Generating DNA Code Words Using Forbidding and Enforcing Systems

Daniela Genova[1] and Kalpana Mahalingam[2]

[1] Department of Mathematics and Statistics
University of North Florida
Jacksonville, FL 32224, USA
d.genova@unf.edu

[2] Department of Mathematics
Indian Institute of Technology
Chennai, TN 600036, India
kmahalingam@iitm.ac.in

Abstract. Research in DNA computing was initiated by Leonard Adleman in 1994 when he solved an instance of an NP-complete problem solely by molecules. DNA code words arose in the attempt to avoid unwanted hybridizations of DNA strands for DNA based computations. Given a set of constraints, generating a large set of DNA strands that satisfy the constraints is an important problem in DNA computing. On the other hand, motivated by the non-determinism of molecular reactions, A. Ehrenfeucht and G. Rozenberg introduced forbidding and enforcing systems (fe-systems) as a model of computation that defines classes of languages based on two sets of constraints. We attempt to establish a connection between these two areas of research in natural computing by characterizing a variety of DNA codes that avoid certain types of cross hybridizations by fe-systems. We show that one fe-system can generate the entire class of DNA codes of a certain property, for example θ-k-codes, and confirm some properties of DNA codes through fe-systems. We generalize by fe-systems some known methods of generating good DNA code words which have been tested experimentally.

Keywords: fe-systems, fe-families, Biomolecular computing, Watson-Crick involution, Hybridization, DNA codes.

1 Introduction

In 1994, Adleman [1] solved an instance of an NP-problem by encoding information into DNA strands and using these strands to perform the computation by themselves. Since then, research on DNA computing has had a tremendous growth. Problems are solved by encoding information on DNA strands. One of the challenges is choosing the right set of strands for computational purposes, as the strands may bind to each other in undesirable ways due to Watson-Crick complementarity. This type of hybridization can occur during a polymerase chain reaction, a self-assembly step, or in the extraction process. Several authors have addressed this issue and proposed various solutions [3, 6, 17, 18]. Generating a large set of DNA code words is a difficult problem. Recently, in [8], a large set of DNA code words was designed using DNA metric spaces and it was shown that such a problem is NP-complete.

In [13], Kari et al. introduced a theoretical approach to the problem of designing code words. Following this approach, a DNA code that avoids all kinds of unwanted partial bindings was introduced in [14] and was called a θ-k-code (see Figure 1). Note that for any word w over the alphabet $\Delta = \{A, C, G, T\}$ and for an antimorphic involution θ such that $\theta(A) = T$, $\theta(C) = G$ and vice versa, $\theta(w)$ denotes the Watson-Crick complement of the strand w.

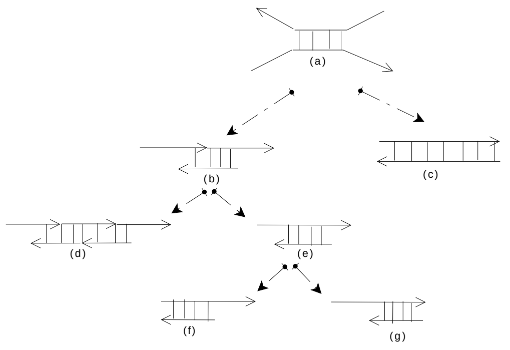

Fig. 1. Various cross hybridizations of molecules avoided by: (a): θ-k-code - one molecule contains a subword of length k and the other its complement. For a suitable k, a θ-k-code also avoids cross bindings of type avoided by (b): θ-comma-free, (c): θ-strict, (d): θ-intercode, (e): θ-infix, (f): θ-prefix, (g): θ-suffix code.

Motivated by the non-determinism in molecular reactions, A. Ehrenfeucht and G. Rozenberg introduced forbidding-enforcing systems, fe-systems, in [4, 5] as a model of computation that defines classes of languages. This model uses a forbidding set to exclude combinations of subwords from the subwords of each language in the family and uses an enforcing set to require certain words to be in the language, provided some pre-specified sets of words are already contained in the language. One fe-system consisting of a forbidding set and an enforcing set defines a class of languages, fe-family. These classes were shown in [11] to be completely different than Chomsky's classes. On the other hand, fe-systems were shown to be suitable for defining the solutions to combinatorial problems and for modeling DNA molecules and splicing with an enzyme [5, 9, 12]. Fe-systems models in membrane computing were introduced in [2] and [7] proposed modeling self-assembly of graphs by fe-systems.

The purpose of this paper is to show that the theory of forbidding and enforcing systems is very suitable for generating DNA codes and to initiate research at the interplay of both areas. We use the fe-systems defined in [4, 5] to characterize the types of DNA codes presented in [13, 14, 16]. We show that one can generate the entire family of DNA codes that avoid certain unwanted cross-hybridizations by a single fe-system, as opposed to classical language theory where one grammar (automaton) generates (accepts) only one language (DNA code). In the process, we also confirm some properties of DNA codes, deriving them through fe-systems. The definitions are recalled in Section 2 and various DNA codes characterizations are presented in Section 3. In Section 4, as an illustration, we present fe-systems that characterize DNA code words generated using methods proposed in [14]. These code words were experimentally tested in [14] and were shown to have no visible cross-hybridizations.

2 Basic Concepts and Definitions

An alphabet is denoted by Σ, the length of a word w over Σ by $|w|$ and the empty word by λ. The free monoid Σ^* contains all words over Σ and the free semigroup $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. For $k \geq 1$, $\Sigma^k = \{w \in \Sigma^* \mid |w| = k\}$ and $\Sigma^{\leqslant k} = \{w \in \Sigma^* \mid |w| \leq k\}$. The set of all languages over Σ is $\mathcal{P}(\Sigma^*)$. For $w \in \Sigma^*$, the set of subwords of w is $\text{Sub}\,(w)$ and $\text{Sub}_k(w) = \text{Sub}\,(w) \cap \Sigma^k$. Extended to languages, $\text{Sub}\,(L)$ denotes all subwords of the language L.

2.1 DNA Involution Codes

We follow the definitions from [13, 14, 16, 15]. An involution $\theta : \Sigma \to \Sigma$ of a set Σ is a mapping such that θ^2 equals the identity mapping, i.e. $\theta(\theta(x)) = x$, for all $x \in \Sigma$. For words $u, v \in \Sigma^*$ and morphic θ we have that $\theta(uv) = \theta(u)\theta(v)$ and for an antimorphic θ, $\theta(uv) = \theta(v)\theta(u)$.

Definition 1. Given an alphabet Σ, let $\theta : \Sigma^* \to \Sigma^*$ be a morphic or an antimorphic involution and $X \subseteq \Sigma^+$. Then the set (language) X is called a:

1. *θ-subword-k-m-code* for some positive integers k and m if for all $u \in \Sigma^k$ we have $\Sigma^* u \Sigma^i \theta(u) \Sigma^* \cap X = \emptyset$ for all $1 \leq i \leq m$.
2. *θ-subword-k-code* for some positive integer k if for all $u \in \Sigma^k$ we have $\Sigma^* u \Sigma^i \theta(u) \Sigma^* \cap X = \emptyset$ for all $i \geq 1$.
3. *θ-strict-code* if $X \cap \theta(X) = \emptyset$.
4. *θ-prefix-code* if $X \cap \theta(X)\Sigma^+ = \emptyset$.
5. *θ-suffix-code* if $X \cap \Sigma^+\theta(X) = \emptyset$.
6. *θ-bifix-code* if X is both a θ-prefix-code and a θ-suffix-code.
7. *θ-intercode of index m* for some integer $m \geq 1$ if $X^{m+1} \cap \Sigma^+\theta(X^m)\Sigma^+ = \emptyset$.
8. *θ-infix-code* if $\Sigma^*\theta(X)\Sigma^+ \cap X = \emptyset$ and $\Sigma^+\theta(X)\Sigma^* \cap X = \emptyset$.
9. *θ-comma-free-code* if $X^2 \cap \Sigma^+\theta(X)\Sigma^+ = \emptyset$.
10. *θ-k-code* for some integer $k > 0$ if $\text{Sub}_k(X) \cap \text{Sub}_k(\theta(X)) = \emptyset$.

A set $X \subseteq \Sigma^+$ is said to be a *θ-strict-P-code* if X is both a θ-P-code and a θ-strict-code, where $P \in$ {prefix, suffix, infix, bifix, comma-free, intercode, k-(code)}.

Example 2. Let $X = \{aa, baa\}$ be a language over the alphabet $\Sigma = \{a, b\}$ and θ be a morphic involution on Σ with $\theta(a) = b$ and $\theta(b) = a$. Then $\theta(X) = \{bb, abb\}$. Note that X is a θ-infix-code since none of the subwords of X are in $\theta(X)$ and X is a θ-comma-free-code since $X^2 = \{a^4, a^2ba^2, ba^4, ba^2ba^2\}$ and none of the words in $\theta(X)$ appears as a subword of any word in X^2. Also, note that $\mathrm{Sub}_3(X) \cap \mathrm{Sub}_3(\theta(X)) = \emptyset$. Hence, X is a θ-3-code. The word $baa = ba\theta(b)$ and hence X is not a θ-subword-1-code but there is no word of the type $uxv\theta(x)w$ in X with $u, w \in \Sigma^*$, $v \in \Sigma^+$ and $|x| = 2$. Thus, X is a θ-subword-2-code.

2.2 Forbidding-Enforcing Systems

Similar to grammars and automata, forbidding-enforcing systems can be used as a language defining tool. Unlike grammars and automata, they define languages based on the principle that "everything that is not forbidden is allowed" as opposed to "everything that is not allowed is forbidden". In this paper, we use the model of fe-systems introduced in [4, 5] where one fe-system defines a family of languages. The relevant definitions are recalled below.

Definition 3. A *forbidding set* is a family of finite nonempty subsets of Σ^+ for some alphabet Σ; each element of a forbidding set is called a *forbidder.*

Given a forbidding set $\mathcal{F}$ and a language K: (i) for $F \in \mathcal{F}$, we say that K *is consistent with the forbidder* F, written $K\, con\, F$, if and only if $F \nsubseteq \mathrm{Sub}(K)$ and (ii) we say that K *is consistent with the forbidding set* $\mathcal{F}$, denoted $K\, con\, \mathcal{F}$, if and only if $K\, con\, F$ for each $F \in \mathcal{F}$.

The family of languages consistent with the forbidding set $\mathcal{F}$ is denoted by $\mathcal{L}(\mathcal{F})$, i.e. $\mathcal{L}(\mathcal{F}) = \{K \mid K\, con\, \mathcal{F}\}$. The class (family) of languages $\mathcal{L}(\mathcal{F})$ is an *f-family* and is said to be *defined by* $\mathcal{F}$.

If a language $L \notin \mathcal{L}(\mathcal{F})$ then L *is not consistent with* $\mathcal{F}$, denoted by $L\, ncon\, \mathcal{F}$.

Note that if $K\, con\, \mathcal{F}$, every subset of K is also consistent with $\mathcal{F}$ [5]. Also, $\mathcal{F} = \emptyset$ if and only if $\mathcal{L}(\mathcal{F}) = \mathcal{P}(\Sigma^*)$.

Example 4. Let $\Sigma = \{a, b\}$. For the forbidding set $\mathcal{F} = \{\{ab, ba\}, \{aa, bb\}\}$ from [5], the languages in $\mathcal{L}(\mathcal{F})$ are precisely the subsets of the languages $K_1 = a^*b \cup a^*$, $K_2 = ba^* \cup a^*$, $K_3 = b^*a \cup b^*$, and $K_4 = ab^* \cup b^*$.

Definition 5. An *enforcing set* is a family of ordered pairs (X, Y) such that for some alphabet Σ all $X, Y \subseteq \Sigma^+$, X, Y are finite, and each $Y \neq \emptyset$; each element of an enforcing set is called an *enforcer.*

Given an enforcing set $\mathcal{E}$ over Σ and a language K: (i) for $(X, Y) \in \mathcal{E}$ we say that a language K *satisfies the enforcer* (X, Y), written $K\, sat\, (X, Y)$, if and only if $X \subseteq K$ implies $Y \cap K \neq \emptyset$ and (ii) we say that K *satisfies the enforcing set* $\mathcal{E}$, written $K\, sat\, \mathcal{E}$, if and only if $K\, sat\, (X, Y)$ for each $(X, Y) \in \mathcal{E}$.

The class (family) of all languages that satisfy the enforcing set $\mathcal{E}$ is denoted by $\mathcal{L}(\mathcal{E})$, i.e. $\mathcal{L}(\mathcal{E}) = \{K \mid K\, sat\, \mathcal{E}\}$. The class of languages $\mathcal{L}(\mathcal{E})$ is an *e-family* and is said to be *defined by* $\mathcal{E}$.

If a language $L \notin \mathcal{L}(\mathcal{E})$, then L *does not satisfy* $\mathcal{E}$, denoted by $L\, nsat\, \mathcal{E}$.

For enforcers $(\emptyset, Y)$, $K \cap Y \neq \emptyset$ for all $K \in \mathcal{L}(\mathcal{E})$. Assume that all $(X, Y) \in \mathcal{E}$ are non-trivial, i.e. $X \cap Y = \emptyset$. Thus, $\mathcal{L}(\mathcal{E}) = \mathcal{P}(\Sigma^*)$ if and only if $\mathcal{E} = \emptyset$.

Example 6. Let $\Sigma = \{a, b\}$ and $\mathcal{E} = \{(\emptyset, \{b\}), (\{b\}, \{b^2\}), \ldots, (\{b^n\}, \{b^{n+1}\}), \ldots\}$. Then, $\mathcal{E}$ "enforces" b^* in every language in $\mathcal{L}(\mathcal{E})$, i.e. $\mathcal{L}(\mathcal{E}) = \{K \mid b^* \subseteq K\}$.

Definition 7. A *forbidding-enforcing system* (fe-system) over an alphabet Σ is a construct $(\mathcal{F}, \mathcal{E})$, where $\mathcal{F}$ is a forbidding set and $\mathcal{E}$ is an enforcing set over Σ. The class of languages $\mathcal{L}(\mathcal{F}, \mathcal{E})$, the *fe-family*, defined by this fe-system consists of all languages that are consistent with $\mathcal{F}$ and satisfy $\mathcal{E}$, i.e $\mathcal{L}(\mathcal{F}, \mathcal{E}) = \mathcal{L}(\mathcal{F}) \cap \mathcal{L}(\mathcal{E})$.

Example 8. For $\Sigma = \{a, b\}$, $\mathcal{F}$, K_3, and K_4 from Example 4 and $\mathcal{E}$ from Example 6, we have that $\mathcal{L}(\mathcal{F}, \mathcal{E}) = \{K \mid K \subseteq K_3, K_4 \text{ and } b^* \subseteq K\}$.

Observe that $\mathcal{L}(\emptyset, \mathcal{E}) = \mathcal{L}(\mathcal{E})$ for all $\mathcal{E}$ and $\mathcal{L}(\mathcal{F}, \emptyset) = \mathcal{L}(\mathcal{F})$ for all $\mathcal{F}$. In this respect, every f-family and every e-family can be regarded as an fe-family.

3 Characterizing Involution Codes with fe-Systems

One benefit of studying DNA codes with fe-systems is that one fe-system can define an entire class of DNA codes $\mathcal{L}(\mathcal{F}, \mathcal{E})$ as opposed to just one DNA code X that has been the norm in constructing/studying DNA code words using classical formal language theory.

3.1 Characterizations by f-Families

We begin with a finite forbidding set that defines the entire class of θ-subword-k-m-codes, i.e. for given positive integers k and m one can construct a forbidding set $\mathcal{F}$ such that any language L in the f-family $\mathcal{L}(\mathcal{F})$ is a θ-subword-k-m-code and any θ-subword-k-m-code is in the f-family $\mathcal{L}(\mathcal{F})$.

For the rest of this paper, assume that the alphabet Σ is given and θ is a morphic or an antimorphic involution on Σ extended to Σ^*, unless stated otherwise. Also, to ease notation, by $L \in \mathcal{L}(\mathcal{F})$ (resp. $L \in \mathcal{L}(\mathcal{E})$, $L \in \mathcal{L}(\mathcal{F}, \mathcal{E})$) we mean those L for which $\lambda \notin L$ and $L \neq \emptyset$.

Proposition 9. *Let $k, m \geq 1$ be integers. For every $u \in \Sigma^k$ construct $\mathcal{F}_u = \{\{uw\theta(u)\} \mid w \in (\Sigma^{\leqslant m} \setminus \{\lambda\})\}$ and let $\mathcal{F} = \cup_{u \in \Sigma^k} \mathcal{F}_u$. Then L is a θ-subword-k-m-code if and only if $L \in \mathcal{L}(\mathcal{F})$.*

Proof. Note that L is a θ-subword-k-m-code if and only if $\Sigma^* u \Sigma^i \theta(u) \Sigma^* \cap L = \emptyset$ for all $1 \leq i \leq m$, which holds if and only if no word in L has a subword of the kind $uw\theta(u)$, where $u \in \Sigma^k$ and $w \in (\Sigma^{\leqslant m} \setminus \{\lambda\})$, i.e. if and only if $L \, \mathit{con} \, F$ for every $F \in \mathcal{F}$, i.e. if and only if $L \in \mathcal{L}(\mathcal{F})$. $\square$

The next result presents a way to construct a forbidding set that defines the entire class of θ-subword-k-codes.

Proposition 10. *Let $k \geq 1$ be an integer. For every $u \in \Sigma^k$ construct $\mathcal{F}_u = \{\{uw\theta(u)\} \mid w \in \Sigma^+\}$ and let $\mathcal{F} = \cup_{u\in\Sigma^k}\mathcal{F}_u$. Then L is a θ-subword-k-code if and only if $L \in \mathcal{L}(\mathcal{F})$.*

Proof. Observe that L is a θ-subword-k-code if and only if $\Sigma^* u \Sigma^i \theta(u)\Sigma^* \cap L = \emptyset$ for all $u \in \Sigma^k$ and for all $i \geq 1$, which holds if and only if no word in L has a subword of the kind $uw\theta(u)$, where $u \in \Sigma^k$ and $w \in \Sigma^+$, i.e. if and only if $L\,con\,F$ for every forbidder $F \in \mathcal{F}$, i.e. if and only if $L \in \mathcal{L}(\mathcal{F})$. $\square$

Proposition 10 uses an infinite forbidding set. The next result uses a finite forbidding set, but provides only a sufficient condition for θ-subword-k-codes.

Proposition 11. *Let $k \geq 1$ be an integer. Consider $\mathcal{F} = \{\{u, \theta(u)\} \mid u \in \Sigma^k\}$. Then for every $L \in \mathcal{L}(\mathcal{F})$, L is a θ-subword-k-code.*

Proof. Assume that $L \in \mathcal{L}(\mathcal{F})$. Then, $\{u, \theta(u)\} \not\subseteq \text{Sub}\,(L)$ for every $u \in \Sigma^k$. Hence, for every $x \in L$ and for every $u \in \Sigma^k$, x cannot contain both u and $\theta(u)$ as subwords. Thus, $\Sigma^* u \Sigma^i \theta(u) \Sigma^* \cap L = \emptyset$ for all $u \in \Sigma^k$ and for all $i \geq 1$, i.e. L is a θ-subword-k-code. $\square$

Note that the converse of the above proposition does not necessarily hold, since a θ-subword-k-code X may have a word x with a subword $u_1u_2v_2$ such that $u = u_1u_2$ and $\theta(u) = v_1v_2$ with $u_2 = v_1$ for some $u \in \Sigma^k$, i.e. u and $\theta(u)$ overlap in x. Such X will not be consistent with the forbidding set $\mathcal{F}$ from Proposition 11 as illustrated in the next example.

Example 12. Consider $\Sigma = \{a, b\}$ and a morphic θ with $\theta(a) = b$ and $\theta(b) = a$. Note that the set $X = \{aa, aba\}$ is a θ-subword-2-code by Definition 1. Construct the forbidding set $\mathcal{F}$ from Proposition 11 for $k = 2$, namely $\mathcal{F} = \{\{aa, bb\}, \{ab, ba\}\}$. Notice that the second forbidder $\{ab, ba\} \subseteq \text{Sub}\,(X)$ since both ab and ba are in the subwords of aba. Hence, $X \notin \mathcal{L}(\mathcal{F})$.

Since the forbidding set of Example 4 is the same as $\mathcal{F}$ in Example 12, we note that every nonempty subset of K_i for $i = 1, \ldots, 4$ from Example 4, not containing λ, is a θ-subword-2-code for the morphic θ from Example 12, but $\mathcal{L}(\mathcal{F})$ does not contain all of the θ-subword-2-codes for this θ.

In the remainder of this section, the forbidding sets constructed from a given X are finite when X is a finite code.

Proposition 13. *Let $X \subset \Sigma^+$ be given and let $\mathcal{F} = \{\{u\} \mid u \in \theta(X)\}$. Then X is a θ-strict-infix-code if and only if $X \in \mathcal{L}(\mathcal{F})$.*

Proof. Consider $\mathcal{F} = \{\{u\} \mid u \in \theta(X)\}$. Note that X is a θ-strict-infix-code if and only if $\Sigma^* \theta(X) \Sigma^* \cap X = \emptyset$, which holds if and only if $\theta(X) \cap \text{Sub}\,(X) = \emptyset$, if and only if $F \not\subseteq \text{Sub}\,(X)$ for every $F \in \mathcal{F}$ if and only if $X \in \mathcal{L}(\mathcal{F})$. $\square$

The next characterization follows directly from the definitions in Section 2.

Proposition 14. *Let $X \subset \Sigma^+$ be given and let $\mathcal{F}_u = \{\{ua\} \mid a \in \Sigma\} \cup \{\{au\} \mid a \in \Sigma\}$ and let $\mathcal{F} = \cup_{u\in\theta(X)}\mathcal{F}_u$. Then X is a θ-infix-code if and only if $X \in \mathcal{L}(\mathcal{F})$.*

Proof. Consider $\mathcal{F}$ as defined in the proposition. Assume X is a θ-infix-code. Then, $\Sigma^*\theta(X)\Sigma^+ \cap X = \emptyset$ and $\Sigma^+\theta(X)\Sigma^* \cap X = \emptyset$. This implies that u is not a proper prefix and not a proper suffix of a word in X and $u \notin (\mathrm{Sub}\,(X) \setminus X)$ for every $u \in \theta(X)$, i.e. for every $u \in \theta(X)$, $u \notin \mathrm{Sub}\,(X)$ unless $u = x$ for some $x \in X$. Hence, $X\, con\, F$ for every $F \in \mathcal{F}$. Therefore, $X \in \mathcal{L}(\mathcal{F})$. Conversely, assume $X \in \mathcal{L}(\mathcal{F})$. Then, for every $u \in \theta(X)$ it holds that $u \notin \mathrm{Sub}\,(X)$ unless $u = x$ for some $x \in X$. Therefore, X is a θ-infix-code. □

Proposition 15. *Let $X \subset \Sigma^+$ be given, $m \geq 1$ be an integer and $\mathcal{F} = \{\{u\} \mid u \in \theta(X^m)\}$. Then X is a θ-strict-intercode of index m if and only if $X^{m+1} \in \mathcal{L}(\mathcal{F})$.*

Proof. Let $\mathcal{F} = \{\{u\} \mid u \in \theta(X^m)\}$. Note that X is a θ-strict-intercode of index m if and only if $\Sigma^*\theta(X^m)\Sigma^* \cap X^{m+1} = \emptyset$, which holds if and only if $\theta(X^m) \cap \mathrm{Sub}\,(X^{m+1}) = \emptyset$, i.e. if and only if $X^{m+1}\, con\, F$ for every $F \in \mathcal{F}$. □

Similarly, one can define a θ-intercode by a forbidding set.

Proposition 16. *Let $X \subset \Sigma^+$ be given, $m \geq 1$ be an integer and for every $u \in \theta(X^m)$ let $\mathcal{F}_u = \{\{aub\} \mid a, b \in \Sigma\}$ and let $\mathcal{F} = \cup_{u\in\theta(X^m)}\mathcal{F}_u$. Then X is a θ-intercode of index m if and only if $X^{m+1} \in \mathcal{L}(\mathcal{F})$.*

Proof. Consider $\mathcal{F}$ as defined in the proposition. Assume X is a θ-intercode of index m. Then, $\Sigma^+\theta(X^m)\Sigma^+ \cap X^{m+1} = \emptyset$. This implies that $aub \notin \mathrm{Sub}\,(X^{m+1})$ for every $a, b \in \Sigma$, where a and b are not necessarily distinct, and every $u \in \theta(X^m)$. Hence, $X^{m+1}\, con\, F$ for every $F \in \mathcal{F}$. Therefore, $X^{m+1} \in \mathcal{L}(\mathcal{F})$. Conversely, assume $X^{m+1} \in \mathcal{L}(\mathcal{F})$. Then, for every $u \in \theta(X^m)$ and every $a, b \in \Sigma$ it holds that $aub \notin \mathrm{Sub}\,(X^{m+1})$. Therefore, $\Sigma^+\theta(X^m)\Sigma^+ \cap X^{m+1} = \emptyset$. Hence, X is a θ-intercode. □

Remark 17. Note that a θ-strict-intercode of index 1 is a θ-strict-comma-free-code and a θ-intercode of index 1 is a θ-comma-free-code.

The corollaries below follow from Propositions 15 and 16 and the above remark.

Corollary 18. *Let $X \subset \Sigma^+$ be given and let $\mathcal{F} = \{\{u\} \mid u \in \theta(X)\}$. Then X is a θ-strict-comma-free-code if and only if $X^2 \in \mathcal{L}(\mathcal{F})$.*

Corollary 19. *Let $X \subset \Sigma^+$ be given and for every $u \in \theta(X)$ construct $\mathcal{F}_u = \{\{aub\} \mid a, b \in \Sigma\}$ and let $\mathcal{F} = \cup_{u\in\theta(X)}\mathcal{F}_u$. Then X is a θ-comma-free-code if and only if $X^2 \in \mathcal{L}(\mathcal{F})$.*

We present some θ-k-codes characterizations next. Note that the forbidding set in the following proposition is finite, even if X is infinite.

Proposition 20. *Let $X \subset \Sigma^+$ and $k \geq 1$ be given and let $\mathcal{F} = \{\{u\} \mid u \in \mathrm{Sub}\,_k\theta(X)\}$. Then X is a θ-k-code if and only if $X \in \mathcal{L}(\mathcal{F})$.*

Proof. By definition, X is a θ-k-code if and only if $\mathrm{Sub}_k(X) \cap \mathrm{Sub}_k(\theta(X)) = \emptyset$, which holds if and only if $u \notin \mathrm{Sub}(X)$ for every $u \in \mathrm{Sub}_k\theta(X)$, i.e. if and only if $F \not\subseteq \mathrm{Sub}(X)$ for every $F \in \mathcal{F}$ if and only if $X \in \mathcal{L}(\mathcal{F})$. □

Observe that if X is a θ-m-code for some fixed $m \geq 1$, then X is a θ-k-code for any $k \geq m$. This fact is confirmed by the forbidding-enforcing theory, since if one constructs $\mathcal{F}_k = \{\{u\} \mid u \in \mathrm{Sub}_k\theta(X)\}$ and lets $\mathcal{F} = \cup_{k\geq m}\mathcal{F}_k$, the resulting forbidding set will not be subword incomparable and according to [4], it will be equivalent to the forbidding set from Proposition 20 for $k = m$.

While Proposition 20 is a direct "restatement" of the definition for θ-k-codes in terms of fe-systems, the following proposition presents a sufficient condition for θ-k-codes that can be used for generating θ-k-codes with specific additional restrictions on the subwords. It provides a forbidding set, which for a given k defines a family of θ-k-codes.

Proposition 21. *Given a finite alphabet Σ and a fixed $k \geq 1$ let $\Sigma^k = P \cup Q \cup R$ such that $\theta(P) = Q$ and $\theta(x) = x$ for all $x \in R$ and P, Q, and R are pairwise disjoint. Let $H = Q \cup R$ and let $\mathcal{F} = \{\{u\} \mid u \in H\}$. Then, for all $L \in \mathcal{L}(\mathcal{F})$, L is a θ-k-code.*

Proof. Assume $L \in \mathcal{L}(\mathcal{F})$. Suppose there exists $v \in Sub_k(\theta(L))$ such that $v \in \mathrm{Sub}(L)$. Then, $v \notin H$, since $\{u\} \not\subseteq \mathrm{Sub}(L)$ for every $u \in H$. Hence, $v \in P$. Therefore, $\theta(v) \in Q$. Since $v \in \mathrm{Sub}_k(\theta(L))$, there exists $y \in L$ such that $v \in \mathrm{Sub}(\theta(y))$. Thus, there exists $x \in \mathrm{Sub}(y)$ such that $v = \theta(x)$. Then, $\theta(v) = \theta(\theta(x)) = x$. Since $\theta(v) \in Q$, it follows that $\{x\} \in \mathcal{F}$ and $x \notin \mathrm{Sub}(L)$, which contradicts our supposition that $v \in \mathrm{Sub}(\theta(L))$ and hence $\theta(v) \in \mathrm{Sub}(L)$. Thus, L is a θ-k-code. □

We conclude this subsection with a necessary and sufficient condition for θ-k-codes. Observe that the forbidding set in the next proposition is finite.

Proposition 22. *Let $k \geq 1$ be an integer. Consider $\mathcal{F} = \{\{u, \theta(u)\} \mid u \in \Sigma^k\}$. Then, L is a θ-k-code if and only if $L \in \mathcal{L}(\mathcal{F})$.*

Proof. Let L be a θ-k-code and take an arbitrary forbidder $F \in \mathcal{F}$. Then $F = \{u, \theta(u)\}$ for some $u \in \Sigma^k$. Suppose $\{u, \theta(u)\} \subseteq \mathrm{Sub}(L)$. Then $u \in \mathrm{Sub}(L)$ implies $\theta(u) \in \mathrm{Sub}(\theta(L))$. Hence, $\theta(u) \in \mathrm{Sub}_k(L) \cap \mathrm{Sub}_k(\theta(L))$, which contradicts the assumption that L is a θ-k-code. Therefore, $\{u, \theta(u)\} \not\subseteq \mathrm{Sub}(L)$ and $L \in \mathcal{L}(\mathcal{F})$. Conversely, assume L is not a θ-k-code. This implies that there exists $u \in \mathrm{Sub}_k(L) \cap \mathrm{Sub}_k(\theta(L))$. Since $u \in \mathrm{Sub}_k(\theta(L))$, there exists $v \in \mathrm{Sub}_k(L)$ such that $u = \theta(v)$. Then $\theta(u) = \theta(\theta(v)) = v \in \mathrm{Sub}_k(L)$, implying $\{u, \theta(u)\} \subseteq \mathrm{Sub}(L)$. Hence, $L \notin \mathcal{L}(\mathcal{F})$. □

Example 23. Let $\Sigma = \{a, b\}$, $k = 2$, and θ be a morphic involution that maps $a \rightarrow b$ and $b \rightarrow a$. Then $\mathcal{F}$ is precisely the forbidding set from Example 4 (and from Example 12) and the θ-2-codes are precisely the subsets of the languages K_i for $i = 1, \ldots, 4$. If θ is antimorphic, $\mathcal{F} = \{\{aa, bb\}, \{ab\}, \{ba\}\}$ and every θ-2-code is either a subset of a^* or a subset of b^*. For $k = 3$ and morphic θ

with $a \to b$ and $b \to a$, $\mathcal{F} = \{\{aaa, bbb\}, \{aab, bba\}, \{aba, bab\}, \{baa, abb\}\}$ and for antimorphic θ we have $\mathcal{F} = \{\{aaa, bbb\}, \{aab, abb\}, \{aba, bab\}, \{baa, bba\}\}$. Observe that the latter $\mathcal{L}(\mathcal{F})$ contains X from Example 30.

Notice that Propositions 22 and 11 confirm the fact that every θ-k-code is a θ-subword-k-code through fe-systems.

3.2 Characterizations by e-Families

While the previous subsection showed that fe-systems with empty enforcing sets are capable of characterizing DNA codes, this subsection focuses on DNA code characterizations obtained by enforcing sets only. Recall that one can view enforcing sets as fe-systems with empty forbidding sets. We begin with a characterization of θ-strict-codes.

Proposition 24. *Let $X \subseteq \Sigma^+$ and let $z \in \Sigma^+$ such that $z \notin X$. For each $w \in X$ construct $\mathcal{E}_w = \{(\{w, u\}, \{z\}) \mid u \in \theta(X)\}$ and let $\mathcal{E} = \cup_{w \in X} \mathcal{E}_w$. Then, X is a θ-strict-code if and only if $X \in \mathcal{L}(\mathcal{E})$.*

Proof. Assume that X is a θ-strict-code. Then, $X \cap \theta(X) = \emptyset$ implies that $\{w, u\} \nsubseteq X$ for every enforcer $(\{w, u\}, \{z\}) \in \mathcal{E}$, since $u \notin X$ for all $u \in \theta(X)$. Thus, X satisfies every enforcer in $\mathcal{E}$ trivially. Therefore, $X \in \mathcal{L}(\mathcal{E})$. Conversely, assume that X is not a θ-strict-code. This implies that $X \cap \theta(X) \neq \emptyset$. Hence, there exists $y \in X \cap \theta(X)$. Thus, there exists an enforcer $(\{y\}, \{z\}) \in \mathcal{E}$ with $y \in X$ and $z \notin X$. It follows that $X \, nsat \, (\{y\}, \{z\})$. Hence, $X \notin \mathcal{L}(\mathcal{E})$. □

Proposition 25. *Let $X \subseteq \Sigma^+$ and let $z \in \Sigma^+$ such that $z \notin X$. For each $w \in X$ construct $\mathcal{E}'_w = \{(\{w, ut\}, \{z\}) \mid u \in \theta(X), t \in \Sigma^+\}$ and $\mathcal{E}''_w = \{(\{w, su\}, \{z\}) \mid u \in \theta(X), s \in \Sigma^+\}$. Let $\mathcal{E}' = \cup_{w \in X} \mathcal{E}'_w$, $\mathcal{E}'' = \cup_{w \in X} \mathcal{E}''_w$, and let $\mathcal{E} = \mathcal{E}' \cup \mathcal{E}''$. Then, the following statements hold.*

1. *X is a θ-prefix-code if and only if $X \in \mathcal{L}(\mathcal{E}')$.*
2. *X is a θ-suffix-code if and only if $X \in \mathcal{L}(\mathcal{E}'')$.*
3. *X is a θ-bifix-code if and only if $X \in \mathcal{L}(\mathcal{E})$.*

Proof. 1. Assume that X is a θ-prefix-code. Then, $X \cap \theta(X)\Sigma^+ = \emptyset$ implies that $\{w, ut\} \nsubseteq X$ for every enforcer $(\{w, u\}, \{z\}) \in \mathcal{E}'$, since $ut \notin X$ for all $u \in \theta(X)$ and all $t \in \Sigma^+$. Thus, X satisfies every enforcer in $\mathcal{E}'$ trivially. Therefore, $X \in \mathcal{L}(\mathcal{E}')$. Conversely, assume that X is not a θ-prefix-code. This implies that $X \cap \theta(X)\Sigma^+ \neq \emptyset$. Hence, there exists $y \in X \cap \theta(X)\Sigma^+$. Thus, there exists an enforcer $(\{y\}, \{z\}) \in \mathcal{E}'$ with $y \in X$ and $z \notin X$. It follows that $X \, nsat \, (\{y\}, \{z\})$. Hence, $X \notin \mathcal{L}(\mathcal{E}')$.
2. Similar to 1.
3. Follows from 1. and 2. above and the property for enforcing sets $\mathcal{L}(\mathcal{E}' \cup \mathcal{E}'') = \mathcal{L}(\mathcal{E}') \cap (\mathcal{E}'')$. □

The following result can be proved in a similar way.

Proposition 26. *Let $X \subseteq \Sigma^+$ and let $z \in \Sigma^+$ such that $z \notin X$. For each $w \in X$ construct $\mathcal{E}'_w = \{(\{w, ut\}, \{z\}) \mid u \in \theta(X), t \in \Sigma^+\}$, $\mathcal{E}''_w = \{(\{w, su\}, \{z\}) \mid u \in \theta(X), s \in \Sigma^+\}$, and $\mathcal{E}'''_w = \{(\{w, puq\}, \{z\}) \mid u \in \theta(X), p, q \in \Sigma^+\}$. Let $\mathcal{E}' = \cup_{w \in X} \mathcal{E}'_w$, $\mathcal{E}'' = \cup_{w \in X} \mathcal{E}''_w$, $\mathcal{E}''' = \cup_{w \in X} \mathcal{E}'''_w$ and let $\tilde{\mathcal{E}} = \mathcal{E}' \cup \mathcal{E}'' \cup \mathcal{E}'''$. Then, X is a θ-infix-code if and only if $X \in \mathcal{L}(\tilde{\mathcal{E}})$.*

Since, $\mathcal{E} \subseteq \tilde{\mathcal{E}}$ for $\mathcal{E}$ and $\tilde{\mathcal{E}}$ from Propositions 25 and 26 respectively, and since $\mathcal{E} \subseteq \tilde{\mathcal{E}}$ implies $\mathcal{L}(\tilde{\mathcal{E}}) \subseteq \mathcal{L}(\mathcal{E})$, the forbidding-enforcing theory confirms the known fact that every θ-infix-code is a θ-bifix-code.

Similarly, we obtain the following characterization.

Proposition 27. *Let $X \subseteq \Sigma^+$ and let $m \geq 1$ be an integer. Let $z \in \Sigma^+$ such that $z \notin X^{m+1}$. For each $w \in X^{m+1}$ construct $\mathcal{E}_w = \{(\{w, sut\}, \{z\}) \mid u \in \theta(X^m), s, t \in \Sigma^+\}$ and let $\mathcal{E} = \cup_{w \in X^{m+1}} \mathcal{E}_w$, Then, X is a θ-intercode of index m if and only if $X^{m+1} \in \mathcal{L}(\mathcal{E})$.*

The following corollary is a consequence of Proposition 27 and Remark 17.

Corollary 28. *Let $X \subseteq \Sigma^+$ and $z \in \Sigma^+$ such that $z \notin X^2$. For each $w \in X^2$ construct $\mathcal{E}_w = \{(\{w, sut\}, \{z\}) \mid u \in \theta(X), s, t \in \Sigma^+\}$ and let $\mathcal{E} = \cup_{w \in X^2} \mathcal{E}_w$. Then, X is a θ-comma-free-code if and only if $X^2 \in \mathcal{L}(\mathcal{E})$.*

In the previous subsection, some ways to present θ-k-codes as f-families were proposed. We conclude this section with characterizing θ-k-codes by enforcing sets only and present an example. Note that the enforcing set in the next proposition is finite, even if X is infinite.

Proposition 29. *Let $X \subseteq \Sigma^+$ and $k > 0$ be an integer. Let $z \in \Sigma^+$ such that $z \notin \mathrm{Sub}_k(X)$. For each $w \in \mathrm{Sub}_k(X)$ construct $\mathcal{E}_w = \{(\{w, u\}, \{z\}) \mid u \in \mathrm{Sub}_k(\theta(X))\}$ and let $\mathcal{E} = \cup_{w \in \mathrm{Sub}_k(X)} \mathcal{E}_w$. Then, X is a θ-k-code if and only if $\mathrm{Sub}_k(X) \in \mathcal{L}(\mathcal{E})$.*

Proof. Assume X is a θ-k-code. Then, $\mathrm{Sub}_k(X) \cap \mathrm{Sub}_k(\theta(X)) = \emptyset$ implies that for every $u \in \mathrm{Sub}_k(\theta(X))$, $u \notin \mathrm{Sub}_k(X)$. Hence, for every enforcer $\{w, u\} \in \mathcal{E}$ we have that $\{w, u\} \not\subseteq \mathrm{Sub}_k(X)$ and thus $\mathrm{Sub}_k(X)$ satisfies every enforcer trivially. Therefore, $\mathrm{Sub}_k(X) \in \mathcal{L}(\mathcal{E})$. Conversely, assume that X is not a θ-k-code. Then, there exists $y \in \mathrm{Sub}_k(X) \cap \mathrm{Sub}_k(\theta(X))$, which implies that there exists an enforcer $(\{y\}, \{z\}) \in \mathcal{E}$ with $\{y\} \subseteq \mathrm{Sub}_k(X)$ and $z \notin \mathrm{Sub}_k(X)$. Since $\mathrm{Sub}_k(X)$ does not satisfy $(\{y\}, \{z\})$, we have that $\mathrm{Sub}_k(X) \notin \mathcal{L}(\mathcal{E})$. $\square$

Example 30. Consider $X = \{aab, baab\}$ over the alphabet $\Sigma = \{a, b\}$ and let θ be an antimorphic involution that maps $a \to b$ and $b \to a$. Then $\theta(X) = \{abb, abba\}$ and it is clear that $\mathrm{Sub}_3(X) \cap \mathrm{Sub}_3(\theta(X)) = \emptyset$. Hence, X is a θ-3-code by Definition 1. Since $\mathrm{Sub}_3(X) = \{aab, baa\}$, we construct $\mathcal{E}_{aab} = \{(\{aab, abb\}, \{z\}),$ $(\{aab, bba\}, \{z\})\}$ and $\mathcal{E}_{baa} = \{(\{baa, abb\}, \{z\}),\ (\{baa, bba\}, \{z\})\}$ where $z \notin \mathrm{Sub}_3(X)$. One can verify that for $\mathcal{E} = \mathcal{E}_{aab} \cup \mathcal{E}_{baa}$, $\mathrm{Sub}_3(X) \in \mathcal{L}(\mathcal{E})$.

4 Generating Good Codes by fe-Systems

In the previous section, we characterized some DNA codes through forbidding sets only and/or through enforcing sets only. In this section, we show how an fe-system with a nonempty forbidding set and a nonempty enforcing set can generate θ-k-codes. In contrast to the very special kind of enforcers used in the previous section, where the strictly enforced word z was chosen not to belong to the given set, we use a more general type of enforcers to emphasize the computational, rather than definitional capabilities of fe-systems.

In [14], the authors introduced some methods to generate DNA code words X such that X^+ has the same property. These theoretically generated codes were also tested experimentally in [14] and were shown to have no visible cross-hybridizations. In this section, we construct fe-systems that generalize the methods in [14]. As an illustration, the sets of DNA code words X, X^+ generated in Proposition 4.9 in [14] are in the fe-family. Similar fe-systems constructions can be given for various other methods described in [14].

Proposition 31. *Let Σ be an alphabet and θ a morphic or an antimorphic involution such that $\theta(a) \neq a$ for each symbol $a \in \Sigma$. Let $b, c \in \Sigma$ such that $\theta(b) = c$. Consider the forbidding set $\mathcal{F} = \{\{c\}\} \cup \{\{u\} \mid u \in (\Sigma \setminus \{b, c\})^k\}$. Then $L \in \mathcal{L}(\mathcal{F})$ implies L is a θ-k code.*

Proof. Assume L is not a θ-k-code. Then there exists $u \in \Sigma^k$ such that $u \in \mathrm{Sub}_k(L) \cap \mathrm{Sub}_k(\theta(L))$. If u contains the symbol c, then $L\, ncon\, \mathcal{F}$, so we may assume that $u \in (\Sigma \setminus \{c\})^k$. Since $u \in \mathrm{Sub}\,(\theta(L))$ there exists $v \in \mathrm{Sub}\,(L)$ such that $u = \theta(v)$. This implies that $\theta(u) = \theta(\theta(v) = v \in \mathrm{Sub}\,(L)$. So, both u and $\theta(u)$ are in the subwords of L. If u contains the symbol b, then $\theta(u)$ contains c, which implies that $L\, ncon\, \mathcal{F}$. Otherwise, $u \in (\Sigma \setminus \{b, c\})^k$ and thus $\{u\} \in \mathcal{F}$, which also implies $L\, ncon\, \mathcal{F}$. In all cases, $L \notin \mathcal{L}(\mathcal{F})$. □

Note that the above condition is sufficient for θ-k-codes but not necessary, i.e. there exist θ-k-codes that are not in $\mathcal{L}(\mathcal{F})$, as shown in the next example.

Example 32. Let b, c, Σ and θ be as in Proposition 31. Let $Y = (\Sigma \setminus \{b, c\})^k$ and let $Y = P \cup Q \cup R$ where for all $x \in R$, $\theta(x) = x$, and for all $x \in P$, $\theta(x) \in Q$ and vice versa with P, Q, and R pairwise disjoint. Then for $Z = P \cup \{c^+\}$, $Z \notin \mathcal{L}(\mathcal{F})$ since both $c, u \in \mathrm{Sub}\,(Z)$ for $u \in P \subseteq (\Sigma \setminus \{b, c\}^k)$. However, one can verify that Z is indeed a θ-k-code.

Given $\Delta = \{A, C, T, G\}$, let θ be a morphic or an antimorphic involution. In particular, θ can be the Watson-Crick complementarity such that $\theta(A) = T$, $\theta(T) = A$, $\theta(C) = G$, and $\theta(G) = C$, where θ is antimorphic. We obtain the following consequence of Proposition 31.

Corollary 33. *Let the forbidding set $\mathcal{F} = \{\{G\}\} \cup \{\{u\} \mid u \in (\Delta \setminus \{G, C\})^k\}$ be given. Then $L \in \mathcal{L}(\mathcal{F})$ implies L is a θ-k code.*

Corollary 33 generalizes the experiment performed in [14] in that the forbidder $\{G\}$ models the fact that only three types of nucleotides were used in the sequence design and the rest of the forbidders ensure that every k consecutive nucleotides contain a C. It imposes less restrictions on the set of strands compared to the stricter requirements proposed in Proposition 4.9 in [14]. Note that X, X^+ from Proposition 4.9 in [14] are in the above $\mathcal{L}(\mathcal{F})$, but there is $L \in \mathcal{L}(\mathcal{F})$, such that $L^+ \notin \mathcal{L}(\mathcal{F})$. Hence, the following stronger condition.

Proposition 34. *Let Σ be an alphabet and θ a morphic or an antimorphic involution such that $\theta(a) \neq a$ for each symbol $a \in \Sigma$. Let $b, c \in \Sigma$ such that $\theta(b) = c$ and $k \geq 2$. Consider the forbidding set $\mathcal{F} = \{\{c\}\} \cup \{\{u\} \mid u \in (\Sigma \setminus \{b, c\})^k\}$ and the enforcing set $\mathcal{E} = (\emptyset, \{ub \mid u \in (\Sigma \setminus \{c\})^{k-1}\}) \cup \{(\{u, v\}, \{uv\}) \mid u, v \in \Sigma^*\}$. If $L \in \mathcal{L}(\mathcal{F}, \mathcal{E})$ then L is a θ-k-code. Furthermore, if $L \in \mathcal{L}(\mathcal{F}, \mathcal{E})$ then $L = L^+$.*

Proof. If L is not a θ-k-code, then by Proposition 31, $L \notin \mathcal{L}(\mathcal{F})$ and hence $L \notin \mathcal{L}(\mathcal{F}, \mathcal{E})$. Assume that $L \in \mathcal{L}(\mathcal{F}, \mathcal{E})$. Then $L \, sat \, \mathcal{E}$ and hence if any two words u and v are in L, their concatenation uv is also in L, which is the definition of L^+. Consequently, $L = L^+$. □

Corollary 35. *Let the alphabet be Δ, $\mathcal{F} = \{\{G\}\} \cup \{\{u\} \mid u \in \{A, T\}^k\}$, and $\mathcal{E} = (\emptyset, \{uC \mid u \in \{A, T, C\}^{k-1}\}) \cup \{(\{u, v\}, \{uv\}) \mid u, v \in \Delta^*\}$. Then, $L \in \mathcal{L}(\mathcal{F}, \mathcal{E})$ implies that L is a θ-k-code. Furthermore, if $L \in \mathcal{L}(\mathcal{F}, \mathcal{E})$ then $L = L^+$.*

Fe-systems can be used as a definitional tool or viewed as a one-step computation, but they can also be used as a computational tool which models the evolution of a molecular system. The Γ-tree presented in [5] is based on the idea that one can start with smaller sets of strands and "build" larger sets by applying enforcers in such a way that the resulting sets comply with the forbidden conditions.

In [14], 10 θ-5-codes of length 20 were selected from X (Prop. 4.9) and were tested experimentally. It was shown that no duplexes were detected when all 10 θ-5-codes were annealed and also no cross-hybridizations were observed.

Corollary 35 generalizes this experiment and can be used for a Γ-tree computation as follows. Fix an integer $k \geq 1$. Let $S = Y$, where $(\emptyset, Y)$ is the first enforcer of $\mathcal{E}$. Input S. Observe that $S \, con \, \mathcal{F}$. Apply all applicable enforcers from $\mathcal{E}$ (denote them by $\mathcal{E}_1$), i.e. the enforcers of the kind (X, Y) with $X \subseteq S$. Observe that $S \, sat \, (\emptyset, S)$. The rest of the applicable enforcers ensure that any two strands from S can now anneal. The resulting set of strands S_1 contains all the strands from S and from S^2 and $S_1 \in \mathcal{L}(\mathcal{F}, \mathcal{E}_1)$. In the next step, apply all applicable enforcers to S_1 (denote them by $\mathcal{E}_2$). Apply these enforcers in one step again, i.e. allow all possible annealing of strands from S and S^2 to occur. The resulting set S_2 contains all the strands from S, S^2, S^3, and S^4, i.e. $S_2 = S \cup S^2 \cup S^3 \cup S^4$ and $S_2 \in \mathcal{L}(\mathcal{F}, \mathcal{E}_2)$. The process can be applied as many steps as desired, e.g. until sequences of a desired length have been generated. After the i^{th} step, the resulting set S_i is in the fe-family $\mathcal{L}(\mathcal{F}, \mathcal{E}_i)$ and S_i is a θ-k-code. The process can terminate at any step n by discarding all enforcers that are not applicable up to step n to obtain an enforcing set $\mathcal{E}_n$. We observe that $S_n \in \mathcal{L}(\mathcal{F}, \mathcal{E}_n)$, S_n is a θ-k-code and the fe-system $(\mathcal{F}, \mathcal{E}_n)$ is finite for any $n \geq 1$.

Note that in the experiment performed in [14] the input S consisted of 10 θ-5-codes of length 20. The above algorithm can be modified by replacing the first enforcer by $(\emptyset, S)$ and continuing until a desired set of strands has been generated, i.e. a predetermined length of strands has been reached.

5 Concluding Remarks

Since forbidding-enforcing systems impose restrictions on the subwords and words of a language, they can be used to model the restrictions imposed by unwanted hybridizations and thus, provide a natural framework to study DNA codes. This paper investigated ways to generate DNA codes using fe-systems that define classes of languages (classes of DNA codes). We showed that one fe-system can define an entire class of codes, for example θ-k-codes for a given k, as opposed to just one language (code) generated by a grammar or accepted by an automaton. We see this work as the beginning of research connecting the two areas. Using fe-systems, we confirmed some known properties of DNA codes, which shows a potential for discovering new properties of these codes through fe-systems. DNA codes can also be studied through other variants of fe-systems, such as the single-language fe-system model as defined in [9]. Using the connection between a family of languages defined by a forbidding set as in [5] and a set of words defined by the same forbidding set as described in [9, 10] and the results in this paper, one can generate all words (strands) complying with a certain type of codes, any subset of which will be a desired code. Computation using fe-systems was described through evolving along the Γ-tree introduced in [5]. Defining Γ-tree computations that generate specific DNA codes can have applications in laboratory experiments.

On the other hand, applying fe-systems to DNA codes may enrich the theory of forbidding and enforcing by suggesting new directions in investigating proposed fe-systems, as well as, a need for defining and studying new fe-system models. Different variants of fe-systems defining languages, words, or graphs were shown to be capable of defining solutions to NP-complete problems [5, 9, 12]. Since laboratory experiments in DNA computing can be modeled by languages and graphs, including DNA codes in the set of structures defined by fe-systems adds to the development of the theory of fe-systems as a natural framework to study molecular computation.

Acknowledgement. The authors thank Nataša Jonoska for numerous discussions and helpful suggestions to link our results with experimental work. This work has been partially supported by a UNF Faculty Development Scholarship Grant for Daniela Genova.

References

1. Adleman, L.: Molecular computation of solutions to combinatorial problems. Science 266, 1021–1024 (1994)
2. Cavaliere, M., Jonoska, N.: Forbidding and enforcing in membrane computing. Natural Computing 2, 215–228 (2003)
3. Deaton, R., Murphy, R., Rose, J., Garzon, M., Franceschetti, D., Jr., S.S.: A DNA based implementation of an evolutionary search for good encodings for DNA computation. In: Proc. IEEE Conference on Evolutionary Computation ICEC-97, pp. 267–271 (1997)
4. Ehrenfeucht, A., Hoogeboom, H.J., Rozenberg, G., van Vugt, N.: Forbidding and enforcing. In: DNA Based Computers V, pp. 195–206. AMS DIMACS, Providence, RI (2000), volume 54
5. Ehrenfeucht, A., Rozenberg, G.: Forbidding-enforcing systems. Theoretical Computer Science 292, 611–638 (2003)
6. Faulhammer, D., Cukras, A.R., Lipton, R.J., Landweber, L.F.: Molecular computation: RNA solutions to chess problems. Proceedings of the National Academy of Sciences, USA 97(4), 1385–1389 (2000)
7. Franco, G., Jonoska, N.: Forbidding and enforcing conditions in DNA self-assembly of graphs. Nanotechnology: Science and Computation, Natural Computing Series Part I pp. 105–118 (2006)
8. Garzon, M.: On codeword design in metric spaces. Natural Computing 8(3), 571–588 (2009)
9. Genova, D.: Defining languages by forbidding-enforcing systems. Models of Computation in Context, Computability in Europe CiE 2011 (B. Löwe, et al eds), Lecture Notes in Computer Science 6735, 92–101 (2011)
10. Genova, D.: Forbidding sets and normal forms of language forbidding-enforcing systems. Proc. of 6th Int. Conf. on Language and Automata Theory and Applications LATA 2012 (A. H. Dediu, C. Martin-Vide eds.), Lecture Notes in Computer Science 7183, 289–300 (2012)
11. Genova, D., Jonoska, N.: Topological properties of forbidding-enforcing systems. Journal of Automata, Languages and Combinatorics 11(4), 375–397 (2006)
12. Genova, D., Jonoska, N.: Forbidding and enforcing on graphs. Theoretical Computer Science 429, 108–117 (2012)
13. Hussini, S., Kari, L., Konstantinidis, S.: Coding properties of DNA languages. Theoretical Computer Science 290, 1557–1579 (2003)
14. Jonoska, N., Mahalingam, K., Chen, J.: Involution codes: with application to DNA coded languages. Natural Computing 4, 141–162 (2005)
15. Kari, L., Konstantinidis, S., Losseva, E., Wozniak, G.: Sticky-free and overhang-free DNA languages. Acta Informatica 40, 119–157 (2003)
16. Kari, L., Mahalingam, K.: DNA codes and their properties. Proceedings of DNA Computing 12 (C.Mao, T.Yokomori eds.), Lecture Notes in Computer Science 4287, 127–142 (2006)
17. Liu, Q., Wang, L., Frutos, A.G., Condon, A., Corn, R.M., Smith, L.M.: DNA computing on surfaces. Nature 403, 175–179 (2000)
18. Marathe, A., Condon, A.E., Corn, R.M.: On combinatorial word design. In: Preliminary Preproceedings of the 5th International Meeting on DNA Based Computers, pp. 75–88. Boston (1999)