complement-reflective languages, which are generalizations of the notions of reflective and anti-reflective languages, [13], and relates them to complement-freedom.

This paves the way to methods of constructing complement-free languages as the one described in Proposition 7.

Proposition 8 gives a sufficient condition under which the catenation of two languages is complement-free.

Propositions 9, 10 address similar problems but this time for the anti-morphic involution case which is the type that the DNA Watson/Crick involution belongs to. Because of the additional "flipping" complication that such an involution entails, the results are slightly different.

Definition 1 *Let θ be an involution of X^*. A language K is called θ-free if $K^2 \cap X^+\theta(K)X^+ = \emptyset$. If, in addition, $K \cap \theta(K) = \emptyset$ then K is called strictly*

θ-free. For convenience, we agree to call K strictly θ-free when $K \setminus \{1\}$ is strictly θ-free.

As an example, consider the DNA involution τ of Δ^* and the language $ACC\Delta^2$. Then, $ACC\Delta^2 ACC\Delta^2 \cap \Delta^+(\Delta^2 GGT)\Delta^+ = \emptyset$. Hence, $ACC\Delta^2$ is τ-free and, in fact, strictly τ-free. The same language is strictly free for the complement involution as well. On the other hand, $A\Delta^2 A$ is strictly τ-compliant but not τ-free, as $ACTAATAA \in (A\Delta^2 A)^2 \cap \Delta^+\tau(A\Delta^2 A)\Delta^+$.

Lemma 1 *Let θ be a morphic or antimorphic involution and let K be a non-empty subset of X^+.*

(i) If K is θ-free then both K and $\theta(K)$ are θ-compliant.

(ii) If K is strictly θ-free then $K^2 \cap X^*\theta(K)X^* = \emptyset$.

Proof. (i) Assume K is θ-free but suppose it is not θ-compliant, that is, we have $X^*\theta(K)X^+ \cap K \neq \emptyset$ or $X^+\theta(K)X^* \cap K \neq \emptyset$. The first case implies $KX^*\theta(K)X^+ \cap KK \neq \emptyset$ which contradicts the assumption about K. The second case is also impossible and, therefore, K is θ-compliant. That $\theta(K)$ is θ-compliant follows from the fact that a language L is θ-compliant if and only if $\theta(L)$ is θ-compliant.

(ii) Assume K is strictly θ-free but suppose $u_1u_2 = x\theta(v)y$ for some words $u_1, u_2, v \in K$ and $x, y \in X^*$. Then at least one of x and y is empty. If they are both empty then $K^2 \cap \theta(K) \neq \emptyset$ which implies $\theta(K)\theta(K) \cap K \neq \emptyset$ and then that K is not θ-compliant; a contradiction. If x is empty then it follows that $\theta(K) \cap K \neq \emptyset$, or $X^*\theta(K)X^+ \cap K \neq \emptyset$, or $X^*KX^+ \cap \theta(K) \neq \emptyset$, depending on whether $|u_1|$ is equal to, less than, or greater than $|\theta(v)|$. In any case a contradiction arises. The case of $y = 1$ is dual. $\diamond$

Proposition 1 *Let θ be a morphic or antimorphic involution and let K be a non-empty subset of X^+. If K is strictly θ-free then also K^+ is strictly θ-free.*

Proof. Assume K is strictly θ-free, that is $K^2 \cap X^+\theta(K)X^+ = \emptyset$, but suppose K^+ is not strictly θ-free; then, there is a positive integer m and words $w \in K^+$ and $u_j \in K$, where $j = 1, \ldots, m$, such that $w \in X^+\theta(u_1 \cdots u_m)X^+$ or $w = \theta(u_1 \cdots u_m)$. In either case, there is a positive integer n and words $w_1, \ldots, w_n$ in K such that $w = w_1 \cdots w_n$ and $w \in X^*\theta(u_1 \cdots u_m)X^*$. Then, the assumption about θ implies $w_1 \cdots w_n \in X^*\theta(u_1)X^*$. This in turn implies that there are integers k and r with $r \geq 1$ such that the first symbol of $\theta(u_1)$ occurs in w_{k+1} and the last symbol of $\theta(u_1)$ occurs in w_{k+r}. If $r = 1$ then $w_{k+r} = w_{k+1}$ and $w_{k+1} \in X^*\theta(u_1)X^*$ which is impossible. If $r = 2$ then there is a non-empty suffix of w_{k+1}, say s_1, and a non-empty prefix of w_{k+2}, say p_2, such that $\theta(u_1) = s_1p_2$. Moreover, there are words p_1 and s_2 such that $w_{k+1} = p_1s_1$ and $w_{k+2} = p_2s_2$. This implies $p_1\theta(u_1)s_2 \in K^2$ which is a contradiction. Finally, if $r > 2$ then $\theta(u_1) = sw_{k+2} \cdots w_{k+r-1}p$ where s is a non-empty suffix of w_{k+1} and p a non-empty prefix of w_{k+r}. This implies $\theta(u_1) \in X^+KX^+$ which again is impossible. $\diamond$

It is shown next that the properties of strict θ-compliance and strict θ-freedom impose restrictions on the words of a language in terms of density and completeness. A language L is called dense, [1], if every word is a subword of some word of L; that is, $L \cap X^*wX^* \neq \emptyset$ for every $w \in X^+$. The language L is complete if L^* is dense, [1].

Proposition 2 *Let θ be a morphic or antimorphic involution and let K be a non-empty subset of X^+.*

(i) If K is dense then K is not strictly θ-compliant.

(ii) If K is complete then K is not strictly θ-free.

Proof. Assume K is dense and consider any word u in K. As $\theta(u) \in X^+$, $K \cap X^*\theta(u)X^* \neq \emptyset$ which implies that K is not strictly θ-compliant. Now assume K is complete and consider any word w in K^+. As $\theta(w) \in X^+$, one has $K^+ \cap X^*\theta(w)X^* \neq \emptyset$ which implies that K^+ is not strictly θ-compliant. Then, using Proposition 1, it follows that K is not strictly θ-free. $\diamond$

To make results more intuitive, we will prove the following four propositions for the particular case where θ is the complement involution. Note that the results can be generalized to refer to any arbitrary morphic involution.

There are many examples of complement-free languages. $L = AC^+A$ is an infinite complement-free language as

$$\{AC^{n_1}AAC^{n_2}A \mid n_1, n_2 > 0\} \cap \Delta^+TG^+T\Delta^+ = \emptyset.$$

For any $L_1 \subseteq AC^*$ and $L_2 \subseteq C^+A$, L_1L_2 is a complement-free language. Indeed, $L_1L_2 = AC^+A$ which is complement-free.

Every subset of a complement-free language is itself complement-free.

Let $L \subseteq \Delta^+$ be a finite language and let m =max$\{|u| \mid u \in L\}$. Then $L' = AC^mLAC^m$ is complement-free. Indeed, assume $L'^2 \cap \Delta^+\overline{L'}\Delta^+ \neq \emptyset$. Then, $AC^mw_1AC^mAC^mw_2AC^m = \alpha_1TG^m\overline{w_3}TG^m\alpha_2$ for some $w_1, w_2, w_3 \in L$, $\alpha_1, \alpha_2 \in \Delta^+$. This implies TG^m is a subword of w_1 or w_2 – contradiction as $|w_1|$, $|w_2| \leq m$.

An example of a language L satisfying $L \cap \overline{L} = \emptyset$ is $\Delta^*\{A, C\}$.

For a language $L \subseteq X^+$ denote by

$$L_{pref} = \{x \in X^+ \mid xy \in L \text{ for some } y \in X^+\}$$

$$L_{suff} = \{y \in X^+ \mid xy \in L \text{ for some } x \in X^+\}$$

the language of non-empty proper prefixes of L and the language of non-empty proper suffixes of L respectively.

Proposition 3 *Let $L \subseteq X^+$, $L \neq \emptyset$, be a language. The following statements are equivalent:*

(1) L is complement-free.

(2) L is c-compliant and $\overline{L} \cap L_{suff}L_{pref} = \emptyset$.

(3) L is c-compliant and $L^2 \cap L_{pref}\overline{L}L_{suff} = \emptyset$.

(4) $\overline{L}$ is complement-free.

Proof. (1) $\Longrightarrow$ (2)

Assume L is complement-free, i.e., $L^2 \cap X^+\overline{L}X^+ = \emptyset$.

According to Lemma 1, L is c-compliant.

Let us show now that $\overline{L} \cap L_{suff}L_{pref} = \emptyset$.

Suppose that this is not true, i.e., there exists a word $w \in \overline{L} \cap L_{suff}L_{pref}$.

Then $w = xy, x \in L_{suff}, y \in L_{pref}$, which implies that there exist words $u, v \in X^+$ such that $ux \in L$, $yv \in L$.

As $w \in \overline{L}$, we have $\overline{w} \in L$. This implies that

$$uwv = (ux)(yv) = u(xy)v \in L^2 \cap X^+\overline{L}X^+$$

which contradicts the fact that L is complement-free.

(2) $\Longrightarrow$ (1) Assume L is c-compliant and $\overline{L} \cap L_{suff}L_{pref} = \emptyset$. Suppose that L is not complement-free, that is, $L^2 \cap X^+\overline{L}X^+ \neq \emptyset$. Then there exist $u_1, u_2, w \in L$ such that $u_1u_2 = x\overline{w}y$, $x, y \in X^+$. As L is c-compliant, $\overline{w}$ is not a subword of u_1 or of u_2. Consequently, the only possibility is that $\overline{w} = \overline{w_1w_2}$ where $u_1 = x\overline{w_1}$, $u_2 = \overline{w_2}y$, $w_1, w_2 \in X^+$. This means $\overline{w_1} \in L_{suff}$, $\overline{w_2} \in L_{pref}$. Then, $\overline{w} = \overline{w_1w_2} \in \overline{L} \cap L_{suff}L_{pref}$, i.e. $\overline{L} \cap L_{suff}L_{pref} \neq \emptyset$, which contradicts the assumptions.

(1) $\Longrightarrow$ (3). If L is complement-free then L is c-compliant. Moreover, as $L^2 \cap X^+\overline{L}X^+ = \emptyset$ we have that $L^2 \cap L_{pref}\overline{L}L_{suff} = \emptyset$.

(3) $\Longrightarrow$ (1) Assume L is c-compliant and $L^2 \cap L_{pref}\overline{L}L_{suff} = \emptyset$. Suppose that $L^2 \cap X^+\overline{L}X^+ \neq \emptyset$, i.e. there exist $u_1, u_2, v \in L$, $x, y \in X^+$ such that $u_1u_2 = x\overline{v}y$. If $\overline{v}$ would be a subword of u_1 or u_2 this would imply, according to the c-compliance of L that either $x = 1$ or $y = 1$ – a contradiction.

Consequently, the only possible situation is $u_1 = x\overline{v_1}, u_2 = \overline{v_2}y$, $\overline{v} = \overline{v_1v_2}$ with $x, y, v_1, v_2 \in X^+$. This further implies that $x \in L_{pref}$, $y \in L_{suff}$. As $v \in L$, we have that

$$u_1u_2 = x\overline{v_1v_2}y \in L^2 \cap L_{pref}\overline{L}L_{suff} \neq \emptyset$$

– a contradiction. Therefore, our assumption that L is not complement-free was false.

(1) $\Longleftrightarrow$ (4) is obvious as $L^2 \cap X^+\overline{L}X^+ = \emptyset$ implies $\overline{L}^2 \cap X^+LX^+ = \emptyset$. $\diamond$

Definition 2 *Let $\theta : X^* \longrightarrow X^*$ be a morphic or antimorphic involution. For a non-empty language $L \subseteq X^+$ define*

$L_s = \{z \in X^+ |\ ux \in L, xz \in \theta(L)$ *for some* $u, x \in X^+\}$,

$L_{is} = \{x \in X^+ |\ wx \in \theta(L), xv \in L$ *for some* $w, v \in X^+\}$,

$L_p = \{x \in X^+ |\ xv \in \theta(L), vy \in L$ *for some* $y, v \in X^+\}$,

$L_{ip} = \{x \in X^+ |\ ux \in L, xv \in \theta(L)$ *for some* $u, v \in X^+\}$.

Note that in the particular case, where we talk about a morphic involution we have that $L_{is} = \theta(L_{ip})$. Indeed $x \in L_{is}$ means $wx \in \theta(L), xv \in L$ for some $w, v \in X^+$ which means that $\theta(w)\theta(x) \in L$, $\theta(x)\theta(v) \in \theta(L)$ which, in turn, means that $\theta(x) \in L_{ip}$, that is, $x \in \theta(L_{ip})$.

Proposition 4 *Let $L \subseteq X^+$ be a non-empty language and $\theta : X^* \longrightarrow X^*$ be an antimorphic involution. Then $\theta(L)_s = \theta(L_p)$ and $\theta(L)_p = \theta(L_s)$.*

Proof. For the first equality, $u \in \theta(L_p)$ amounts to $\theta(u) \in L_p$ which means $\theta(u)v \in \theta(L)$, $vy \in L$, for some $v, y \in X^+$. This means $\theta(v)u \in L$, $\theta(y)\theta(v) \in \theta(L)$, that is, $u \in \theta(L)_s$.

For the second equality take $u \in \theta(L)_p$. Then $uv \in L$, $vy \in \theta(L)$ for some $v, y \in X^+$ which means $\theta(v)\theta(u) \in \theta(L)$, $\theta(y)\theta(v) \in L$, i.e. $\theta(u) \in L_s$, that is, $u \in \theta(L_s)$. $\diamond$

Proposition 5 *Let $L \subseteq X^+$ be a non-empty language and $\theta : X^* \longrightarrow X^*$ be a morphic involution. Then $\theta(L)_s = \theta(L_s)$ and $\theta(L)_p = \theta(L_p)$.*

Proof. If $z \in \theta(L)_s$ we have that $ux \in \theta(L)$, $xz \in L$ for some $u, x \in X^+$. This means $\theta(u)\theta(x) \in L$ and $\theta(x)\theta(z) \in \theta(L)$ i.e. $\theta(z) \in L_s$, that is, $z \in \theta(L_s)$. The other equality can be proved in a similar way. $\diamond$

Take the particular case where θ is the complement involution.

Proposition 6 *Let $L \subseteq X^+$ be a non-empty language. Then L is complement-free if and only if L is complement-compliant and one of the following conditions holds:*

(1) $L_s \cap L_{is} = \emptyset$,
(2) $L_p \cap L_{ip} = \emptyset$,
(3) $L \cap \overline{L_p}L_p = \emptyset$,
(4) $L \cap L_s\overline{L_s} = \emptyset$.

Proof. For the first implication let us assume that $\emptyset \neq L \subseteq X^+$ is a complement-free language, i.e., $L^2 \cap X^+\overline{L}X^+ = \emptyset$.

According to Lemma 1, L is complement compliant.

Let us show (1). Suppose that there exists $x \in L_s \cap L_{is} \neq \emptyset$. Then we have $uy \in L, yx \in \overline{L}$, for some $y, u \in X^+$ and $wx \in \overline{L}$, $xv \in L$ for some $w, v \in X^+$. Then

$$uyxv = (uy)(xv) = u(yx)v \in L^2 \cap X^+\overline{L}X^+ \neq \emptyset - \text{ a contradiction.}$$

Let us show (2). Suppose that there exists $x \in L_p \cap L_{ip} \neq \emptyset$. Then $xv \in \overline{L}, vy \in L$ for some $v, y \in X^+$, and $ux \in L, xw \in \overline{L}$ for some $u, w \in X^+$. Then

$$uxvy = (ux)(vy) = u(xv)y \in L^2 \cap X^+\overline{L}X^+ \neq \emptyset - \text{ a contradiction.}$$

To show (3), suppose $L \cap \overline{L_p}L_p \neq \emptyset$. Then there exist $u, v \in L_p$ such that $\overline{u}v \in L$. As $u \in L_p$ we have that $ux \in \overline{L}$, $xy \in L$ for some $x, y \in X^+$. As $v \in L_p$ we have that $vx' \in \overline{L}$, $x'y' \in L$ for some $x', y' \in X^+$. Then

$$\overline{u}vx'y' = (\overline{u}v)(x'y') = \overline{u}(vx')y' \in L^2 \cap X^+\overline{L}X^+ - \text{ a contradiction.}$$

Finally, to show (4), suppose $L \cap L_s\overline{L_s} \neq \emptyset$. Then there exist $u, v \in L_s$ such that $u\overline{v} \in L$. As $u \in L_s$ we have that $xy \in L$, $yu \in \overline{L}$ for some $x, y \in X^+$. As $v \in L_s$ we have that $x'y' \in L$, $y'v \in \overline{L}$ for some $x', y' \in X^+$. Then

$$xyu\overline{v} = (xy)(u\overline{v}) = x(yu)\overline{v} \in L^2 \cap X^+\overline{L}X^+ - \text{ a contradiction.}$$

For the reverse implication we have to show that if L is complement-compliant and satisfies one of the conditions (1), (2), (3) or (4) then L is complement-free. Suppose that L is complement-compliant but is not complement-free. Then

$$L^2 \cap X^+\overline{L}X^+ \neq \emptyset,$$

which implies that there exist $u_1, u_2, w \in L$ and $r, s \in X^+$ such that $u_1u_2 = r\overline{w}s$. As L is complement compliant and $r, s \neq 1$, the only possibility is that $u_1 = rx \in L$, $\overline{w} = xy \in \overline{L}$, $u_2 = ys \in L$, for some $x, y \in X^+$. It follows then that

$x \in L_p \cap L_{ip} \neq \emptyset$,
$y \in L_s \cap L_{is} \neq \emptyset$,
$u_1 = rx \in L \cap \overline{L_p}L_p \neq \emptyset$,
$u_2 = ys \in L \cap L_s\overline{L_s} \neq \emptyset$,
which contradict conditions (2), (1), (3) and (4) respectively. $\diamond$

Note that Proposition 6 holds also if we replace the complement involution with an arbitrary morphic involution.

Definition 3 *A language $L \subseteq X^+$, $L \neq \emptyset$ is called complement-reflective iff for all $x, y \in X^+$ we have that $xy \in L$ implies $\overline{yx} \in L$.*

Definition 4 *A language $L \subseteq X^+$, $L \neq \emptyset$ is called anti complement-reflective iff for all $x, y \in X^+$ we have that $xy \in L$ implies $\overline{yx} \notin L$.*

Note that every complement-free language is anti complement-reflective. Indeed, if for a complement-free language L we would have $xy \in L$ and $\overline{yx} \in L$ for some $x, y \in X^+$ then

$$(\overline{yx})(\overline{yx}) = \overline{yxyx} = \overline{y}(\overline{xy})\overline{x} \in X^+\overline{L}X^+ \cap L^2$$

which violates the condition of complement-freedom.

However, one can find anti complement-reflective languages that are not complement- free. Indeed, take $L = \{A^nT^{2n} \mid n \geq 1\}$ over the alphabet $\Delta = \{A, C, G, T\}$ with the usual complement function. Then L is anti complement-reflective but not complement-free. Indeed, $xy \in L$ implies $xy = A^nT^{2n}$ for some $n \geq 1$. For $\overline{yx}$ to belong to L, it must be the case that $x \in A^+$ and $y \in T^+$. Then $\overline{yx} = A^{2n}T^n \notin L$ therefore L is anti complement-reflective.

To show that L is not complement-free, take $u_1u_2 \in L$, that is, $u_1u_2 = A^nT^{2n}A^mT^{2m}$ for some $n, m \geq 1$. As it is possible to find some indices n, m, p such that $v \in L$, $v = A^pT^{2p}$ and $\overline{v} = T^pA^{2p}$ a subword of u_1u_2, it results that L is not complement-free.

In general the concatenation of two complement-free languages may not be complement- free. For example, take $L_1, L_2 \subseteq \Delta^+$,

$$L_1 = \{CACA, AC\}, \;\; L_2 = \{TGTG, GT\}.$$

Both L_1 and L_2 are complement-free. On the other hand L_1L_2 contains the word $u = CACATGTG$ therefore we have that $uu = CACATGTGCACATGTG \in (L_1L_2)^2$ and the word $v = ACGT \in L_1L_2$ has the property that $\overline{v} = TGCA$ is a subword of uu. This means that L_1L_2 is not complement-free.

Nevertheless, for a given finite language $L \subseteq \Delta^+$ we can always find a word $u \in \Delta^+$ such that uL is complement-free.

Indeed, if $L = \{u_1, u_2, \ldots, u_n\}$ is a finite language and m =max$\{|u| \mid u \in L\}$ then for the word $u = A^{m+1}C$, $m \geq 1$ the language uL is complement-free.

Take $\alpha_1\alpha_2 \in uLuL$. Then $\alpha_1\alpha_2 = A^{m+1}Cu_iA^{m+1}Cu_j$ for some $1 \leq i, j \leq n$. Consider $\alpha \in uL$. Then $\overline{\alpha} = T^{m+1}G\overline{u_k}$ for some $1 \leq k \leq n$. The only possibility for $\overline{\alpha}$ to be a subword of $\alpha_1\alpha_2$ would be that $T^{m+1}G$ is a subword of u_i or of u_j which is impossible because of the way m has been defined. Consequently, uL is complement-free.

Proposition 7 *For any finite language $L \subseteq \Delta^+$ there exists an infinite language $R \subseteq \Delta^+$ such that RL is a complement-free language.*

Proof. Let $L \subseteq \Delta^+$ be a finite complement-free language and m =max$\{|u| \mid u \in L\}$. Let $R = \{AC^{m+1+n}A|\; n \geq 1\}$. Then RL is a complement-free language.

Indeed, $RL = \bigcup_{u \in L}\{AC^{m+1+n}Au|\; n \geq 1\}$. Let $\alpha_1, \alpha_2, \alpha \in RL$. Then

$$\alpha_1\alpha_2 = AC^{m+1+n_1}AuAC^{m+1+n_2}Av, \alpha = AC^{m+1+n_3}Aw, \text{ with } u, v, w \in L.$$

The only possibility for $\overline{\alpha} = TG^{m+1+n_3}T\overline{w}$ to be a subword of $\alpha_1\alpha_2$ is for $TG^{m+1+n_3}T$ to be a subword of u or v which cannot happen because of the way m was defined. Consequently, RL is complement-free. $\diamond$

The next question to address is: given two languages $A, B \subseteq X^+$, when is AB a complement-free language?

Proposition 8 *Let $A, B \subseteq X^+$ be two non-empty languages. Assume that $A \cap \overline{B} = \emptyset$ and that $A \cup B$ is complement compliant. If $A_s \cap \overline{B_p} = \emptyset$ then AB is a complement-free language.*

Proof. Assume we have two languages A, B satisfying the required conditions and suppose that AB is not complement free. Then there exist $u_1, u_2, u_3 \in A$ and $v_1, v_2, v_3 \in B$, $r, s \in X^+$ such that $u_1v_1u_2v_2 \in (AB)^2$ and $r\overline{u_3v_3}s \in X^+\overline{AB}X^+$ with

$$u_1v_1u_2v_2 = r\overline{u_3v_3}s.$$

Since $A \cup B$ is complement compliant we cannot have $\overline{u_3}$ or $\overline{v_3}$ be strict subwords of any of u_1, v_1, u_2, v_2, nor can we have u_1, v_1, u_2, v_2 be strict subwords of $\overline{u_3}$ or $\overline{v_3}$.

Since $A \cap \overline{B} = \emptyset$ and therefore also $B \cap \overline{A} = \emptyset$, v_1 cannot equal $\overline{u_3}$, nor can u_2 equal $\overline{v_3}$.

Since $r, s \in X^+$, u_1 cannot equal $\overline{u_3}$ nor can v_2 equal $\overline{v_3}$.

Consequently, the only possible cases are:

Case (1). For some $x, y, x', y', s' \in X^+$ we have $u_1 = rx \in A$, $\overline{u_3} = xy \in \overline{A}$, $v_1 = yx' \in B$, $\overline{v_3} = x'y' \in \overline{B}$, $u_2 = y's' \in A$ and $s = s'v_2 \in X^+$. From here we conclude that $y \in A_s \cap \overline{B_p}$ - a contradiction.

Case (2). For some $x, y, x', y', r' \in X^+$ we have $r = u_1r' \in X^+$, $v_1 = r'x \in B$, $\overline{u_3} = xy \in \overline{A}$, $u_2 = yx' \in A$, $\overline{v_3} = x'y' \in \overline{B}$, $v_2 = y's \in B$. From here we deduce that $x' \in \overline{A_s} \cap B_p \neq \emptyset$ - a contradiction. $\diamond$

Corollary 1 *Let $A, B \subseteq X^+$ be two non-empty languages. If $A \cup B$ is strictly complement compliant and $A_s \cap \overline{B_p} = \emptyset$ then AB is complement free.*

Proof. The fact that $A \cup B$ is strictly complement compliant implies that $(A \cup B) \cap \overline{(A \cup B)} = (A \cap \overline{(A \cup B)}) \cup (B \cap \overline{(A \cup B)}) = \emptyset$. This further implies that both $A \cap \overline{(A \cup B)} = \emptyset$ and $B \cap \overline{(A \cup B)} = \emptyset$. We have now that

$$A \cap \overline{B} \subseteq A \cap \overline{(A \cup B)} = \emptyset$$

$$B \cap \overline{A} \subseteq B \cap \overline{(A \cup B)} = \emptyset$$

which imply that $A \cap \overline{B} = B \cap \overline{A} = \emptyset$ and we can now use the preceding proposition. $\diamond$

Remark that the preceding proposition and corollary hold also for any arbitrary morphic involution, not only for the complement involution.

The case of antimorphic involutions, i.e., of involutions of the type of the DNA involution, is slightly different.

Proposition 9 *Let $\theta : X^* \longrightarrow X^*$ be an antimorphic involution and $\emptyset \neq L \subseteq X^+$ be a language. Then L is θ-free if and only if L is θ-compliant and one of the following conditions hold:*

(1) $L_s \cap L_{is} = \emptyset$,

(2) $L_p \cap L_{ip} = \emptyset$,

(3) $L \cap \theta(L_s)L_p = \emptyset$,

(4) $L \cap L_s\theta(L_p) = \emptyset$.

Proof. Let L and θ be like in the proposition and assume that L is θ-free. According to Lemma 1, L is θ-compliant.

To show (1), let us suppose there exists $x \in L_s \cap L_{is} \neq \emptyset$. This means $uy \in L, yx \in \theta(L)$, for some $y, u \in X^+$ and $wx \in \theta(L), xv \in L$ for some $w, v \in X^+$. Then

$$uyxv \in L^2 \cap X^+\theta(L)X^+, \text{ a contradiction.}$$

To show (2), suppose there exists $x \in L_p \cap L_{ip}$. As $x \in L_p$ we have that $xv \in \theta(L)$, $vy \in L$ for some $v, y \in X^+$. As $x \in L_{ip}$ we have that $ux \in L, xw \in \theta(L)$ for some $u, w \in X^+$. Then

$$uxvy \in L^2 \cap X^+\theta(L)X^+, \text{ a contradiction.}$$

Let us show now (3) by supposing, for the sake of contradiction, that $L \cap \theta(L_s)L_p \neq \emptyset$. Then there exist $u \in L_s$, $v \in L_p$ such that $\theta(u)v \in L$. As $u \in L_s$, we have that $xy \in L$, $yu \in \theta(L)$ for some $x, y \in X^+$. As $v \in L_p$, we have that $vw \in \theta(L)$, $wz \in L$, for some $w, z \in X^+$. Then,

$$\theta(u)vwz \in L^2 \cap X^+\theta(L)X^+, \text{ a contradiction.}$$

To show (4), suppose $L \cap L_s\theta(L_p) \neq \emptyset$. Then there exist $u \in L_s, v \in L_p$ such that $u\theta(v) \in L$. As $u \in L_s$, we have that $xy \in L, yu \in \theta(L)$ for some $x, y \in X^+$. As $v \in L_p$ we have that $vw \in \theta(L), wz \in L$ for some $w, z \in X^+$. Then,

$$xyu\theta(v) \in L^2 \cap X^+\theta(L)X^+, \text{ a contradiction.}$$

Conversely, we have to prove that θ-compliance and any of the conditions (1), (2), (3), (4) imply θ-freedom.

Suppose, for the sake of contradiction, that L is not θ-free, that is, $L^2 \cap X^+\theta(L)X^+ \neq \emptyset$. Then there exist $u_1, u_2, w \in L$ such that $u_1u_2 = x\theta(w)y$ for some $x, y \in X^+$.

As L is θ-compliant and $x, y \in X^+$, $\theta(w)$ cannot be a subword of either u_1 or u_2, nor can u_1, u_2 be subwords of $\theta(w)$.

Therefore, the only possibility is that $u_1 = xr \in L$, $\theta(w) = rs \in \theta(L)$, $u_2 = sy \in L$, for some $r, s \in X^+$. This implies that $r \in L_p \cap L_{ip} \neq \emptyset$, contradicting (2) and that $s \in L_s \cap L_{is} \neq \emptyset$, contradicting (1).

Moreover, $xr \in L$, which implies $\theta(r)\theta(x) \in \theta(L)$ because θ is an antimorphism. This, together with the fact that $\theta(s)\theta(r) \in L$ imply $\theta(x) \in L_s$. As θ is an involution, this implies $x \in \theta(L_s)$. Consequently,

$$xr \in L \cap \theta(L_s)L_p, \text{ a contradiction with (3).}$$

Finally, $sy \in L$ which implies $\theta(y)\theta(s) \in \theta(L)$. This, together with the fact that $\theta(s)\theta(r) \in L$ imply that $\theta(y) \in L_p$ which means $y \in \theta(L_p)$. Therefore, $sy \in L \cap L_s\theta(L_p)$, contradiction with (4). $\diamond$

Proposition 10 *Let $A, B \subseteq X^+$ be two non-empty languages and $\theta : X^* \longrightarrow X^*$ be an antimorphic involution. Assume, furthermore, that A and B are strictly θ-compliant and $A \cup B$ is θ-compliant. If $B_s \cap \theta(A_s) = A_p \cap \theta(B_p) = \emptyset$ then AB is a θ-free language.*

Proof. Suppose AB is not θ-free. Then there exist $u_1, u_2, u_3 \in A$ and $v_1, v_2, v_3 \in B$, $r, s \in X^+$ such that $u_1v_1u_2v_2 \in (AB)^2$, $r\theta(u_3v_3)s \in X^+\theta(AB)X^+$ and

$$u_1v_1u_2v_2 = r\theta(v_3)\theta(u_3)s.$$

Since $A \cup B$ is θ-compliant we cannot have $\theta(u_3)$ or $\theta(v_3)$ be strict subwords of any of u_1, v_1, u_2, v_2 and neither can we have u_1, u_2, v_1, v_2 be strict subwords of $\theta(u_3)$ or $\theta(v_3)$.

As A and B are strictly θ-compliant and therefore $A \cap \theta(A) = B \cap \theta(B) = \emptyset$, v_1 cannot equal $\theta(v_3)$ nor can u_2 equal $\theta(u_3)$.

Since $r, s \neq 1$, u_1 cannot equal $\theta(v_3)$ nor can v_2 equal $\theta(u_3)$.

Consequently, the only possible cases are:

Case 1. $u_1 = rx \in A$, $\theta(v_3) = xy \in \theta(B)$, $v_1 = yx' \in B$, $\theta(u_3) = x'y' \in \theta(A)$, $u_2 = y's' \in A$, $s = s'v_2$, for some $x, y, x', y', s' \in X^+$.

As $x'y' \in \theta(A)$, $y's' \in A$ we have that $x' \in A_p$. As $yx' \in B$, $xy \in \theta(B)$ implies $\theta(x')\theta(y) \in \theta(B)$, $\theta(y)\theta(x) \in B$ we have $\theta(x') \in B_p$. This further implies $x' \in \theta(B_p)$. We conclude that $x' \in A_p \cap \theta(B_p)$ - a contradiction.

Case 2. $r = u_1 r' \in X^+$, $v_1 = r'x \in B$, $\theta(v_3) = xy \in \theta(B)$, $u_2 = yx' \in A$, $\theta(u_3) = x'y' \in \theta(A)$, $v_2 = y's \in B$ for some $x, y, x', y', r' \in X^+$.

As $r'x \in B$, $xy \in \theta(B)$ we have $y \in B_s$. As $x'y' \in \theta(A)$, $yx' \in A$ we have that $\theta(y')\theta(x') \in A$, $\theta(x')\theta(y) \in \theta(A)$ which implies $\theta(y) \in A_s$, that is, $y \in \theta(A_s)$. Consequently we have $y \in B_s \cap \theta(A_s)$- a contradiction. $\diamond$

4 Decidability issues

Given a family of sets of codewords, we ask if we can construct an effective algorithm that takes as input a description of a set of codewords from the given family and outputs the answer yes or no depending on whether or not the set is θ-free or θ-compliant (has good encoding properties). If such an algorithm exists, the question is deemed "decidable", otherwise it is "undecidable". This section considers the problem of deciding τ-compliance and τ-freedom where τ is the DNA involution. As in the case of infix and comma-free codes, – see [8] –, it turns out that these properties are decidable for regular languages but undecidable for context-free languages. We use regular expressions to represent regular languages and context-free grammars for context-free languages. If E is a regular expression (context-free grammar), $L(E)$ denotes the language represented by (generated by) E.

Proposition 11 *Let τ be the DNA involution. The following two problems are decidable.*

1. Input: A regular expression E.

Output: YES or NO depending on whether $L(E)$ is τ-free.

2. Input: A regular expression E.

Output: YES or NO depending on whether $L(E)$ is τ-compliant.

Proof. For the first problem, the algorithm is as follows:

(1) Construct a regular expresion E_1 such that $L(E_1) = \tau(L(E))$;

(2) Construct a regular expresion E_2 such that $L(E_2) = L(E)L(E)$;

(3) Construct a regular expression E_3 such that $L(E_3) = \Delta^+ L(E_1)\Delta^+$;

(4) If $L(E_2) \cap L(E_3) \neq \emptyset$ then output NO; else output YES.

For step (1), one can compute first a regular expression E_1' such that $L(E_1') = \mu(L(E))$; that is, $L(E_1)'$ is the mirror image of $L(E)$. Then, one can construct the regular expression E_1 by replacing every $\delta \in \Delta$ that occurs in E_1' with the complement of δ. The test condition in step (4) is decidable since the intersection of two regular languages is computable, and the emptiness problem for regular languages is decidable. Finally, using properties of regular expressions, one can verify that the steps (2) and (3) are computable as well.

For the second problem the algorithm is similar. $\diamond$

Proposition 12 *Let τ be the DNA involution. The following problems are undecidable:*

1. Input: A context-free grammar F.
Output: YES or NO depending on whether or not $L(F)$ is τ-free.
2. Input: A context-free grammar F.
Output: YES or NO depending on whether or not $L(F)$ is τ-compliant.

Proof. Consider the following version of the Post Correspondence Problem (PCP): Given an alphabet Σ and two morphisms $g, h : \Sigma^* \to \{C, G\}^*$, decide whether $E(g, h) = \emptyset$, where $E(g, h) = \{z \in \Sigma^+ \mid g(z) = h(z)\}$. This problem is undecidable – see [6], for instance. We assume that the given problems are decidable and then obtain contradictions by deducing that PCP must be decidable as well.

First, assume the first problem is decidable and consider an instance $g, h : \Sigma^* \to \{C, G\}^*$ of PCP. Then, $\Sigma = \{\alpha_1, \ldots, \alpha_n\}$ for some $n \geq 1$. Construct a context-free grammar $F = (\Delta, \{S, S_g, S_h, \}, S, R)$ such that the set R consists of the following rules:

$S \longrightarrow T^2 S_g T \mid A^2 S_h A^2 T$,
$S_g \longrightarrow A^j T^2 \tau g(\alpha_j) \mid A^j T S_g \tau g(\alpha_j)$, for all $i \in I$,
$S_h \longrightarrow h(\alpha_i) A^2 T^i \mid h(\alpha_i) S_h A T^i$, for all $i \in I$,

where $I = \{1, 2, \ldots, n\}$. One verifies that $L(F) = L_g \cup L_h$, where

$$L_g = \{T(TA^{j_1}) \cdots (TA^{j_k}) T^2 \tau g(\alpha_{j_1} \cdots \alpha_{j_k}) T \mid k \in \mathbf{N}; j_1, \ldots, j_k \in I\}$$

and

$$L_h = \{A^2 h(\alpha_{i_1} \cdots \alpha_{i_m}) A (AT^{i_m}) \cdots (AT^{i_1}) A^2 T \mid m \in N; i_1, \ldots, i_m \in I\}.$$

We show that $L(F)$ is not τ-free if and only if $E(g, h) \neq \emptyset$. This implies that PCP is decidable if τ-freedom is decidable for $L(F)$. In the sequel, we write $u_{g,k}$ for any word in L_g with parameter k and $v_{h,m}$ for any word in L_h with parameter m.

First assume there is a word $z = \alpha_{i_1} \cdots \alpha_{i_m} \in \Sigma^+$ with $g(z) = h(z)$. Let w be any word in $L(F)$. Then,

$$A^2 h(z) A (AT^{i_m}) \cdots (AT^{i_1}) A^2 T w \in A \tau(u_{g,m}) T w,$$

where $\tau(u_{g,m}) = A g(z) A^2 (T^{i_m} A) \cdots (T^{i_1} A) A$. This implies $L_h L(F) \cap \Delta^+ \tau(L_g) \Delta^+ \neq \emptyset$. Hence, $L(F)$ is not τ-free. Conversely, assume $L(F)$ is not τ-free. Then, there are $x, y \in \Delta^+$ such that $(L_g \cup L_h) L(F) \cap x \tau(L_g \cup L_h) y \neq \emptyset$. We show $E(g, h) \neq \emptyset$, by distinguishing four cases:

Case 1: $v_{h,m}w = x\tau(u_{g,k})y$ for some $w \in L(F)$. Note that $x\tau(u_{g,k})y = xAg(\alpha_{j_1}\cdots\alpha_{j_k})A^2(T^{j_k}A)\cdots(T^{j_1}A)Ay = xAg(\alpha_{j_1}\cdots\alpha_{j_k})A(AT^{j_k})\cdots(AT^{j_1})A^2y$. If the factor $h(\alpha_{i_1}\cdots\alpha_{i_m})$ of $v_{h,m}$ occurs in x then the factor $g(\alpha_{j_1}\cdots\alpha_{j_k})$ of $x\tau(u_{g,k})y$ must occur in w. We consider two subcases:

(a) $w \in L_g \Rightarrow xA \in v_{h,m}\Delta^+T^2$ which is impossible.

(b) $w = v_{h,r} \in L_h \Rightarrow xA = v_{h,m}A^2$, $g(\alpha_{j_1}\cdots\alpha_{j_k}) = h(\alpha_{l_1}\cdots\alpha_{l_r})$, and $A(AT^{j_k})\cdots(AT^{j_1})A^2y = A(AT^{l_r})\cdots(AT^{l_1})A^2T$. If $k = r$ then $E(g,h) \neq \emptyset$. If $k > r$ then there is $s \geq 1$ such that $(AT^{j_s})\cdots(AT^{j_1})A^2y = A^2T$ which is impossible. If $k < r$ again we get a contradiction.

Now suppose that the factor $h(\alpha_{j_1}\cdots\alpha_{j_m})$ of $v_{h,m}$ is the same as the factor $g(\alpha_{j_1}\cdots\alpha_{j_k})$ of $x\tau(u_{g,k})y$. Then, $A^2 = xA$, and $A(AT^{i_m})\cdots(AT^{i_1})A^2Tw = A(AT^{j_k})\cdots(AT^{j_1})A^2y$. As before, the only possiblility is $m = k$ which implies $E(g,h) \neq \emptyset$.

Case 2: $v_{h,m}w = x\tau(v_{h,r})y$ for some $w \in L(F)$. In this case, $x\tau(v_{h,r})y = xAT^2(A^{l_1}T)\cdots(A^{l_r}T)T\tau h(\alpha_{l_1}\cdots\alpha_{l_r})T^2y$. If the factor $h(\alpha_{i_1}\cdots\alpha_{i_m})$ of $v_{h,m}$ occurs in x then $\tau h(\alpha_{l_1}\cdots\alpha_{l_r})$ must occur in w and, therefore w ends with T^2y. Moreover, in this case, $y \in \{A,T\}^+$. This implies $w \in \{A,T\}^+\tau h(\alpha_{l_1}\cdots\alpha_{l_r})T^2 \{A,T\}^+ \cap (\{A,T\}^+\tau g(\Sigma^+)T \cup \{A,T\}^+h(\Sigma^+)A\{A,T\}^+)$ which is impossible. Now suppose that the factor $h(\alpha_{i_1}\cdots\alpha_{i_m})$ of $v_{h,m}$ is the same as the factor $\tau h(\alpha_{l_1}\cdots\alpha_{l_r})$ of $x\tau(v_{h,r})y$. Then, $A^2 = xAT^2(A^{l_1}T)\cdots(A^{l_r}T)T$ which is impossible.

Case 3: $u_{g,k}w = x\tau(u_{g,s})y$ for some $w \in L(F)$. Note that $x\tau(u_{g,s})y = xAg(\alpha_{p_1}\cdots\alpha_{p_s})A(AT^{p_s})\cdots(AT^{p_1})A^2y$. If the factor $g(\alpha_{p_1}\cdots\alpha_{p_s})$ of $x\tau(u_{g,s})y$ is the same as the factor $\tau g(\alpha_{j_1}\cdots\alpha_{j_k})$ of $u_{g,k}$ then $xA \in \Delta^+T^2$ which is impossible. If the factor $g(\alpha_{p_1}\cdots\alpha_{p_s})$ of $x\tau(u_{g,s})y$ occurs in w, we consider two subases:

(a) $w \in L_g \Rightarrow A(AT^{p_s})\cdots(AT^{p_1})A^2y = T$ which is impossible.

(b) $w \in L_h \Rightarrow A(AT^{p_s})\cdots(AT^{p_1})A^2y = A(AT^{l_r})\cdots(AT^{l_1})A^2T$ assuming $w = v_{h,r}$. As before, one shows that the only possibility is $s = r$ and, therefore, $E(g,h)$ contains $\alpha_{l_1}\cdots\alpha_{l_r} = \alpha_{p_1}\cdots\alpha_{p_s}$.

Case 4: $u_{g,k}w = x\tau(v_{h,m})y$ for some $w \in L(F)$. Again one considers as above two possibilities for the occurence of the factor $\tau g(\alpha_{j_1}\cdots\alpha_{j_k})$ of $u_{g,k}$ in $x\tau(v_{h,m})y$ and shows that either a contradiction is obtained, or $E(g,h) \neq \emptyset$.

For the second problem, assume it is decidable and consider again an instance $g,h : \Sigma^* \to \{C,G\}^*$ of PCP. Construct a context-free grammar F which generates the language $L_g \cup L_h$ such that

$$L_g = \{T^2A^{i_1}T\cdots TA^{i_m}T^2\tau g(\alpha_{i_m})\cdots\tau g(\alpha_{i_1})T \mid m \in \mathbf{N};\ i_1,\ldots,i_m \in I\}$$

and

$$L_h = \{A^2h(\alpha_{i_1})\cdots h(\alpha_{i_m})A^2T^{i_m}A\cdots AT^{i_1}A^3 \mid m \in \mathbf{N};\ i_1,\ldots,i_m \in I\}.$$

Using similar arguments as before one can verify that the language $L_g \cup L_h$ is τ-compliant if and only if $E(g,h) \neq \emptyset$. $\diamond$

5 Splicing systems that preserve good encoding

The preceding sections studied conditions under which sets of DNA codewords have good encoding properties, like DNA compliance and τ-freedom, where τ is the DNA involution. The next question is to characterize initial sets of codewords having the additional feature that the good encoding properties are preserved during *any* computation starting out from the initial set. As a computational model we have chosen the computation by splicing [7], [12], [9]. Proposition 14 states that if the splicing base (initial set of codewords) is strictly θ-free then all the sequences that may appear along any computation will not violate the property of θ-freedom. Proposition 15 is a stronger result stating that for any computational system based on splicing one can construct an equivalent one with "good", i.e. θ-compliance and θ-freedom properties, being preserved during any computation. Finally, Proposition 16 provides a method of construction of a strictly θ-free code that can serve as splicing base, i.e. initial soup, for a computation. Moreover, it is proved that from the point of view of the efficiency of representing information, the constructed code is close to optimal.

A multiset M over an alphabet Σ is a collection of words in Σ^* such that a word can occur in M more than once. More formally, a multiset M is a mapping of Σ^* into $\mathbf{N}$ such that $M(w)$ is the number of copies of the word w in the multiset. For a multiset M, we write $\text{supp}(M)$ to denote the set of (distinct) words that occur in M; that is, $\text{supp}(M) = \{w \in \Sigma^* \mid M(w) > 0\}$. A splicing system – see [9] – is a quadruple $\gamma = (\Sigma, T, A, R)$ such that Σ is an alphabet, T is a subset of Σ, the set of terminal symbols, A is a multiset over Σ, the initial collection of words, and R is a set of splicing rules of the form $\alpha_1\#\beta_1\$\alpha_2\#\beta_2$ with $\alpha_1, \alpha_2, \beta_1, \beta_2$ being words in Σ^*. For two multisets M and M' we write $M \Longrightarrow_\gamma M'$, if there is a splicing rule $\alpha_1\#\beta_1\$\alpha_2\#\beta_2$ in R and two words in M of the form $x_1\alpha_1\beta_1y_1$ and $x_2\alpha_2\beta_2y_2$ such that M' is obtained from M by replacing those words with $x_1\alpha_1\beta_2y_2$ and $x_2\alpha_2\beta_1y_1$ – see [9]. The language, $L(\gamma)$, *generated* by a splicing system $\gamma = (\Sigma, T, A, R)$ is the set $\{w \in T^* \mid \exists M,\ A \Longrightarrow_\gamma M,\ w \in \text{supp}(M)\}$. The *computation language of* γ is the set of words over Σ that can appear in any computation step of γ; that is, the language

$$\{w \mid w \in \text{supp}(M) \text{ and } A \Longrightarrow_\gamma^* M, \text{ for some multiset } M\}$$

Note that the language generated by γ is a subset of the computation language of γ.

When the computation language of a splicing system involves only symbols of the alphabet Δ, with the usual functionality of these symbols, one wants to prevent situations like the ones described in the introduction and this requires choosing languages with certain combinatorial properties. In this section, we consider a type of splicing systems in which only symbols of the alphabet Δ are used and require that the language of computation of such systems is strictly θ-free, where θ is a morphic or antimorphic involution. To this end, we utilize Proposition 1 by requiring that the computation language is of the form K^* for some strictly θ-free subset K of Δ^+. Moreover, we require that K is a splicing base, as defined below, to ensure that, after performing a splicing operation between two words in K^*, the resulting words are still in K^*

Definition 5 *A splicing base is a set of words K such that, for all $x_1, x_2, y_1, y_2 \in \Delta^*$ and for all $v_1, u_1, v_2, u_2 \in K^*$ with $|v_1 u_1| > 0$ and $|v_2 u_2| > 0$, if $x_1 v_1 u_1 y_1$, $x_2 v_2 u_2 y_2 \in K^*$ then $x_1 v_1 u_2 y_2$, $x_2 v_2 u_1 y_1 \in K^*$.*

In the following lemma it is shown that the splicing base condition is equivalent to the following:

(C_K) For all $x \in \Delta^*$ and for all $u \in K$, if xu is a prefix of K^* then $x \in K^*$ and if ux is a suffix of K^* then $x \in K^*$.

Remark: A splicing base is not necessarily a code. For example, $K = \{A, A^2\}$ is not a code but it is a splicing base.

Lemma 2 *(i) Condition (C_K) implies the following: For all $x \in \Delta^*$ and for all $v \in K^+$, if xv is a prefix of K^* then $x \in K^*$, and if vx is a suffix of K^* then $x \in K^*$.*

(ii) A set of words K is a splicing base if and only if (C_K) holds.

(iii) Every splicing base which is a code is a bifix code.

(iv) Every comma-free code is a splicing base.

Proof. (i) Assume (C_K) holds, and suppose xv is a prefix of K^* and $v = v_1 \ldots v_n$ for some positive integer n and some words $v_1, \ldots, v_n \in K$. As $(xv_1 \cdots v_{n-1})v_n$ is a prefix of K^*, (C_K) implies $xv_1 \cdots v_{n-1} \in K^*$. If $n = 1$ then $x \in K^*$; otherwise one repeats the same argument to show $xv_1 \cdots v_{n-2} \in K^*$. It follows ultimately that $x \in K^*$. Analogously, one can show $x \in K^*$ when vx is a suffix of K^*.

(ii) For the 'only if' part, assume xv is a prefix of K^* for some $v \in K$ and $x \in \Delta^*$. Then, there is a word y such that $xvy \in K^*$. As $x1vy \in K^*$ and $1v11 \in K^*$, the premise implies $x111, 1vvy \in K^*$. Hence, $x \in K^*$. Analogously, one can show $x \in K^*$ when vx is a suffix of K^*. For the 'if' part, assume (C_K) holds and consider $x_1 v_1 u_1 y_1$, $x_2 v_2 u_2 y_2 \in K^*$ with $x_1, x_2, y_1, y_2, v_1, u_1, v_2, u_2$ as specified in the premise. First we show $x_1 \in K^*$. If $v_1 \in K^+$, as $x_1 v_1$ is a prefix of K^*, part (i) implies $x_1 \in K^*$. If $v_1 = 1$ then $u_1 \in K^+$ and $x_1 u_1$ is a prefix of K^*. Hence, again $x_1 \in K^*$. Similarly one shows $x_2, y_2, y_1 \in K^*$. It follows then that $x_1 v_1 u_2 y_2$, $x_2 v_2 u_1 y_1 \in K^*$.

(iii) Let K be a splicing base which is a code. Suppose there are $v, u \in K$ and $y \in \Delta^+$ such that $u = vy$. As vy is a suffix of K^*, $y \in K^+$ which contradicts the fact that K is a code. Hence, K must be a prefix code. Similarly one verifies that K is a suffix code as well.

(iv) Let K be a comma-free code. This implies that, for all $x, y \in \Delta^*$ and for all $v \in K$, if $xvy \in K^*$ then $x, y \in K^*$ (see [1]). Now suppose xv is a prefix of K^* and uy is a suffix of K^*. Then, there are $x_1, y_1 \in \Delta^*$ such that xvy_1, $x_1uy \in K^*$. This implies $x, y \in K^*$ and, therefore, (C_K) holds.

(v) The statement follows from part (iv) and the fact that every comma-free code is bifix and uniformly synchronous with delay 1. $\diamond$

We define now a special type of splicing systems that involve only symbols of the alphabet Δ.

Definition 6 *Let K be a splicing base. A K-based splicing system is a quadruple $\beta = (K, T, A, R)$ such that $T \subseteq K$, A is a multiset with* $\mathrm{supp}(A) \subseteq K^*$, *and R is a subset of $K^*\#K^*\$K^*\#K^*$ with the property that $v_1\#u_1\$v_2\#u_2 \in R$ implies $|v_1u_1| > 0$ and $|v_2u_2| > 0$.*

For two multisets M and M', the relationship $M \Longrightarrow_\beta M'$ is defined as in the case of ordinary splicing systems – see [9]; that is, M' is obtained from M by applying a splicing rule to two words in $\mathrm{supp}(M)$. The language $L(\beta)$ *generated* by a K-based splicing system β is the set $\{w \in T^* \mid \exists M, A \Longrightarrow_\beta^* M$ and $w \in \mathrm{supp}(M)\}$. The *computation language* of the system β is the set $\{w \in \Delta^* \mid \exists M, A \Longrightarrow_\beta^* M$ and $w \in \mathrm{supp}(M)\}$. As shown next, this language is a subset of K^*.

Proposition 13 *For every splicing base K and for every K-based splicing system β, the computation language of β is a subset of K^*.*

Proof. The statement follows easily by induction when we note that $\mathrm{supp}(A) \subseteq K^*$ and that if $M \Longrightarrow_\beta M'$ and $\mathrm{supp}(M) \subseteq K^*$ then also $\mathrm{supp}(M') \subseteq K^*$ using the fact that K is a splicing base. $\diamond$

Proposition 14 *Let θ be a morphic or antimorphic involution and let β be a K-based splicing system, where K is a splicing base. If K is strictly θ-free then the computation language of β is strictly θ-free.*

Proof. Follows easily from Proposition 1 and Proposition 13. $\diamond$

In the rest of the section we consider the question of whether an arbitrary splicing system γ can be translated to a K-based splicing system β over Δ such that the computation language of β is strictly θ-free and the languages generated by β and γ are isomorphic.

Definition 7 *Let θ be a morphic or antimorphic involution of Δ^* and let $\mathbf{K}$ be an infinite family of splicing bases such that every splicing base in $\mathbf{K}$ is a strictly θ-free code. The involution-free splicing class $\mathbf{C}_{\theta,\mathbf{K}}$ is the set of all K-based splicing systems, where $K \subseteq K'$ for some $K' \in \mathbf{K}$.*

Proposition 15 *Let $\mathbf{C}_{\theta,\mathbf{K}}$ be an involution-free splicing class and let Σ be an alphabet. For every splicing system $\gamma = (\Sigma, T, A, R)$ there is a K-based splicing system $\beta = (K, F, B, P)$ in $\mathbf{C}_{\theta,\mathbf{K}}$ and an isomorphism $f : T^* \to F^*$ such that the computation language of β is strictly θ-free and $L(\beta) = f(L(\gamma))$.*

For the proof of the above statement we need the following result about $(1,1)$-free splicing systems. A splicing system (Σ, T, A, R) is called $(1,1)$-free if $\alpha_1\beta_1 \neq 1$ and $\alpha_2\beta_2 \neq 1$, for every splicing rule $\alpha_1\#\beta_1\$\alpha_2\#\beta_2$ in R.

Lemma 3 *For every splicing system γ there is a $(1,1)$-free splicing system γ' such that $L(\gamma) = L(\gamma')$.*

Proof. Let $\gamma = (\Sigma, T, A, R)$ be a splicing system. If γ is already $(1,1)$-free then $\gamma' = \gamma$. Now consider the case where γ is not $(1,1)$-free and $1 \notin L(\gamma)$. The set R contains rules $\alpha_1\#\beta_1\$\alpha_2\#\beta_2$ with $|\alpha_1\beta_1| \cdot |\alpha_2\beta_2| = 0$. Let R_1 be the set of such rules. Define a $(1,1)$-free system $\gamma' = (\Sigma, T, A, R')$ such that R' results from R by replacing every rule $s \in R_1$ with a set of rules Q_s as follows:

If $|\alpha_1\beta_1| = 0$ and $|\alpha_2\beta_2| > 0$ then $Q_s = \{1\#\delta\$\alpha_2\#\beta_2 \mid \delta \in \Delta\} \cup \{\delta\#1\$\alpha_2\beta_2 \mid \delta \in \Delta\}$; if $|\alpha_1\beta_1| > 0$ and $|\alpha_2\beta_2| = 0$ then $Q_s = \{\alpha_1\#\beta_1\$1\#\delta \mid \delta \in \Delta\} \cup \{\alpha_1\#\beta_1\$\delta\#1 \mid \delta \in \Delta\}$; if $|\alpha_1\beta_1| = |\alpha_2\beta_2| = 0$ then $Q_s = \{1\#\delta_1\$1\#\delta_2 \mid \delta_1, \delta_2 \in \Delta\} \cup \{1\#\delta_1\$\delta_2\#1 \mid \delta_1, \delta_2 \in \Delta\} \cup \{\delta_1\#1\$1\#\delta_2 \mid \delta_1, \delta_2 \in \Delta\} \cup \{\delta_1\#1\$\delta_2\#1 \mid \delta_1, \delta_2 \in \Delta\}$.

It should be clear that, for every $s \in R_1$, if a splicing rule $r \in Q_s$ is applied to a pair of words (w_1, w_2) to obtain (z_1, z_2) then also the rule s can be applied to (w_1, w_2) to obtain (z_1, z_2). And conversely, for every $s \in R_1$, if s is applied to a pair of non-empty words (w_1, w_2) to obtain (z_1, z_2), then a rule $r \in Q_s$ can be applied to (w_1, w_2) to obtain (z_1, z_2).

Now let $A \Longrightarrow_\gamma^* M \Longrightarrow_\gamma N$ be a computation of γ in which some rule $s \in R$ is used in the step $M \Longrightarrow_\gamma N$, and assume $A \Longrightarrow_{\gamma'}^* M$ is a computation of γ'. If $s \notin R_1$ then $s \in R'$ and, therefore, $M \Longrightarrow_{\gamma'} N$. If $s \in R_1$ then, as $1 \notin \operatorname{supp}(M)$, the rule s applies to a pair (w_1, w_2) of non-empty words of M. Then also a rule $r \in Q_s$ can be applied to (w_1, w_2) to obtain N. Hence, again $M \Longrightarrow_{\gamma'} N$. Thus, $A \Longrightarrow_{\gamma'}^* M \Longrightarrow_{\gamma'} N$ is a valid computation of γ' which implies $L(\gamma) \subseteq L(\gamma')$. Similarly, one verifies that $L(\gamma') \subseteq L(\gamma)$ as well.

Finally, for the case where $1 \in L(\gamma)$ we note that there is a splicing system $\zeta = (\Sigma', F, B, P)$ with $L(\zeta) = L(\gamma) \setminus \{1\}$. Hence, also $L(\zeta) = L(\zeta')$ for some $(1,1)$-free splicing system $\zeta' = (\Sigma', F, B, P')$. Then, the splicing system $\gamma' = (\Sigma', F, B', P')$ with $B' = B \cup \{1\}$ is $(1,1)$-free as well and $L(\gamma') = L(\gamma)$. $\diamond$

Let $h : S \to T$ be a bijection, where S and T are sets of words. If M is a multiset with $\operatorname{supp}(M) \subseteq S$ then $h(M)$ is the multiset with $\operatorname{supp} h(M) \subseteq T$ and $h(M)(w) = M(h^{-1}(w))$ for all $w \in T$; that is, the number of copies of the word w in $h(M)$ is equal to the number of copies of the word $h^{-1}(w)$ in M.

Proof of Proposition 15. Let $\mathbf{C}_{\theta,\mathbf{K}}$ be an involution-free splicing class and consider any splicing system $\gamma' = (\Sigma', T, A', R')$. By Lemma 3, there is a $(1,1)$-free splicing system $\gamma = (\Sigma, T, A, R)$ such that $L(\gamma') = L(\gamma)$. As $\mathbf{K}$ is infinite, there is $K_1 \in \mathbf{K}$ with $|K_1| \geq |\Sigma|$. Hence, there is a subset K of K_1 of the same cardinality as Σ. As K is a code, there is an isomorphism h of Σ^* onto K^*. We agree to extend h to $(\Sigma \cup \{\#, \$\})^*$ such that $h(\#) = \#$ and $h(\$) = \$$. Now define a K-based splicing system $\beta = (K, F, B, P)$ as follows: $B = h(A)$, $F = h(T)$,

$P = h(R)$. As K is strictly θ-free, Proposition 14 implies that the computation language of β is strictly θ-free. Now consider the restriction $f : T^* \to F^*$ of h on T^*. Then f is an isomorphism. It remains to show that $L(\beta) = f(L(\gamma))$. First we consider the following claims:

Claim 1: for every non-negative integer n, if $B = N_0 \Longrightarrow_\beta \cdots \Longrightarrow_\beta N_n$ is a computation of β then there is a computation $A = M_0 \Longrightarrow_\gamma \cdots \Longrightarrow_\gamma M_n$ of γ such that $N_i = h(M_i)$ for $i = 0, \dots, n$.

Claim 2: For every non-negative integer n, if $A = M_0 \Longrightarrow_\gamma \cdots \Longrightarrow_\gamma M_n$ is a computation of γ then $B = h(M_0) \Longrightarrow_\beta \cdots \Longrightarrow_\beta h(M_n)$ is a computation of β.

We only prove Claim 1; the proof of Claim 2 is easier. As $B = h(A)$, the claim holds for $n = 0$. Now assume the claim holds for some $n \geq 0$ and consider a computation $B = N_0 \Longrightarrow_\beta \cdots \Longrightarrow_\beta N_n \Longrightarrow_\beta N_{n+1}$ of B of length $n + 1$. By the induction hypothesis, there is a computation $A = M_0 \Longrightarrow_\gamma \cdots \Longrightarrow_\gamma M_n$ of γ with $h(M_i) = N_i$. We need to show $M_n \Longrightarrow_\gamma M_{n+1}$ for some multiset M_{n+1} with $h(M_{n+1}) = N_{n+1}$. As $N_n \Longrightarrow_B N_{n+1}$, there is a sufficient number of copies of two words $x_1v_1u_1y_1$ and $x_2v_2u_2y_2$ in N_n and a rule $v_1\#u_1\$v_2\#u_2 \in P$ such that a copy of each $x_1v_1u_2y_2$ and $x_2v_2u_1y_1$ is added in N_{n+1}. As $v_1u_1 \in K^+$ and $x_1v_1u_1y_1 \in K^+$ and K is a splicing base, $x_1 \in K^*$ and $y_1 \in K^*$. Similarly, one has $x_2, y_2 \in K^*$. As $h : \Sigma^* \to K^*$ is an isomorphism, there are words $p_1, p_2, s_1, s_2, \alpha_1, \alpha_2, \beta_1, \beta_2 \in \Sigma^*$ such that $x_i = h(p_i)$, $y_i = h(s_i)$, $\alpha_i = h(v_i)$, and $\beta_i = h(u_i)$ for $i = 1, 2$. Also, as $P = h(R)$, $\alpha_1\#\beta_1\$\alpha_2\#\beta_2 \in R$. Moreover, as $N_n(x_1v_1u_1y_1) = M_n(p_1\alpha_1\beta_1s_1)$ and $N_n(x_2v_2u_2y_2) = M_n(p_2\alpha_2\beta_2s_2)$ it follows that $M_n \Longrightarrow_\gamma M_{n+1}$ with $N_{n+1} = h(M_{n+1})$, which proves the claim.

Finally we show $L(\beta) = f(L(\gamma))$. As $L(\gamma) \subseteq T^*$ and $f(w) = h(w)$ for all w in $L(\gamma)$, it is sufficient to show $L(\beta) = h(L(\gamma))$. Let $w \in L(\beta)$; then there is a computation $B \Longrightarrow_\beta^* N$ with $w \in F^* \cap \operatorname{supp}(N)$. Claim 1 implies that there is a computation $A \Longrightarrow_\gamma^* M$ with $N = h(M)$. As $N(w) = M(h^{-1}(w))$ and $N(w) > 0$, $h^{-1}(w) \in \operatorname{supp}(M)$. As $F = h(T)$ and $w \in F^*$, $h^{-1}(w) \in T^*$. Hence, $h^{-1}(w) \in L(\gamma)$ which gives $w \in h(L(\gamma))$. Hence, $L(\beta) \subseteq h(L(\gamma))$. Analogously, one shows $h(L(\gamma)) \subseteq L(\beta)$. $\diamond$

We close the section with a simple construction of an infinite comma-free code, $K_{m,\infty}$, which is strictly free for the DNA involution τ. By Lemma 2 *(iv)*, the code is a splicing base as well. Moreover, we define an infinite family $\{K_{m,n} \mid n \in \mathbf{N}\}$ of finite subsets of $K_{m,\infty}$ whose information rate tends to $(1 - 1/m)$. The involution τ and the family $\{K_{m,n} \mid n \in \mathbf{N}\}$ define a τ-free splicing class which can be used to 'simulate' every splicing system according to Proposition 15. The fact about the information rate of the codes $K_{m,n}$ concerns the efficiency (or redundancy) of these codes when used to represent information. In general, the information rate of a finite code K over some alphabet X is the ratio

$$\frac{\log_{|X|} |K|}{(\sum_{v \in K} |v|)/|K|}$$

When that rate is close to 1, the code is close to optimal.

Proposition 16 *Let m and n be non-negative integers with $m > 1$, let $K_{m,\infty} = A^mT(\Delta^{m-1}T)^*C^m$, and let $K_{m,n} = \cup_{i=0}^{n} A^mT(\Delta^{m-1}T)^iC^m$. Then,*

(i) $K_{m,\infty}$ and, therefore, $K_{m,n}$ are comma-free codes and strictly τ-free, where τ is the DNA involution

(ii) The information rate of $K_{m,n}$ tends to $(1 - 1/m)$ as $n \to \infty$.

Proof. (i) First, suppose $K_{m,\infty}$ is not a comma-free code. Then, there are three codewords, not necessarily distinct, $A^mTx_1T\cdots x_pTC^m$, $A^mTy_1T\cdots y_qTC^m$ and $A^mTz_1T\cdots z_rTC^m$, and two non-empty words u_1, u_2 such that

$$u_1A^mTz_1T\cdots z_rTC^mu_2 = A^mTx_1T\cdots x_pTC^mA^mTy_1T\cdots y_qTC^m.$$

Let w be $u_1A^mTz_1T\cdots z_rTC^mu_2$. There are exactly two occurences of C^m in w and two occurences of A^m in w. Also, as $|u_2| > 0$ and $w \in \Delta^*C^mu_2 \cap \Delta^+C^mA^mTy_1T\ldots y_qTC^m$, it follows that $u_2 = A^mTy_1T\cdots y_qTC^m$. Analogously, it follows that $u_1 = A^mTx_1T\cdots x_pTC^m$. Then, $|A^mTz_1T\cdots z_rTC^m| = 0$ which is a contradiction. Hence, $K_{m,\infty}$ is a comma-free code.

Now suppose $K_{m,\infty}$ is not strictly τ-free. Again there are three codewords as above and two, possibly empty, words u_1 and u_2 such that

$$u_1G^mA\tau(z_r)A\cdots A\tau(z_1)AT^mu_2 = A^mTx_1T\cdots x_pTC^mA^mTy_1T\cdots y_qTC^m.$$

This implies that G^m occurs in $A^mTx_1T\cdots x_pTC^mA^mTy_1\cdots y_qTC^m$ which is impossible.

(ii) Let $p = 4^{m-1}$ and let $M_i = A^mT(\Delta^{m-1}T)^iC^m$. As $K_{m,n} = \bigcup_{i=0}^{n} M_i$ and $|M_i| = p^i$, one has $|K_{m,n}| = \sum_{i=0}^{n} p^i$ which implies $|K_{m,n}| = (p^{n+1} - 1)/(p - 1)$. Now, for each $w \in M_i$, $|w| = im+2m+1$. Also, as $\sum_{w\in M_i} |w| = p^i(im+2m+1)$, it follows that

$$\sum_{w\in K_{m,n}} |w| = \sum_{i=0}^{n}\sum_{w\in M_i} |w| = (2m+1)\frac{p^{n+1}-1}{p-1} + m\sum_{i=0}^{n} ip^i.$$

But

$$\sum_{i=0}^{n} ip^i = (p + p^2 + \cdots + p^n) + (p^2 + \cdots + p^n) + \cdots + (p^{n-1} + p^n) + p^n$$

$$= \sum_{l=1}^{n}\sum_{i=l}^{n} p^i = \sum_{l=1}^{n} p^l\frac{p^{n-l+1}-1}{p-1} = \frac{p}{p-1}(np^n - \frac{p^n-1}{p-1}).$$

Hence, the average word length of $K_{m,n}$ is $(\sum_{w\in K_{m,n}} |w|)/|K_{m,n}| =$

$$(2m+1)+\frac{mp}{p^{n+1}-1}(np^n-\frac{p^n-1}{p-1}) = (2m+1)+\frac{p^{n+1}}{p^{n+1}-1}(mn)-\frac{mp(p^n-1)}{(p^{n+1}-1)(p-1)}.$$

As $n \to \infty$, one has $(\sum_{w\in K_{m,n}} |w|)/|K_{m,n}| \sim mn$. Also, $\log_4 |K_{m,n}| = \log_4(p^{n+1} - 1) - \log_4(p - 1) \sim n\log_4 p = n(m - 1)$, and the claim follows. $\diamond$

References

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

[2] R. Deaton, R. Murphy, M. Garzon, D.R. Franceschetti, S.E. Stevens. Good encodings for DNA-based solutions to combinatorial problems. *DNA-based computers II*, in AMS DIMACS Series, vol.44, L.F.Landweber, E.Baum Eds., 1998, 247-258.

[3] 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.

[4] M. Garzon, R. Deaton, P. Neathery, D.R. Franceschetti, R.C. Murphy. A new metric for DNA computing. Proc. *2nd Genetic Programming Conference*, Stanford, CA, 1997, Morgan-Kaufmann, 472-478.

[5] 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, 684-690.

[6] T. Harju, J. Karhumäki. Morphisms. In *Handbook of Formal Languages,* vol. 1, G. Rozenberg, A. Salomaa, Eds., Springer Verlag, Berlin, 1997, 439–510.

[7] T.Head. Formal language theory and DNA: an analysis of the generative capacity of recombinant behaviors. *Bulletin of Mathematical Biology*, 49(1987) 737–759.

[8] H.Jürgensen, S.Konstantinidis. Codes. In *Handbook of Formal Languages*, vol.1, G.Rozenberg, A.Salomaa, Eds., Springer Verlag, Berlin, 1997, 511-607.

[9] L.Kari. DNA computing: arrival of biological mathematics. *The Mathematical Intelligencer*, vol.19, nr.2, Spring 1997, 9–22.

[10] L.Kari, R.Kitto, G.Thierrin. Codes, involutions and DNA encoding. Workshop on Coding Theory, London, Ontario, July 2000. To appear.

[11] A.Marathe, A.Condon, R.Corn. On combinatorial DNA word design. *DNA based Computers V*, DIMACS Series, E.Winfree, D.Gifford Eds., AMS Press, 2000, 75-89.

[12] G.Paun, G.Rozenberg, A.Salomaa. *DNA Computing: New Computing Pradigms*, Springer Verlag, Berlin, 1998.

[13] H.J.Shyr, *Free Monoids and Languages*, Hon Min Book Company, Taichung, Taiwan, R.O.C., 1991.