Bond-free Languages: Formalizations, Maximality and Construction Methods

Lila Kari[1], Stavros Konstantinidis[2], Petr Sosík[1,3]

[1]Department of Computer Science, The University of Western Ontario,
London, ON, Canada, N6A 5B7, {lila,sosik}@csd.uwo.ca

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

[3]Institute of Computer Science, Silesian University,
Opava, Czech Republic

Abstract

The problem of negative design of DNA languages is addressed, that is, properties and construction methods of large sets of words that prevent undesired bonds when used in DNA computations. We recall a few existing formalizations of the problem and then define the property of sim-bond-freedom, where sim is a similarity relation between words. We show that this property is decidable for context-free languages and polynomial-time decidable for regular languages. The maximality of this property also turns out to be decidable for regular languages and polynomial-time decidable for an important case of the Hamming similarity. Then we consider various construction methods for Hamming bond-free languages, including the recently introduced method of templates, and obtain a complete structural characterization of all maximal Hamming bond-free languages. This result is applicable to the θ-k-code property introduced by Jonoska and Mahalingam.

1 Introduction

The field of DNA computing is based on the fact that one can encode input data into a collection of *(single-stranded) DNA molecules* and then apply on them a sequence of *operations*, which results in a modified collection of molecules. This process can be interpreted as a computation for which the output is obtained by decoding the data contained in some of the resulting molecules. In practice, the collection of DNA molecules exists as a 'soup' inside a test *tube* under controlled physical conditions.

1.1 Bonds between DNA molecules

Most of the operations involved in DNA computations rely on the capability of controlling the *bonds* that can be formed between DNA molecules. Such bonds are created due to the well-known Watson-Crick *complementarity* property of the four nucleotides A, C, G, T, which are the building blocks of DNA molecules. More specifically, the nucleotide A is complementary to T, and C is complementary to G. This property is important in conjunction with the fact that every molecule

```
5'- A G T T C C -3'        5'- v A G T T C C w -3'        5'- v A G T T C C w
    | | | | | |                  | | | | | |                      | | | | | |  x
3'- T C A A G G -5'        3'- z T C A A G G y -5'        3'- z T C A A G G y

        (a)                          (b)                           (c)
```

Figure 1: Vertical bars represent bonds between complementary nucleotides. In (b), the complementary parts $5'-AGTTCC-3'$ and $5'-GGAACT-3'$ of the DNA molecules $5'-vAGTTCCw-3'$ and $5'-yGGAACTz-3'$ bind together. In (c), the molecule $5'-vAGTTCCwxyGGAACTz-3'$ is twisted at x and its complementary parts bind together.

```
5'- v A G A T T C C G T G w -3'        5'- v A G A T T C C G T G w
      | | |   | | |   | |                    | | |   | | |   | |  x
3'- z T C T C A G G A A C y -5'        3'- z T C T C A G G A A C y

              (a)                                    (b)
```

Figure 2: In (a), parts of two DNA molecules bind together although these parts are not perfect complements of each other. In (b), the same parts appear in one molecule.

has a certain *orientation*, which is denoted by placing the symbols '$5'-$' and '$-3'$' at the two ends of the sequence of nucleotides comprising the molecule. For example, the molecules $5'-ACCGT-3'$ and $3'-ACCGT-5'$ are different – they have different chemical properties.

Under favorable physical tube conditions, if a molecule of the form $5'-X_1X_2\cdots X_k-3'$, where each X_i is a nucleotide, encounters the molecule $5'-Y_k\cdots Y_2Y_1-3'$ in which each nucleotide Y_i is the complement of X_i, then the pairs (X_i, Y_i) will form k chemical bonds and a double-stranded structure will be created – see Figure 1(a).
It is important to note that bonds can be formed even between complementary *parts* of two molecules, provided that these parts are *sufficiently long* – see Figure 1(b). Moreover, a molecule containing two complementary parts can bind to itself, or to a copy of itself – see Figure 1(c).

The bonds shown in Figure 1 are formed between parts that are *perfect* complements of each other. In practice, however, it is possible that two parts of molecules will bind together even if some of their corresponding nucleotides are not complementary to each other – see Figure 2.

1.2 The problem of undesirable bonds

The success of a DNA operation relies on the assumption that no accidental bonds can be formed between molecules in the tube before the operation is initiated, or even during the operation. With this motivation, one of the foremost problems in DNA computing today is the following.

Problem 1 Define a large, potential collection of DNA molecules such that there can be no (sufficiently long and possibly imperfect) complementary parts in any two molecules, and no (sufficiently long and possibly imperfect) complementary parts in any one molecule.

In many cases in the literature, this problem is addressed in conjunction with the *uniqueness* problem, which involves designing molecules whose parts are different between each other. The motivation here is that, usually, a DNA operation is intended only for molecules containing a specific pattern (or specific patterns) of nucleotides. In this paper, however, we focus on Problem 1.

1.3 Notation for molecules and bonds

We proceed now with establishing the notation that would allow us to describe formalizations of Problem 1. Specifically, we define the terms *word, subword, language, involution, and codeword.*

A given alphabet can be used to form sequences of symbols that are called *words.* For example, 01001 is a word over the alphabet $\{0,1\}$. The *length* of a word w is denoted by $|w|$. For example, $|01001| = 5$. The prime example of an alphabet will be the DNA alphabet $\{A,C,G,T\}$. In this case, we agree that the left end of a *DNA word* represents the $5'-$end of the corresponding DNA molecule. For example, the word $CCATGT$ represents the molecule $5' - CCATGT - 3'$. If a word w can be written in the form xyz – this is the catenation of some words x, y and z – then we say that y is a *subword* of w. A *language* is any set of words. We shall use the expression 'x is a subword of a language' as a shorthand for x is a subword of some word in the language. One use of a language L is to represent all the possible distinct copies of DNA molecules that might appear in a tube. In this case, we refer to L as a *tube language* and we assume that every word in L is of length at least k, for some *parameter* k. This parameter represents the smallest length of two molecule parts for which it is possible to form a stable bond.

To represent the complementarity of nucleotides we use the concept of antimorphic involution introduced in [17]. In general an *involution* of an alphabet Σ is a function $\theta : \Sigma \to \Sigma$ such that $\theta(\theta(a)) = a$, for all symbols a in Σ. The involution is called *antimorphic* if we extend it to words such that $\theta(a_1 \cdots a_n) = \theta(a_n) \cdots \theta(a_1)$, where each a_i is a symbol in Σ. The prime example of an antimorphic involution will be the *DNA involution* τ such that

$$\tau(A) = T,\ \tau(T) = A,\ \tau(C) = G,\ \tau(G) = C.$$

For example, $\tau(ACCGTT) = AACGGT$. In general, for two DNA words x and y of length k, *the identity* $\tau(x) = y$ *represents the fact that the molecules (or parts of molecules)* $5' - x - 3'$ *and* $5' - y - 3'$ *could bind to each other.* According to the requirement in Problem 1, if $k = 6$, the words $ACCGTT$ and $AACGGT$ should not be subwords of the tube language L.

In the literature on DNA encodings, the tube language L is usually equal to, or a subset of, K^+, where K is a finite language whose elements are called *codewords.* The language K^+ consists of all words that are obtained by concatenating one or more codewords from K. For a nonnegative integer n, the notation K^n is used for the set of all words that are obtained by concatenating any n codewords from K. In general, K might contain codewords of different lengths. In many cases, however, the set K consists of words of a certain fixed length l. In this case, we shall refer to K as a *code of length* l.

1.4 Formalizations of the problem of undesirable bonds

With the preceding terminology in mind, Problem 1 is called the negative word design problem in [25]. Now we recall a few existing formalizations of Problem 1 and we propose a new one, which appears to be closer to the intuition behind the problem. It should be noted, however, that all formalizations are inter-related in some interesting ways.

One of the most recent attempts to address Problem 1 appears in [12]. In that paper, the authors require that a tube language L must satisfy the following property.

P1[k]**:** If x and y are any subwords of L of length k then $x \neq \tau(y)$.

A language satisfying this property is called a τ-k-code in [12]. An advantage of this formalization is that the property is defined *independently* of the structure of L. This property is also considered implicitly in [3] and [6]. In particular, reference [3] considers tube languages of the form $(sZ)^+$ satisfying $\mathbf{P1}[k]$, where s is a fixed word of length k and Z is a code of length k – the notation sZ represents the set of all words sz such that z is in Z.

In [11], the authors introduce the concept of a *strictly τ-free code* K, which is a generalization of the notion of comma-free code, and show that the language K^+ must be strictly τ-free as well. Here we shall assume that K is of fixed length k. In this formalization the tube language L is equal to K^+. Using the tools of [11], it can be shown that L is a strictly τ-free language *iff* (if and only if) L satisfies the following property

P2$[k]$**:** If x is a subword of L of length k and v is a codeword in K then $x \neq \tau(v)$.

We note that similar properties are considered also in [19] and [20].

As noted earlier, parts of DNA molecules can bind to each other even if they are not perfect complements of each other. Hence, although sufficient, the condition $\tau(x) = y$ might not be necessary for the DNA words x and y to stick together. The common approach to deal with this is to modify the above condition by using the Hamming distance function $H(\cdot, \cdot)$. More specifically, for two words x and y of length k, *the relation $H(x, \tau(y)) \leq d$ represents the fact that the molecules (or parts of molecules) $5' - x - 3'$ and $5' - y - 3'$ could bind to each other.* Here, d is a nonnegative integer less than k.

In [23] and [31], the authors consider codes K of length k satisfying the following property

P3$[d, k]$**:** If u and v are any codewords in K then $H(u, \tau(v)) > d$.

In fact the above property is studied in conjunction with the uniqueness property $H(K) > d$.

Reference [7] introduces the H-measure for two words x and y of length k and explains how this measure can be used to encode instances of the Hamiltonian Path problem. A similar measure is defined in [2] and is applied to codes of length k whose words can be concatenated in arbitrary ways. Thus, the tube language here is $L = K^+$. The code K satisfies certain uniqueness conditions as well as conditions related to Problem 1. In particular, the tube language $L = K^+$ satisfies the following property.

P4$[d, k]$**:** If x is a subword of L of length k and v is a codeword in K then $H(x, \tau(v)) > d$.

We note that also reference [27] considers this property for languages of the form $K_1 K_2 \cdots K_m$, where each K_i is a certain code of length k.

With '$H(x, \tau(y)) \leq d$' as the criterion for x and y to bind together, it appears that $\mathbf{P4}[d, k]$ is the strictest property in the literature for addressing Problem 1. This property, however, is not sufficient in general for avoiding undesirable bonds in the tube. To see this, consider the case where

$$d = 1, \quad k = 5, \quad K = \{ACGAT, CCGAA\}.$$

One can verify that K^+ satisfies $\mathbf{P4}[d, k]$ and that the DNA words

$$ACGATACGATCCGAA \quad ACGATCCGAACCGAA$$

are in K^+ and contain the subwords $GATCC$ and $CGATC$ such that

$$H(GATCC, \tau(CGATC)) \leq 1.$$

Motivated by the above observation, we introduce the following property of a tube language L.

P5$[d, k]$**:** If x and y are any subwords of L of length k then $H(x, \tau(y)) > d$.

Note that, as in the case of $\mathbf{P1}[k]$, the new property is defined independently of the structure of L. Any tube language satisfying this property will be called a *$(\tau, H_{d,k})$-bond-free language.*

```
5'- v A A G C G T T C G A w -3'
      \ \  | | |   | | |
   3'-z T T G C A G G C T y -5'
```

Figure 3: Two DNA molecules in which the parts $5' - AAGCGTTCGA - 3'$ and $5' - TCGGACGTT - 3'$ bind together although these parts have different lengths.

1.5 Connections

We list now a few interesting connections among the properties **P1–P5**. We note that the condition $x \neq \tau(y)$ is equivalent to $H(x, \tau(y)) > 0$.

P3 and P5: In Section 3 we introduce the *subword closure operation* $\otimes$ such that, for any code S of length k, the language $S^{\otimes}$ consists of all words w of length at least k with the property that every subword of length k of w is in S. Moreover we show how to construct $S^{\otimes}$ from S in linear time. In Section 4.3, we show that if a code K satisfies $\mathbf{P3}[d, k]$ then the language $K^{\otimes}$ satisfies $\mathbf{P5}[d, k]$. Hence, any constructions of codes K for **P3** are relevant to **P5**.

P4 and P2: It is evident that any language K^+ satisfying $\mathbf{P4}[d, k]$ also satisfies $\mathbf{P2}[k]$. Moreover, $\mathbf{P4}[0, k]$ is identical to $\mathbf{P2}[k]$. In Section 4.2, we show that, for every code Q of length q, if the language Q^+ satisfies $\mathbf{P2}[q]$ then the language $(Q^{d+1})^+$ satisfies $\mathbf{P4}[d, q(d+1)]$, for any $d > 0$. Hence, any constructions of strictly τ-free codes K are relevant to **P4**.

P4 and P5: It is evident that any language K^+ satisfying $\mathbf{P5}[d, k]$also satisfies $\mathbf{P4}[d, k]$. Moreover, it can be shown that if K^+ satisfies $\mathbf{P4}[d, k]$ then $(K^2)^+$ satisfies $\mathbf{P5}[d, k]$. Hence, any constructions of codes K for **P4** are relevant to **P5**.

P5 and P1: Obviously, any language satisfying $\mathbf{P5}[d, k]$ also satisfies $\mathbf{P1}[k]$. Moreover, the property $\mathbf{P1}[k]$ coincides with $\mathbf{P5}[0, k]$. In Section 4.2, we show that every language satisfying $\mathbf{P1}[q]$, for some positive integer q, also satisfies $\mathbf{P5}[d, q(d+1)]$ for every $d > 0$, and conversely, if the language is of the form K^+ and satisfies $\mathbf{P5}[d, k]$ then it satisfies $\mathbf{P1}[k - d]$ as well. Hence, any constructions of languages satisfying **P1** are relevant to **P5**.

1.6 A more general formalization: (θ, sim)-bond-freeness

The choice of the Hamming distance in the condition '$H(x, \tau(y))$' for *similarity* between words is a very natural one and has attracted a lot of interest in the literature. One might argue, however, that parts of two DNA molecules could form a stable bond even if they have different lengths. In Figure 3, for example, the bound parts of the two molecules have lengths 10 and 9. Such hybridizations (and even more complex ones) are addressed in [1]. Based on this observation, the condition for two subwords x and y to bind together should be

$$|x|, |y| \geq k \quad \text{and} \quad Lev(x, \tau(y)) \leq d.$$

The symbol $|u|$ denotes the length of the word u and $Lev(u, v)$ is the *Levenshtein distance* between the words u and v – this is the smallest number of substitutions, insertions and deletions of symbols required to transform u to v. With this formulation, the condition for similarity based on the Hamming distance can be rephrased as follows

$$|x|, |y| \geq k \quad \text{and} \quad H(x, \tau(y)) \leq d,$$

where we assume that $H(u, v) = \infty$ if the words u and v have different lengths. In general, for any similarity relation $\mathrm{sim}(\cdot, \cdot)$ between words and for every involution θ, we define the following property of a language L.

$\mathbf{P}[\theta, \mathrm{sim}]$: If x and y are any nonempty subwords of L then $\mathrm{sim}(x, \theta(y))$ is false.

Any language satisfying $\mathbf{P}[\theta, \mathrm{sim}]$ is called a (θ, sim)*-bond-free language*.

The precise definition of a similarity relation is given in Section 2. There it is shown that the relations '$|u|, |v| \geq k$ and $H(u, v) \leq d$' and '$|u|, |v| \geq k$ and $Lev(u, v) \leq d$' are indeed similarity relations. For these relations we shall use the notation

$$H_{d,k} \quad \text{and} \quad Lev_{d,k},$$

respectively.

1.7 Structure of the paper

In this paper we are interested in maximal bond-free languages. Let M be a fixed language and let $\mathbf{P}$ be a property of languages. We say that a language L is a *maximal* $\boldsymbol{P}$ *subset of* M if there is no word w in $M - L$ such that $L \cup \{w\}$ satisfies $\mathbf{P}$. We note that maximality is a central theme in the theory of variable-length codes [14].

In Section 2 we recall the general tools developed in [20] for proving decidability questions for various language properties, and we show that one can decide in quadratic time whether a given regular language is (θ, sim)-bond-free. We also address this problem for the special cases where sim is equal to $H_{d,k}$ or $Lev_{d,k}$. Moreover, we show that this problem is decidable even when the given language is context-free. Then we recall the general tools developed in [18] and [20] for deciding whether a given language is maximal with respect to a certain property, and we establish the decidability of whether a given regular language is maximal with respect to the (θ, sim)-bond-free property.

The decision method for maximality presented in Section 2 is not of polynomial time. In Section 3, however, we are able to show a polynomial time algorithm for testing whether a given regular langauge is $(\theta, H_{d,k})$-bond-free, for $d = 0$ or $d = 1$. In that section we also introduce the subword closure operation, $\otimes$, which plays an important role in the paper.

In Section 4, we consider the problem of constructing tube languages satisfying the Hamming bond-free property. Firstly, we describe a few direct methods, that is methods that do not rely on other constructions, and then we show how to use languages satisfying $\mathbf{P1}[k]$ to construct new languages satisfying $\mathbf{P5}[d, k]$, that is $(\tau, H_{d,k})$-bond-free languages. Moreover, we obtain a complete structural characterization of *all* maximal $(\tau, H_{d,k})$-bond-free tube languages. In the case of $d = 0$, the characterization is quite explicit and allows us to give the exact number of these languages.

In the last section, we conclude the paper with a summary of our results and a few directions for future research.

2 Decidability Questions about (θ, sim)-bond-freedom

In this section, we recall the general tools developed in [20] for proving decidability questions for various language properties, and we show that one can decide in quadratic time whether a given regular language is (θ, sim)-bond-free. We also address this problem for the special cases where sim is equal to $H_{d,k}$ or $Lev_{d,k}$. Moreover, we show that this problem is decidable even when the given language is context-free. Then, we recall the general tools developed in [18] and [20] for deciding whether a given language is maximal with respect to a certain property, and we establish the decidability of whether a given regular language is maximal with respect to the (θ, sim)-bond-free property.

2.1 Notation and basic tools

Here we introduce some notation about *binary relations, automata and transducers, and binary word operations*. We assume the reader is familiar with the notation of Subsection 1.3.

We shall use a fixed non-singleton alphabet Σ and a fixed involution $\theta : \Sigma \to \Sigma$. The set of all words (over Σ) is denoted by Σ^* and includes the *empty word* λ. The involution θ can be extended to Σ^* as a morphic or antimorphic involution.

Recall that a language L is a set of words, or equivalently a subset of Σ^*. The notation L^c represents the complement of the language L; that is, $L^c = \Sigma^* - L$.

A *binary relation* γ, say, over Σ is a subset of $\Sigma^* \times \Sigma^*$. The expression '(u, v) is in γ' can be rephrased as '$\gamma(u, v)$ is true' when we view γ as a logic predicate. We are interested in binary relations intended to define when two words are similar.

Definition 2.1 *A binary relation* sim *is called a* similarity relation with parameters (t, l), *where* t *and* l *are nonnegative integers, if the following conditions are satisfied.*

(i) If $\mathrm{sim}(u, v)$ *is true then* $\mathrm{abs}(|u| - |v|) \le t$.

(ii) If $\mathrm{sim}(u, v)$ *is true and* $|u|, |v| > l$ *then there are proper subwords* x *and* y *of* u *and* v*, respectively, such that* $\mathrm{sim}(x, y)$ *is true.*

We can interpret the above conditions as follows: (i) the lengths of two similar words cannot be too different and (ii) if two words are similar and long enough, then they contain two similar proper subwords. *In the rest of the section we shall assume that* sim *is a fixed, but arbitrary, similarity relation with parameters* (t, l). It is evident that the relation $H_{d,k}$ defined in Subsection 1.6 is an example of a similarity relation with parameters $(0, k)$. In the next subsection we show that $Lev_{d,k}$ is a similarity relation as well, with parameters $(d, d + k)$.

Automata, Tries and Transducers

A nondeterministic finite automaton with λ productions (or transitions), a *λ-NFA* for short, is a quintuple $A = (S, \Sigma, s_0, F, P)$ such that S is the finite and nonempty set of states, s_0 is the start state, F is the set of final states, and P is the set of productions of the form $sx \to t$, where s and t are states in S, and x is either a symbol in Σ or the empty word. The automaton is called *trim* if every state is reachable from the start state and can reach a final state. If there is no production with $x = \lambda$, the automaton is called an *NFA*. If for every two productions of the form $sx_1 \to t_1$ and $sx_2 \to t_2$ of an NFA we have that $x_1 \ne x_2$ then the automaton is called a *DFA (deterministic finite automaton)*. The language accepted by the automaton A is denoted by $L(A)$. Any language accepted by some λ-NFA is called regular. The *size* $|A|$ of the automaton A is the number $|S| + |P|$.

A *trie* is a DFA with the following structure

$$(\{[p] \mid p \in \mathrm{Pref}(S)\},\ \Sigma,\ [\lambda],\ \{[s] \mid s \in S\},\ P),$$

where S is a finite language, $\mathrm{Pref}(S)$ is the set of all prefixes of S, and the set of productions is equal to

$$P = \{[p]a \to [pa] \mid p \in \mathrm{Pref}(S),\ a \in \Sigma,\ pa \in \mathrm{Pref}(S)\}.$$

Note that each state $[p]$ represents the prefix p of the input word that has been read so far by the automaton. This implies that the trie accepts the language S.

A *finite transducer (in standard form)* is a sextuple $T = (S, \Sigma, \Sigma', s_0, F, P)$ such that Σ' is the output alphabet, the components S, s_0, F are as in the case of λ-NFAs, and the set P consists of productions of the form $sx \to yt$ where s and t are states in S, $x \in \Sigma \cup \{\lambda\}$ and $y \in \Sigma' \cup \{\lambda\}$. The *relation realized by* the transducer T is denoted by $R(T)$. A binary relation is called *rational* if it can be realized by a finite transducer. For any word w, the symbol $T(w)$ represents the set of all outputs of T on input w, that is, $T(w) = \{z \mid (w, z) \in R(T)\}$. The *size* $|T|$ of the transducer T (in standard form) is $|S| + |P|$. For any λ-NFA A, one can construct the λ-NFA A_T of size $O(|T||A|)$ that accepts the language $T(L(A)) = \{z \mid (w, z) \in R(T),\ w \in L(A)\}$ [22].

We refer the reader to [29] or [33] for further details on automata and formal languages.

Binary word operations

Binary word operations are extensively used in this section as an important tool for representing interaction of DNA molecules. A *binary (word) operation* is a mapping $\diamondsuit : \Sigma^* \times \Sigma^* \to 2^{\Sigma^*}$, where 2^{Σ^*} is the set of all subsets of Σ^*. Hence the result of the operation $\diamondsuit$ with operands $u, v \in \Sigma^*$ is generally a language $(u \diamondsuit v) \subseteq \Sigma^*$. In some important particular cases we have $|u \diamondsuit v| = 1$ for $u, v \in \Sigma^*$. If there is no risk of misunderstanding, we then may assume $u \diamondsuit v = w$, $w \in \Sigma^*$, instead of the singleton language $\{w\} \subseteq \Sigma^*$. A typical example is the catenation operation $u \cdot v$.

We extend a binary operation $\diamondsuit$ to any languages X and Y as follows:

$$X \diamondsuit Y = \bigcup_{u \in X, v \in Y} u \diamondsuit v. \tag{1}$$

Let $\diamondsuit$ be a binary operation. The *left inverse* $\diamondsuit^l$ of $\diamondsuit$ is defined as [16]

$$w \in (x \diamondsuit v) \text{ iff } x \in (w \diamondsuit^l v), \text{ for all } v, x, w \in \Sigma^*,$$

and the *right inverse* $\diamondsuit^r$ of $\diamondsuit$ is defined as

$$w \in (u \diamondsuit y) \text{ iff } y \in (u \diamondsuit^r w), \text{ for all } u, y, w \in \Sigma^*.$$

The *reversed* $\diamondsuit'$ of $\diamondsuit$ is defined by $u \diamondsuit' v = v \diamondsuit u$.

Several basic binary operations, together with their inverses, can be found in [24, 15, 18]. Here we shall use binary operations involving trajectories [4, 21, 24]. Consider a *trajectory alphabet* $V = \{0, 1\}$ and assume $V \cap \Sigma = \emptyset$. We call *trajectory* any string $t \in V^*$. A trajectory is essentially a syntactical condition which specifies how a binary word operation $\diamondsuit$ is applied to the letters of its two operands. Let $t \in V^*$ be a trajectory and let α, β be two words over Σ. The *shuffle of α with β on the trajectory t,* denoted by $\alpha \sqcup\!\sqcup_t \beta$, is defined as follows:

$$\begin{aligned} \alpha \sqcup\!\sqcup_t \beta \quad = \quad & \{\alpha_1\beta_1 \ldots \alpha_k\beta_k \mid \alpha = \alpha_1 \ldots \alpha_k,\ \beta = \beta_1 \ldots \beta_k,\ t = 0^{i_1}1^{j_1} \ldots 0^{i_k}1^{j_k}, \text{ where} \\ & |\alpha_m| = i_m \text{ and } |\beta_m| = j_m \text{ for all } m,\ 1 \le m \le k\}. \end{aligned}$$

Example 2.2 Let $\alpha = a_1a_2 \ldots a_8$, $\beta = b_1b_2 \ldots b_5$ and assume that $t = 0^31^20^310101$. The shuffle of α and β on the trajectory t is: $\alpha \sqcup\!\sqcup_t \beta = \{a_1a_2a_3b_1b_2a_4a_5a_6b_3a_7b_4a_8b_5\}$.

The *deletion of β from α on the trajectory t* is the following binary word operation:

$$\begin{aligned} \alpha \rightsquigarrow_t \beta \quad = \quad & \{\alpha_1 \ldots \alpha_k \mid \alpha = \alpha_1\beta_1 \ldots \alpha_k\beta_k,\ \beta = \beta_1 \ldots \beta_k,\ t = 0^{i_1}1^{j_1} \ldots 0^{i_k}1^{j_k}, \text{ where} \\ & |\alpha_m| = i_m \text{ and } |\beta_m| = j_m \text{ for all } m,\ 1 \le m \le k\}. \end{aligned}$$

Example 2.3 Let $\alpha = babaab$, $\beta = bb$ and assume that $t = 001001$. The deletion of β from α on the trajectory t is: $\alpha \rightsquigarrow_t \beta = \{baaa\}$.

Notice also that for given α, β, t we have always $|\alpha \sqcup\!\sqcup_t \beta| \leq 1$, $|\alpha \rightsquigarrow_t \beta| \leq 1$.

A *set of trajectories* is any set $T \subseteq V^*$. The *shuffle (deletion) of* α *with* β *on the set* T, denoted by $\alpha \sqcup\!\sqcup_T \beta$ ($\alpha \rightsquigarrow_T \beta$), is:

$$\alpha \diamondsuit_T \beta = \bigcup_{t \in T} \alpha \diamondsuit_t \beta, \tag{2}$$

where $\diamondsuit$ stands for $\sqcup\!\sqcup$ or $\rightsquigarrow$, respectively. The operations $\sqcup\!\sqcup_T$ and $\rightsquigarrow_T$ generalize to languages due to the general principle (1).

The following results are proven in [4, 21, 24] or follow directly by proof techniques used ibidem.

Lemma 2.4 *Let* T *be a set of trajectories, then*

(i) $\sqcup\!\sqcup^l_T \; = \rightsquigarrow_T$ *and* $\sqcup\!\sqcup^r_T \; = \rightsquigarrow'_{\widetilde{T}}$,

(ii) $\rightsquigarrow^l_T \; = \; \sqcup\!\sqcup_T$ *and* $\rightsquigarrow^r_T \; = \rightsquigarrow_{\widetilde{T}}$,

where $\widetilde{T}$ *is the set of trajectories obtained by replacing all* 0*'s for* 1*'s and vice versa in all the trajectories of* T.

Lemma 2.5 *Let* L_1, L_2 *and* T *be regular languages accepted by the NFA's* A_1, A_2 *and* A_T, *respectively.*

(i) There exists a NFA A *accepting* $L_1 \sqcup\!\sqcup_T L_2$ *of the size* $|A| = \mathcal{O}(|A_1| \cdot |A_2| \cdot |A_T|)$, *constructible in time* $|A|$.

(ii) There exists a λ*-NFA* A' *accepting* $L_1 \rightsquigarrow_T L_2$ *of the size* $|A'| = \mathcal{O}(|A_1| \cdot |A_2| \cdot |A_T|)$, *constructible in time* $|A'|$.

2.2 Decidability for regular and context-free languages

Reference [19] defines several classes of languages (language properties) for avoiding undesired bonds. In [20], it is observed that many of these properties can be expressed using a predicate that involves two binary word operations as parameters.

Definition 2.6 [20] *A mapping* $\mathcal{P} : 2^{\Sigma^*} \longrightarrow \{\text{true}, \text{false}\}$ *is called a* strictly bond-free property (of degree 2)*, if there are binary word operations* $\diamondsuit_{\text{lo}}$, $\diamondsuit_{\text{up}}$ *and an involution* θ *such that for an arbitrary* $L \subseteq \Sigma^*$, $\mathcal{P}(L) = \text{true}$ *iff*

$$\forall w, x, y \in \Sigma^* \; (w \diamondsuit_{\text{lo}} x \cap L \neq \emptyset, \; w \diamondsuit_{\text{up}} y \cap \theta(L) \neq \emptyset) \Rightarrow w = \lambda. \tag{3}$$

Theorem 2.7 (θ, sim)*-bond-freedom is a strictly bond-free property.*

Proof. We define the mappings sim_L and sim_R as follows:

$$\begin{aligned} \text{sim}_L(y) &= \{x \in \Sigma^* \mid \text{sim}(x, y)\}, & (4)\\ \text{sim}_R(x) &= \{y \in \Sigma^* \mid \text{sim}(x, y)\}. & (5) \end{aligned}$$

Recall that a language L is (θ, sim)-bond-free iff

$$\begin{array}{ll}
\forall x_1, y_1, x_2, y_2 \in \Sigma^*,\ w_1, w_2 \in \Sigma^+, & \\
\quad x_1 w_1 y_1,\ x_2 w_2 y_2 \in L \ \Rightarrow\ \text{not } \mathrm{sim}(w_1, \theta(w_2)) & \text{iff} \\
\forall x_1, y_1, x_2, y_2 \in \Sigma^*,\ w_1, w_2 \in \Sigma^+, & \\
\quad x_1 w_1 y_1,\ x_2 \theta(w_2) y_2 \in L \ \Rightarrow\ \text{not } \mathrm{sim}(w_1, w_2) & \text{iff} \\
\forall x_1, w_1, y_1, x_2, w_2, y_2 \in \Sigma^*, & \\
\quad x_1 w_1 y_1,\ x_2 \theta(w_2) y_2 \in L, \mathrm{sim}(w_1, w_2) \Rightarrow (w_1 = \lambda \text{ or } w_2 = \lambda) & \text{iff} \\
\forall x_1, w_1, y_1, x_2, w_2, y_2 \in \Sigma^*, & \\
\quad x_1 w_1 y_1 \in L,\ x_2 w_2 y_2 \in \theta(L), w_2 \in \mathrm{sim}_R(w_1) \Rightarrow (w_1 = \lambda \text{ or } w_2 = \lambda) & \text{iff} \\
\forall x_1, y_1, x_2, y_2, w \in \Sigma^*, & \\
\quad (\{x_1 w y_1\} \cap L \neq \emptyset,\ \{x_2\} \cdot (\mathrm{sim}_R(w) \cap \Sigma^+) \cdot \{y_2\} \cap \theta(L) \neq \emptyset) \Rightarrow w = \lambda & \text{iff} \\
\forall x, y, w \in \Sigma^*, & \\
\quad (w \sqcup\!\sqcup_T x \cap L \neq \emptyset,\ (\mathrm{sim}_R(w) \sqcup\!\sqcup_T y) \cap \theta(L) \neq \emptyset) \Rightarrow w = \lambda, &
\end{array}$$

where $T = 1^* 0^+ 1^*$. $\square$

We note that the above theorem, as well as Corollary 2.9 and Theorem 2.10 (see further below) remain valid even if sim is an arbitrary binary relation that does not necessarily satisfy the conditions in Definition 2.1. The following result has been shown in [20]:

Theorem 2.8 *Let $\mathcal{P}$ be a strictly bond-free property associated with operations $\diamondsuit_{\mathrm{lo}}$, $\diamondsuit_{\mathrm{up}}$. For a language $L \subseteq \Sigma^*$, $\mathcal{P}(L) = \mathrm{true}$ iff*

$$(L \diamondsuit^l_{\mathrm{lo}} \Sigma^*) \rightsquigarrow_{1+} (\theta(L) \diamondsuit^l_{\mathrm{up}} \Sigma^*) = \emptyset. \tag{6}$$

Corollary 2.9 *A language $L \subseteq \Sigma^*$ is (θ, sim)-bond-free, iff*

$$(L \rightsquigarrow_T \Sigma^*) \rightsquigarrow_{1+} \mathrm{sim}_L(\theta(L) \rightsquigarrow_T \Sigma^*) = \emptyset,$$

where $T = 1^* 0^+ 1^*$.

Proof. Comparing Definition 2.6 and the proof of Theorem 2.7, we obtain

$$x \diamondsuit_{\mathrm{lo}} y = x \sqcup\!\sqcup_T y, \quad x \diamondsuit_{\mathrm{up}} y = \mathrm{sim}_R(x) \sqcup\!\sqcup_T y.$$

Observe that for a binary operation $\diamondsuit$,

$$\begin{array}{lll}
z \in \mathrm{sim}_R(x) \diamondsuit y & \text{iff} & \exists w \in \mathrm{sim}_R(x),\ z \in w \diamondsuit y \quad \text{iff} \\
\exists w \in \mathrm{sim}_R(x),\ w \in z \diamondsuit^l y & \text{iff} & \exists w \in z \diamondsuit^l y,\ x \in \mathrm{sim}_L(w), \quad \text{iff} \\
x \in \mathrm{sim}_L(z \diamondsuit^l y), & &
\end{array}$$

as $x \in \mathrm{sim}_L(w)$ iff $w \in \mathrm{sim}_R(x)$ due to (4), (5). Hence we have

$$x \diamondsuit^l_{\mathrm{lo}} y = x \rightsquigarrow_T y, \quad x \diamondsuit^l_{\mathrm{up}} y = \mathrm{sim}_L(x \rightsquigarrow_T y)$$

by Lemma 2.4. The statement now follows by Theorem 2.8. $\square$

The above result allows us to construct an effective algorithm deciding whether a given regular language is (θ, sim)-bond-free.

Theorem 2.10 *Assume that* sim *is a rational relation. The following problem is decidable in quadratic time.*

Input: *NFA A.*

Output: *Y/N, depending on whether $L(A)$ is a (θ, sim)-bond-free language.*

Proof. Let T be a (fixed) transducer realizing the similarity relation sim. Recall from the previous subsection that given a λ-NFA A, the λ-NFA A_T accepts the language $T(L(A))$ and is of size $\mathcal{O}(|T||A|)$. Moreover, given a λ-NFA A, we can construct a λ-NFA of size $O(|A|)$ accepting the language $\theta(L(A))$. Using these facts and Lemma 2.5, we can construct a λ-NFA A' accepting the language

$$(L \rightsquigarrow_T \Sigma^*) \rightsquigarrow_{1+} \mathrm{sim}_L(\theta(L) \rightsquigarrow_T \Sigma^*)$$

in time $\mathcal{O}(|A|^2|T|)$, and so is its size. Then the result follows by Corollary 2.9. □

For the case where sim is one of the similarity relations $H_{d,k}$ or $Lev_{d,k}$ defined in Subsection 1.6, we have the following result.

Corollary 2.11 *The following problem is decidable in time $\mathcal{O}(dk|A|^2)$ (or $\mathcal{O}(dk^2|A|^2)$, respectively):*

Input: *NFA A, integers $d \geq 0$ and $k \geq 1$.*

Output: *Y/N, depending on whether $L(A)$ is a $(\theta, H_{d,k})$-bond-free (or $(\theta, Lev_{d,k})$-bond-free, respectively) language.*

Proof. We only show the case where the similarity relation is $Lev_{d,k}$. We explain how to construct a transducer T of size $O(dk^2)$ realizing this relation. Then, the claim will follow from Theorem 2.10. One can easily construct a transducer T_1 of size $O(d)$ such that $(x, y) \in R(T_1)$ iff $Lev(x, y) \leq d$. Let B be a DFA of size $O(k)$ accepting all words of length at least k. The required transducer T is equal to $(T_1 \uparrow B) \downarrow B$. Here we use the fact that, for each transducer S and λ-NFA C, one can construct the transducers $S \uparrow C$ and $S \downarrow C$, each of size $O(|S||C|)$, realizing the relations $\{(x, y) \in R(S) \mid x \in L(C)\}$ and $\{(x, y) \in R(S) \mid y \in L(C)\}$, respectively – see [22]. The case of $H_{d,k}$ is analogous. □

As shown above, there exists an effective algorithm for testing whether a particular regular language is (θ, sim)-bond-free, for any rational similarity relation sim. Now we address the same problem for the case of context-free languages. We note first that for most of the DNA language properties considered in [11, 19, 20] the problem is undecidable. As the $(\theta$,sim$)$-bond-free property seems to be rather general, it might be surprising that the same problem is decidable. This is shown in Theorem 2.13 using the next lemma.

Lemma 2.12 *For all words u and v, if* $\mathrm{sim}(u, v)$ *is true then there are subwords x and y of u and v, respectively, such that* $\mathrm{sim}(x, y)$ *is true and $|x|, |y| \leq t + l$.*

Proof. By inductive use of the condition (ii) in Definition 2.1, there must be subwords x and y of u and v, respectively, such that $\mathrm{sim}(x, y)$ and either $|x| \leq l$ or $|y| \leq l$. Then the condition (i) in that definition implies that $|x|, |y| \leq t + l$. □

Theorem 2.13 *Suppose that the similarity relation* sim *is computable. Then, it is decidable whether a given context-free language is $(\theta$,* sim*$)$-bond-free.*

Proof. Let L be the given language (by means of a context-free grammar, for instance). Observe that the language $L \rightsquigarrow_T \Sigma^*$, where $T = 1^*0^+1^*$, is the set of all subwords of L. This set is a context-free language, as T and Σ^* are regular languages [21]. Also, by definition, L is not (θ, sim)-bond-free iff there are nonempty subwords u, v of L such that $\mathrm{sim}(u, \theta(v))$ holds. In this case, Lemma 2.12 implies that there are subwords x and y of u and v, respectively, such that $|x|, |y| \le t + l$. Hence, to decide whether L is (θ, sim)-bond-free or not, it is enough to test for all such subwords x, y of L whether $\mathrm{sim}(x, \theta(y))$ holds. □

Recall that the relation $H_{d,k}$ defined in Subsection 1.6 is a similarity relation. Next we show that $Lev_{d,k}$ is a similarity relation as well, which implies that the above theorem applies when sim is one of these two relations – see Corollary 2.15.

Lemma 2.14 *The binary relation $Lev_{d,k}$ is a similarity relation with parameters $(d, d + k)$.*

Proof. An alignment of two words u and v is a sequence of pairs

$$((u_1, v_1), \ldots, (u_n, v_n))$$

such that $u = u_1 \cdots u_n$ and $v = v_1 \cdots v_n$ and, for each index i, u_i and v_i are in $\Sigma \cup \{\lambda\}$ with at least one of them being nonempty. Assume that $Lev_{d,k}(u, v)$ is true. Then $Lev(u, v) \le d$ and, therefore, there exists an alignment α as above such that there are at most d pairs (u_i, v_i) in α with $u_i \ne v_i$. Obviously, $\mathrm{abs}(|u| - |v|) \le d$. Now suppose that $|u|, |v| > k + d$. There is a prefix

$$\alpha_1 = ((u_1, v_1), \ldots, (u_j, v_j))$$

of α such that the word $x = u_1 \cdots u_j$ is of length $k + d$. This implies that the word $y = v_1 \cdots v_j$ is of length at least k. If $|y| \le k + d$ then the proper subwords x and y satisfy $Lev_{d,k}$, as required. If $|y| > k + d$ then, as before, there is a prefix $((u_1, v_1), \ldots, (u_r, v_r))$ of α_1 such that the word $y_1 = v_1 \cdots v_r$ is of length $k + d$, which implies that the word $x_1 = u_1 \cdots u_r$ is of length at least k. Hence, again, the proper subwords x_1 and y_1 satisfy $Lev_{d,k}$, as required. □

Corollary 2.15 *Let d and k be nonnegative integers with $k \ge 1$. It is decidable whether a given context-free language is $(\theta, H_{d,k})$-bond-free (or $(\theta, Lev_{d,k})$-bond-free).*

2.3 Decidability of maximality for regular languages

In this subsection we use the tools developed in [18, 20] to decide whether a given regular language is a maximal (θ, sim)-bond-free subset of some given regular language. These tools involve the concept of a language inequation.

Let L and M be two languages and let $\diamondsuit$ be a binary word operation. Consider an inequation of the form

$$X \diamondsuit L \subseteq X^c, \quad X \subseteq M. \tag{7}$$

In [18], it is shown that this inequation is equivalent to

$$X \diamondsuit^r X \subseteq L^c, \quad X \subseteq M. \tag{8}$$

A language $S_{\max}$ is a *maximal solution* of (7), or equivalently of (8), if $S_{\max}$ is a solution (i.e. (7) holds true for $X = S_{\max}$) and for each x in $M - S_{\max}$, the language $S_{\max} \cup \{x\}$ is not a solution.

Let S be a solution of (7), or equivalently of (8). We call the language

$$R = M - (S \cup S \diamondsuit L \cup S \diamondsuit^l L)$$

the *residue* of S.

Theorem 2.16 [20] *Let S be a solution of (7), let R be the residue of S, and let $Q = \{z \in \Sigma^* \mid z \in z \diamondsuit L\}$. Then S is maximal iff $R - Q = \emptyset$.*

Lemma 2.17 *For each strictly bond-free property $\mathcal{P}$ there is a binary word operation $\boxdot_{\mathcal{P}}$ such that for a language $L \subseteq \Sigma^*$, $\mathcal{P}(L) = \text{true}$ iff*

$$L \boxdot_{\mathcal{P}} L = \emptyset. \tag{9}$$

Proof. By Theorem 2.8, for $x, y \in \Sigma^*$ we have

$$x \boxdot_{\mathcal{P}} y = (x \diamondsuit^l_{\mathrm{lo}} \Sigma^*) \rightsquigarrow_{1^+} (\theta(y) \diamondsuit^l_{\mathrm{up}} \Sigma^*). \tag{10}$$

$\square$

In the case where $\mathcal{P}$ is the (θ, sim)-bond-free property, we shall use the symbol $\boxdot$ for $\boxdot_{\mathcal{P}}$.

Lemma 2.18 *The left and right inverses of $\boxdot$ are as follows.*

(i) $z \boxdot^l y = (z \sqcup\!\sqcup_{1^+} \mathrm{sim}_L(\theta(y) \rightsquigarrow_T \Sigma^*)) \sqcup\!\sqcup_T \Sigma^*$,

(ii) $x \boxdot^r z = \theta(\mathrm{sim}_R((x \rightsquigarrow_T \Sigma^*) \rightsquigarrow_{0^+} z) \sqcup\!\sqcup_T \Sigma^*)$.

Proof. In [20] it is shown that

(i) $z \boxdot^l_{\mathcal{P}} y = (z \sqcup\!\sqcup_{1^+} (\theta(y) \diamondsuit^l_{\mathrm{up}} \Sigma^*)) \diamondsuit_{\mathrm{lo}} \Sigma^*$,

(ii) $x \boxdot^r_{\mathcal{P}} z = \theta(((x \diamondsuit^l_{\mathrm{lo}} \Sigma^*) \rightsquigarrow_{0^+} z) \diamondsuit_{\mathrm{up}} \Sigma^*)$.

From the proof of Corollary 2.9, recall that

$$x \diamondsuit_{\mathrm{lo}} y = x \sqcup\!\sqcup_T y, \quad x \diamondsuit_{\mathrm{up}} y = \mathrm{sim}_R(x) \sqcup\!\sqcup_T y$$

and

$$x \diamondsuit^l_{\mathrm{lo}} y = x \rightsquigarrow_T y, \quad x \diamondsuit^l_{\mathrm{up}} y = \mathrm{sim}_L(x \rightsquigarrow_T y)$$

The above observations imply that the first claim is correct. For the second claim note that a word w is in $x \diamondsuit^l_{\mathrm{up}} y = \mathrm{sim}_L(x \rightsquigarrow_T y)$ iff there is a word u such that $u \in \mathrm{sim}_R(w)$ and $y \in (u \rightsquigarrow^l_T v)$, iff $y \in (\mathrm{sim}_R(w) \sqcup\!\sqcup_T v)$ iff $y \in w \diamondsuit_{\mathrm{up}} v$. $\square$

Theorem 2.19 *Assume that the similarity relation* sim *is rational. Then the following problem is decidable.*

Input: *NFAs A and B such that $L(A)$ is a (θ, sim)-bond-free subset of $L(B)$.*

Output: *Y/N, depending on whether $L(A)$ is a maximal (θ, sim)-bond-free subset of $L(B)$.*

Proof. Let $L = L(A)$ and $M = L(B)$ and D be a transducer realizing sim. Lemma 2.17 implies that $L \boxdot L = \emptyset$, and (8) implies that $L \boxdot^r \Sigma^* \subseteq L^c$. By Theorem 2.16, we have that L is maximal iff

$$M \cap (L \cup L \boxdot^r \Sigma^* \cup \Sigma^* \boxdot^l L)^c \cap Q^c = \emptyset,$$

where we have used the fact that $\boxdot^{rl} = \boxdot^{l\prime}$ [18]. The language $(L \cup L \boxdot^r \Sigma^* \cup \Sigma^* \boxdot^l L)^c$ can be computed from A, B and D using the following facts for arbitrary λ-NFAs C, C' and transducer F and regular trajectory set S: (i) one can compute λ-NFAs accepting $L(C) \sqcup\!\sqcup_S L(C')$ and $L(C) \rightsquigarrow_S$

$L(C')$ – see Lemma 2.5; (ii) one can compute a λ-NFA C_F accepting the language $\{w \mid w \in F(u),$ for some $u \in L(C)\}$ [22]; (iii) one can compute a transducer F^{-1} such that $w \in F(u)$ iff $u \in F^{-1}(w)$ [33]; and (iv) one can compute λ-NFAs accepting the languages $L(C)^c$ and $\theta(L(C))$.

We need now to compute the set Q^c. For this, observe first that

$$u \boxdot v = \begin{cases} \lambda, & \text{if } u \text{ and } v \text{ contain subwords } x \text{ and } y \text{, respectively,} \\ & \text{such that } \mathrm{sim}(x, \theta(y)) \text{ holds;} \\ \emptyset, & \text{otherwise.} \end{cases}$$

Then, by Lemma 2.12, we see that $u \boxdot v = \lambda$ iff u and v contain subwords x and y of length at most $t+l$ such that $\mathrm{sim}(x, \theta(y))$ holds. As Q consists of all words w such that $w \boxdot w \neq \emptyset$, it follows that each such w must be of the form $x_1 w_1 y_1 = x_2 w_2 y_2$ with $|x_1| \leq |x_2|$ and $|w_1|, |w_2| \leq t+l$ and $\mathrm{sim}(w_1, \theta(w_2))$. Equivalently, w is in Q iff one of the following holds.

- $w = x w_1 y w_2 z$ with $|w_1|, |w_2| \leq t+l$ and $\mathrm{sim}(w_1, \theta(w_2))$;
- $w = x_1 svp y_2$ with $|v| > 0$ and $|w_1|, |w_2| \leq t+l$ and $\mathrm{sim}(w_1, \theta(w_2))$, where $w_1 = sv$ and $w_2 = vp$;
- $w = x_1 svp y_1$ with $|v| > 0$ and $|w_1|, |w_2| \leq t+l$ and $\mathrm{sim}(w_1, \theta(w_2))$, where $w_1 = svp$ and $w_2 = v$.

Note that, for every word u, the set $D(u)$ is finite and computable. Thus, Q can be computed as the union $Q_1 \cup Q_2 \cup Q_3$, where

$$\begin{aligned} Q_1 &= \Sigma^* \Big(\bigcup_{|w_1| \leq t+l} \bigcup_{u \in D(w_1)} (w_1 \Sigma^* \theta(u)) \Big) \Sigma^*, \\ Q_2 &= \Sigma^* \Big(\bigcup_{(s,v,p) \in P_2} svp \Big) \Sigma^*, \\ Q_3 &= \Sigma^* \Big(\bigcup_{(s,v,p) \in P_3} svp \Big) \Sigma^*, \end{aligned}$$

where P_2 is the set of all triples (s, v, p) with $|sv| \leq t+l$, $|v| > 0$, $\theta(vp) \in D(sv)$, and P_1 is the set of all triples (s, v, p) with $|svp| \leq t+l$, $|v| > 0$, $\theta(vp) \in D(svp)$. As P_1 and P_2 are computable, the set Q is computable as well. $\square$

3 Decidability of Maximality in the Hamming Case

The decision method for maximality presented in Subsection 2.3 is not of polynomial time. In this section, however, we are able to show a polynomial time algorithm for testing whether a given regular langauge is $(\theta, H_{d,k})$-bond-free, for $d = 0$ or $d = 1$. Moreover, we introduce the subword closure operation, $\otimes$, which plays an important role in the sequel.

3.1 Some notation

We assume that the reader is familiar with the notation of Subsection 1.3 as well as the terminology about automata and transducers in Subsection 2.1. In particular, we assume that θ is an arbitrary antimorphic involution and τ is the DNA involution.

Let k be a positive integer and let L be a language. We use the notation

$$\mathrm{Sub}_k(L)$$

to represent the set of all subwords of length k of L. Let d be a nonnegative integer and let S be a language containing only words of the same length. The *Hamming ball* $H_d(S)$ of S is the set $\{v \mid H(v,z) \le d, \text{ for some } z \in S\}$. Note that $H_d(S) = S$ when $d = 0$.

Lemma 3.1 *Let k and d be integers with $k \ge 1$ and $d \ge 0$ and let L be a language. The following statements hold true.*

1. $\theta(L^c) = \theta(L)^c$.
2. $\mathrm{Sub}_k(\theta(L)) = \theta(\mathrm{Sub}_k(L))$.
3. $H(\theta(u), \theta(v)) = H(u, v)$, *for all words u and v.*
4. $H_d(\theta(L)) = \theta(H_d(L))$.

Proof. The proof is based on the definitions of the concepts involved and is left to the reader. □

Lemma 3.2 *Let $d \ge 0$ and $k \ge 1$ be integers. A language L is $(\theta, H_{d,k})$-bond-free if and only if*

$$\theta(\mathrm{Sub}_k(L)) \cap H_d(\mathrm{Sub}_k(L)) = \emptyset. \tag{11}$$

Proof. We prove the 'if' part and we leave to the reader the proof of the 'only if' part. Let w_1 and w_2 be subwords of L of length $\ge k$. We show that $H(w_1, \theta(w_2)) > d$. The words w_1 and w_2 can be written as u_1z_1 and z_2u_2, respectively, with $|u_1| = |u_2| = k$. As u_1 is in $\mathrm{Sub}_k(L)$ and $\theta(u_2)$ is in $\theta(\mathrm{Sub}_k(L))$, the assumption of the 'if' part implies that $\theta(u_2) \notin H_d(u_1)$ and, therefore, $H(\theta(u_2), u_1) > d$. Hence, also $H(\theta(w_2), w_1) > d$ holds. □

In the literature on DNA encodings, and in coding theory in general, the set of words that are involved in the application of interest are usually formed by concatenating shorter words of a certain fixed length. Following this practice, we consider languages that are subsets of $(\Sigma^k)^+$, for some positive integer k. We call such languages *k-block languages*. Naturally, any regular k-block language can be represented by a special type of lazy DFA [32], which we call *k-block DFA*. This is a trim deterministic finite automaton such that, for every production $pu \to q$, the word u is of length k and there is no other production of the form $pu \to q'$ – this ensures that the automaton is deterministic. The *size* $|A|$ of a k-block DFA A is the quantity kn, where n is the number of productions in A. As an example, consider the 2-state k-block DFA with the following set of productions, P, where s is the start state, f is the only final state, and K is a finite code whose words are of length k.

$$P = \{su \to f \mid u \in K\} \cup \{fu \to f \mid u \in K\}.$$

Clearly, the language accepted by this k-block DFA is K^+.

3.2 The subword closure operation and the algorithm

Here we consider the problem of deciding *in polynomial time* whether a given regular k-block language that is $(\theta, H_{d,k})$-bond-free is maximal with this property. If the language is given in terms of an ordinary DFA the problem appears to be intractable. On the other hand, we are able to solve this problem affirmatively in the cases of $d = 0$ and $d = 1$, and when the language in question is given in terms of a k-block DFA – see Theorem 3.8. We remind the reader that, in the case of $d = 0$, the property coincides with $\mathbf{P1}[k]$ – see Subsection 1.4. Next we illustrate the concept of maximality with an example.

Example 3.3 Consider the code $K_1 = \{AA, AC, CA, CC\}$ over the DNA alphabet and the 2-block language K_1^+. Let $S_1 = \mathrm{Sub}_2(K_1^+)$. Then, S_1 is equal to K_1 and $S_1 \cap \tau(S_1)$ is empty. Hence, the language K_1^+ is a $(\tau, H_{0,2})$-bond-free subset of $(\Sigma^2)^+$. Moreover, there is no word v in $\Sigma^2 - K_1$ such that the language $(K_1 \cup \{v\})^+$ is $H_{0,2}$-bond-free. For example, if $v = AG$ then GC would be a subword of length 2 of $(K_1 \cup \{v\})^+$ such that $GC = \tau(GC)$. On the other hand, it is possible to add AG as a subword with the constraint that AG cannot be followed by CA or CC. In fact we can add also GA as a subword, provided that GA cannot be preceded by AC, CC, or AG. More specifically, consider the language L_2 accepted by the 2-block DFA $A_2 = (\Sigma, \{1, 2, 3, 4\}, 1, \{2, 3, 4\}, P_2)$, where the set of productions P_2 is equal to

$\{1u \to 2,\ 1v \to 3,\ 1(AG) \to 4 \mid u = AA, CA, GA \text{ and } v = AC, CC\} \cup$
$\{2u \to 2,\ 2v \to 3,\ 2(AG) \to 4 \mid u = AA, CA, GA \text{ and } v = AC, CC\} \cup$
$\{3u \to 2,\ 3v \to 3,\ 3(AG) \to 4 \mid u = AA, CA \text{ and } v = AC, CC\} \cup$
$\{4(AG) \to 4,\ 4(AC) \to 3,\ 4(AA) \to 2\}$.

The language L_2 is a proper superset of K_1^+ and is a $(\tau, H_{0,2})$-bond-free subset of $(\Sigma^2)^+$. In fact in the next example we show that L_2 is maximal using Lemma 3.6.

The operation of subword closure plays an important role in the sequel. Next we give the formal definition and provide a few basic properties of this operation.

Definition 3.4 *Let S be a language containing only words of the same length k, for some positive integer k. The* subword closure $S^{\otimes}$ *of S is the set*

$$\{w \in \Sigma^* \mid |w| \geq k,\ \mathrm{Sub}_k(w) \subseteq S\}$$

.

Lemma 3.5 *Let S be a language containing only words of the same length k, for some positive integer k. The following statements hold true.*

1. *$S = \mathrm{Sub}_k(S^{\otimes})$.*
2. *Let S_1 be a language containing only words of the same length k. Then $S_1 \subseteq S$ iff ${S_1}^{\otimes} \subseteq S^{\otimes}$. This implies that, if $S_1 \neq S$ then ${S_1}^{\otimes} \neq S^{\otimes}$*
3. *If $S = \mathrm{Sub}_k(L)$, for some language L, then $L \subseteq S^{\otimes}$.*

Proof. The proof is based on the definition of $S^{\otimes}$ and is left to the reader. □

Lemma 3.6 *Let M be a language satisfying the condition $\mathrm{Sub}_k(M) \subseteq M \subseteq \Sigma^k\Sigma^*$ and let L be a $(\theta, H_{d,k})$-bond-free subset of M, for some integers $k \geq 1$ and $d \geq 0$. The following statements hold true.*

1. *If the language L is not a maximal $(\theta, H_{d,k})$-bond-free subset of M then the following set is nonempty:*

$$(M-L) \bigcap \left(\mathrm{Sub}_k(L)^{\otimes} \bigcup \left(\theta(H_d(\mathrm{Sub}_k(L))^c) \cap \{w \in \Sigma^k \mid H(w,\theta(w)) > d\}\right)\right).$$

2. *Conversely, if a word w belongs to the above set then the language $L \cup \{w\}$ is a $(\theta, H_{d,k})$-bond-free subset of M and, therefore, L is not a maximal $(\theta, H_{d,k})$-bond-free subset of M.*

Proof. Let $S = \mathrm{Sub}_k(L)$ and let $D = \{w \in \Sigma^k \mid H(w,\theta(w)) > d\}$. By (11), the assumption that L is $(\theta, H_{d,k})$-bond-free is equivalent to the condition

$$\theta(S) \cap H_d(S) = \emptyset.$$

We begin with the first statement. The assumption that L is not maximal implies that there is a word w *of minimum length* with the property that $w \in M - L$ and

$$\theta(S \cup \mathrm{Sub}_k(w)) \,\cap\, H_d(S \cup \mathrm{Sub}_k(w)) \;=\; \emptyset.$$

We need to show that the set $(M-L) \cap (S^{\otimes} \cup (\theta(H_d(S)^c) \cap D))$ is not empty. We distinguish two cases. Firstly, assume that w is of length k. Then, $\mathrm{Sub}_k(w) = \{w\}$ and $\theta(w) \notin H_d(w)$ and $\theta(w) \notin H_d(S)$. This implies that $w \in D \cap \theta(H_d(S)^c)$. Now consider the case where $|w| > k$. We show that $v \in S$, for all words v in $\mathrm{Sub}_k(w)$. Let $v \in \mathrm{Sub}_k(w)$ and suppose that v is not in S. Then $|v| = k$ and $v \notin L$. Moreover, the assumption about M implies that the word v must be in M. As $S \cup \{v\}$ is a subset of $S \cup \mathrm{Sub}_k(w)$, one has that $\theta(S \cup \{v\}) \cap H_d(S \cup \{v\}) = \emptyset$, which contradicts the assumption about the choice of w. Hence, it is the case that $\mathrm{Sub}_k(w) \subseteq S$, which implies that w must be in $S^{\otimes}$.

Now we prove the second statement. Let w be a word in $(M-L) \cap \theta(H_d(S)^c) \cap D$. Then $|w| = k$ and $H(w,\theta(w)) > d$ and $w \notin H_d(\theta(S))$ and, therefore, $H(\theta(w), z) > d$ for all words z in S. We claim that

$$\theta(\mathrm{Sub}_k(L_1)) \cap H_d(\mathrm{Sub}_k(L_1)) \;=\; \emptyset,$$

where $L_1 = L \cup \{w\}$. Indeed the claim follows easily when we note that $\mathrm{Sub}_k(L_1) = S \cup \{w\}$. Hence, L_1 is a $(\theta, H_{d,k})$-bond-free subset of M. Now let u be a word in $(M-L) \cap S^{\otimes}$ and let $L_2 = L \cup \{u\}$. As $\mathrm{Sub}_k(u)$ is a subset of S, we have that $\mathrm{Sub}_k(L_2) = S$, which implies that $\theta(\mathrm{Sub}_k(L_2)) \cap H_d(\mathrm{Sub}_k(L_2))$ is empty. Hence, L_2 is a $(\theta, H_{d,k})$-bond-free subset of M. □

In this paper we shall apply the above lemma for the cases where M is equal to $(\Sigma^k)^+$ or $\Sigma^k\Sigma^*$.

Example 3.7 We show that the language L_2 defined in the previous example is a maximal $(\tau, H_{0,2})$-bond-free subset of $(\Sigma^2)^+$. Assume that L_2 is not maximal. Let $S_2 = \mathrm{Sub}_2(L_2)$ and let $D = \{w \in \Sigma^2 \mid H(w,\tau(w)) > 0\}$. Then S_2 consists of the words AA, AC, AG, CA, CC, GA. As

$$((\Sigma^2)^+ - L) \cap D = \tau(S_2) \;\text{ and }\; \tau(H_0(S_2)^c) = \tau(S_2)^c,$$

the first statement of Lemma 3.6 implies that there is a word w in the set $((\Sigma^2)^+ - L) \cap {S_2}^{\otimes}$. Then the second statement of the lemma implies that $L_2 \cup \{w\}$ is $(\tau, H_{0,2})$-bond-free and, therefore, no subword of w can be in $\tau(S_2)$. In particular, none of CG, GC, and GG can be subwords of w. This leads to a contradiction as follows. Let $w = w_1 \cdots w_m$ with each subword w_i being of length 2. As w is not in L there is a prefix $u = w_1 \cdots w_{i-1}$ of w, for some $i > 1$, such that u takes the automaton A_2 to a state q in $\{3,4\}$ and there is no production of the form $qw_i \to p$. In particular, this is possible when $q = 3$ and $w_i = GA$, or $q = 4$ and $w_i \in \{CA, CC, GA\}$. In the first case u must end with C and in the second one u must end with G, which implies that one of CG, GC, and GG must be a subword of w.

For the sake of simplicity, in the next theorem we assume that θ is the DNA involution. The theorem will remain valid, however, even for an arbitrary antimorphic involution if we adjust the time complexity estimates.

Theorem 3.8 *Let d be a fixed value in $\{0, 1\}$. The following problem is computable in polynomial time.*

Input: *k-block DFA A such that $L(A)$ is a $(\theta, H_{d,k})$-bond-free subset of $(\Sigma^k)^+$.*

Output: *Y/N, depending on whether $L(A)$ is maximal with that property. Moreover, if $L(A)$ is not maximal, output a minimal-length word $w \in (\Sigma^k)^+ - L(A)$ such that $L(A) \cup \{w\}$ is a $(\theta, H_{d,k})$-bond-free subset of $(\Sigma^k)^+$.*

In particular, the time complexity $t(|A|)$ is bounded as follows:

$$t(|A|) = \begin{cases} O(k|A|^3), & \text{if } k \text{ is odd and } d = 0; \\ O(|A|^6), & \text{if } k \text{ is even and } d = 0; \\ O(k^3|A|^6), & \text{if } d = 1. \end{cases}$$

The proof of the above theorem is based on several intermediate lemmata, some of which might be of interest in their own right. In the proofs we use the following facts and notation.

- Every NFA A can be converted to an equivalent trim NFA of size $O(|A|)$ in time $O(|A|)$. Moreover, if A accepts only words of the same length, we can convert it to an equivalent NFA of size $O(|A|)$ with a single final state in time $O(|A|)$.
- Given two NFA's A and B we can use the standard product construction to obtain the NFA $A \cap B$ of size $O(|A||B|)$ accepting the language $L(A) \cap L(B)$. If A is a DFA we can construct a DFA A^c of size $O(|A|)$ accepting the language $L(A)^c$.
- If A is an NFA and p and q are states of A then $A_{p,q}$ denotes the NFA which is identical to A except that p is the start state and q is the final state of $A_{p,q}$. If A is trim with start state s and a single final state f then $\theta(A)$ is the trim NFA of size $O(|A|)$ accepting the language $\theta(L(A))$ such that $\theta(A)$ results from $A_{f,s}$ by replacing each production $qa \to r$ with $r\theta(a) \to q$.
- If a trim NFA A accepts only words of the same length k, for some positive integer k, then the state set of A can be partitioned into $k + 1$ levels (sets of states) such that the start state is the only state in level 0, level k consists of the final states of A, and if $qa \to r$ is a production of A with q being at level $l < k$ then the state r is in level $l + 1$. If A is a DFA then there is a DFA A^{-k} of size $O(|A|)$ accepting the language $\Sigma^k - L(A)$. Moreover, A^{-k} can be constructed from A in time $O(|A|)$.

Lemma 3.9 *For every k-block DFA A, there is an (ordinary) DFA $\hat{A}$ of size $O(|A|)$ such that $L(\hat{A}) = L(A)$. Moreover, $\hat{A}$ can be constructed from A in time $O(|A|)$.*

Proof. For each state q of A, let P_q be the set of productions of A of the form $qu \to r$. The set of productions of $\hat{A}$ consists of all the productions computed by the following procedure: For each state q of A compute a special type of trie T_q, called quasi-trie here, as follows. The start state of T_q is q. For each production $qu \to r$ in P_q, insert into T_q the word u as one would do in an ordinary trie – this will add into T_q all necessary productions – with the following modification: if u is of the form $u_1 a$ with a in Σ, then instead of the production $[u_1]_q\, a \to [u]_q$ insert into the quasi-trie T_q the production $[u_1]_q\, a \to r$. It is evident that this procedure requires time linear with respect to $|A|$. Obviously the state set of $\hat{A}$ consists of the states of A plus all the states $[w]_q$ appearing in some quasi-trie T_q. The start and final states of $\hat{A}$ are exactly those of A. Moreover, for each pair of states (q, r) of A, the set of words accepted by $\hat{A}_{q,r}$ is exactly the set of words $\{u \mid qu \to r$ is a production of $A\}$. This implies that $L(\hat{A}) = L(A)$. □

Lemma 3.10 *Let T be a trie accepting only words of the same length. There is a DFA $T^{\otimes}$ of size $O(|T|)$ accepting the language $L(T)^{\otimes}$. Moreover, $T^{\otimes}$ can be constructed from T in time $O(|T|)$.*

Proof. For a word w of the form aw_1, with $a \in \Sigma$, we denote by $\dot{w}$ the word w_1. The states of $T^{\otimes}$ are exactly those of T and the start and final states of $T^{\otimes}$ are exactly those of T as well. The DFA $T^{\otimes}$ contains all the productions of T plus, for each final state $[w]$ of T – note that there are no productions in T of the form $[w]a \to [u]$ – and for each a in Σ, the production $[w]a \to [\dot{w}a]$ iff $[\dot{w}a]$ is a final state of T. It is evident that this process can be performed in time $O(|T|)$. We show next that $L(T)^{\otimes} \subseteq L(T^{\otimes})$ and we leave the proof of the converse inclusion to the reader.

First note that every word w in $L(T)^{\otimes}$ is of the form $ua_1 \cdots a_m$ with $|u| = k$, $m \geq 0$, and each a_i is in Σ. Let $u_0 = u$ and $u_i = \dot{u}_{i-1} a_i$ for all $i = 1, \ldots, m$. Then, for each index i, the word u_i is in $\mathrm{Sub}_k(w)$, which implies that u_i is in $L(T)$. Thus, on input w the DFA $T^{\otimes}$ will behave as follows: the prefix u of w will take $T^{\otimes}$ to the final state $[u]$ and, in general, if $ua_1 \cdots a_{i-1}$ takes $T^{\otimes}$ to the final state $[u_{i-1}]$ then, as $\dot{u}_{i-1} a$ is in $L(T)$, the prefix $ua_1 \cdots a_i$ of w would take $T^{\otimes}$ to the final state $[u_i]$. Thus, w will be accepted by $T^{\otimes}$ and, therefore, w is also in $L(T^{\otimes})$. □

Lemma 3.11 *Let A be a k-block DFA. There is a trie $T_k(A)$ of size $O(|A|^2)$ accepting the language of all subwords of length k of $L(A)$. Moreover, $T_k(A)$ can be constructed from A in time $O(|A|^2)$.*

Proof. Let Q be the set of states and let n be the number of transitions of A. Then $n = |A|/k$. For each state q in Q, let P_q be the set of transitions of the form $pu \to q$ and let N_q be the set of transitions of the form $qv \to r$. Clearly, $\mathrm{Sub}_k(L(A))$ is the union of $\mathrm{Sub}_k(uv)$, for all pairs of words u and v appearing in the sets P_q and N_q, respectively, of some state q. Using this observation, the trie $T_k(A)$ is constructed as follows. For each state q in Q and for every transitions $pu \to q$ in P_q and $qv \to r$ in N_q, add all subwords of length k of uv into the trie. Note that each pair (u, v) contributes at most $k + 1$ different subwords in $\mathrm{Sub}_k(L(A))$, and there are $\sum_{q \in Q} |P_q||N_q|$ such pairs (u, v). Hence, the number of words inserted into the trie is $(k + 1) \sum_{q \in Q} |P_q||N_q|$. As $\sum_{q \in Q} |N_q| = n$ and $|P_q| \leq n$, for all q, it follows that there are at most $(k + 1)n^2$ words of length k inserted into the trie $T_k(A)$. Hence, $|T_k(A)| = O(|A|^2)$. □

Lemma 3.12 *Consider the following problem.*

Input: *Integers $d \geq 0$ and $k \geq 1$, and NFA A accepting only words of length k.*

Output: *Y/N, depending on whether there is a word w in $L(A)$ such that $H(w, \theta(w)) > d$. Moreover, output a word with this property (if it exists).*

There is an algorithm that computes the problem such that the time complexity $t(|A|, d)$ of the algorithm is bounded as follows:

$$t(|A|, d) = \begin{cases} O(|A|), & \text{if } k \text{ is odd and } d = 0; \\ O(|A|^3 + \frac{d}{2}|A|^3), & \text{otherwise.} \end{cases}$$

Proof. Let s be the start state of A. We can assume that A is a trim NFA with a single final state f – if not, we can convert A to an equivalent such NFA. First note that $w \neq \theta(w)$ for any word of odd length and, therefore, if $d = 0$ and k is odd then $H(w, \theta(w)) > d$ for all words w in $L(A)$. In this case the algorithm consists of picking any path from state s to state f and outputting the word corresponding to this path. In the sequel we assume that k is even or $d > 0$. Let

$$t = \begin{cases} \lfloor d/2 \rfloor, & \text{if } k \text{ is even;} \\ \lfloor (d-1)/2 \rfloor, & \text{if } k \text{ is odd.} \end{cases}$$

One can verify that for any word w, $H(w, \theta(w)) > d$ iff there are words u and u' of the same length such that $w = uu'$ (if $|w|$ is even) or $w \in u\Sigma u'$ (if $|w|$ is odd), and $H(u, \theta(u')) > t$. Now let T_t be a transducer of size $O(t+2)$ with the property that, for all words x and y, $x \in T_t(y)$ iff $H(x, y) > t$. Based on the above observations one can verify that there is a word w in $L(A)$ with $H(w, \theta(w)) > d$ iff there is a state q at level $\lfloor k/2 \rfloor$ and a state r at level $\lceil k/2 \rceil$ that is adjacent to q – in fact $r = q$ if k is even – such that

$$T_t(L(A_{s,q})) \cap \theta(L(A_{r,f})) \neq \emptyset.$$

For the sake of simplicity we describe the algorithm only for the case where k is odd – if k is even then the loop at Step 2 iterates for each state q in Q_1 and assumes that $r = q$ and $a = \lambda$.

1. Let Q_1 be the set of states at level $\lfloor k/2 \rfloor$ and let P be the set of productions of the form $qa \to r$ with $q \in Q_1$;

2. For each production $qa \to r$ in P repeat steps (A)–(B):

- **(A)** Construct the NFA $B = (A_{s,q})_{T_t} \cap \theta(A_{r,f})$;
- **(B)** If there is a **path** from the start to the final state of B do the following:
  - **B1** output **YES**;
  - **B2** Let v be the word corresponding to the **path** and let D_v be a DFA of size $O(|v|)$ accepting $\{v\}$;
  - **B3** Let C be the NFA $(D_v)_{T_t} \cap A_{s,q}$;
  - **B4** Let u be the word corresponding to any path from the start to the final state of C;
  - **B5** output the word $\mathbf{u a \theta(v)}$ and quit;

3. Output **NO**.

Suppose that for some production $qa \to r$ there is a path in Step 2(B), then the word v corresponding to this path belongs to $L((A_{s,q})_{T_t}) \cap \theta(L(A_{r,f}))$. In this case, $\theta(v)$ is in $L(A_{r,f})$ and there is a word $u \in L(A_{s,q})$ such that $v \in T_t(u)$. This implies that $H(u, v) > t$ and $u \in T_t(v)$ and, therefore, the language accepted by the NFA C in Step B3 is nonempty. Moreover, as $H(u, \theta(\theta(v))) > t$, one has that $H(ua\theta(v), \theta(ua\theta(v))) > d$. This establishes the correctness of steps B2–B5. Regarding the time complexity, first note that Step 1 can be performed in time $O(|A|)$ and that the loop in Step 2 iterates at most $O(|A|)$ times. In each iteration, the algorithm computes the NFA B in time $O((t+2)|A|^2)$. If $L(B)$ is nonempty the NFA C is computed in time $O(k(t+2)|A|)$ and then the algorithm terminates. Hence the algorithm operates in time $O(|A|^3 + t|A|^3)$ in the worst case. □

Proof of Theorem 3.8.
The algorithm is as follows. We use S as a shorthand notation for $\mathrm{Sub}_k(L(A))$.

1. If $d = 0$ let A_1 be the trie $T_k(A)$. If $d = 1$ let A_1 be the trie obtained by modifying the construction of $T_k(A)$ as follows: For each word x in S, insert into $T_k(A)$ the word x as well as all words y that differ from x in exactly one position – there are exactly $k(|\Sigma| - 1)$ such words. Note that $L(A_1) = H_d(S)$ and $|A_1| = O(|A|^2)$ if $d = 0$, or $|A_1| = O(k|A|^2)$ if $d = 1$.

2. Let F be the set of words in $L(A)$ of length k. This set can be computed in time $O(|A|)$ and is of cardinality $O(|A|/k)$.

3. Let B_0 be the trie that results if we insert into A_1 the words of $\theta(F)$. This process requires time $O(|F|k)$ and the resulting trie is of size $O(|F|k + |A_1|)$, which is simply $O(|A_1|)$. Note that $L(B_0) = (H_d(S) \cup \theta(L(A))) \cap \Sigma^k$.

4. Let B_1 be a trim DFA of size $O(|A_1|)$, with a single final state, that is equivalent to ${B_0}^{-k}$.

5. Let $A_2 = \theta(B_1)$. Note that $L(A_2) = \Sigma^k \cap \theta(H_d(S)^c) \cap L(A)^c$ and $|A_2| = O(|A_1|)$.

6. Run the algorithm of Lemma 3.12 on input (d, k, A_2). If that algorithm returns YES and a word w, then output **NO** and the word **w**, and quit.

7. Let B be a DFA of size $O(k)$ accepting the language $(\Sigma^k)^+$.

8. Let $A_3 = (B \cap \hat{A}^c) \cap T_k(A)^{\otimes}$. Note that $L(A_3) = ((\Sigma^k)^+ - L(A)) \cap S^{\otimes}$ and $|A_3| = O(k|A|^3)$.

9. Find a shortest path from the start to a final state of A_3. If such a path exists then output **NO** and the **word** corresponding to that path. Else output **YES**.

The correctness and time complexity of the algorithm follow from the preceding lemmata. In particular, we note that if $L(A)$ is not maximal then any minimal-length word that can be added into $L(A)$ must be of length at least k. In Step 6, the algorithm checks for candidate words of length k in the set $L(A_2) \cap \{w \in \Sigma^* \mid H(w, \theta(w)) > d\}$. If none is found, it continues with steps 8 and 9 looking for the shortest word, if any, in the set $L(A_3)$. By Lemma 3.6, if no word is found in steps 6 and 9 the algorithm correctly outputs that $L(A)$ is maximal. □

4 Construction Methods for the Hamming Case

In this section we describe methods for constructing $(\tau, H_{d,k})$-bond-free languages. We focus on languages that are subsets of $(\Sigma^k)^+$ or $\Sigma^k\Sigma^*$. *We assume throughout that k and d are integers, with $k \geq 1$ and $0 \leq d < k$, and τ is the DNA involution.*

4.1 Direct Methods

Here we consider analytical methods that do not rely on previously constructed languages. The first method is based on the concept of a template introduced in [2]. The operation '$\cdot$' between two bits in {0,1} is defined as follows

$$0 \cdot 0 = C,\ 0 \cdot 1 = G,\ 1 \cdot 0 = T,\ 1 \cdot 1 = A.$$

This operation is extended to binary words of the same length in a natural manner. For example, $0101 \cdot 0110 = CAGT$. It can be shown that $b_1 \cdot b_2 = b_3 \cdot b_4$ iff $b_1 = b_3$ and $b_2 = b_4$, for all bits b_1, b_2, b_3, b_4. A *k-template* is any binary word of length k. If x is a k-template and E is subset of $\{0, 1\}^k$ then $x \cdot E = \{x \cdot v \mid v \in E\}$. The construction method of [2] involves choosing a k-template

x and a code E such that $x \cdot E$ satisfies a desired property. In our case, we are interested in k-templates x such that

$$H(x_2 x_1, (x_4 x_3)^R) > d, \tag{12}$$

for all prefixes x_1 and x_3 of x and suffixes x_2 and x_4 of x.

Theorem 4.1 *Let x be a k-template satisfying (12). Then the language $(x \cdot \{0,1\}^k)^+$ is $(\tau, H_{d,k})$-bond-free.*

Proof. Let β be the *binary antimorphic* involution such that $\beta(0) = 1$ and $\beta(1) = 0$. The proof is based on the facts that $\tau(x_1 \cdot g_1) = x_1^R \cdot \beta(g_1)$ and $H(x_1 \cdot g_1, x_2 \cdot g_2) \geq H(x_1, x_2)$, for all binary words x_1, g_1, x_2, g_2 of the same length. □

Observe that the cardinality of the code $x \cdot \{0,1\}^k$ is 2^k. The advantage of the method of templates is that properties of the template x, which is a simple object, are passed gracefully to the code $x \cdot E$, where E is any subset of $\{0,1\}^k$. This is evident, for instance, in Theorem 4.1. Moreover, by choosing a template x with the same number of 0's and 1's, all the words of the code $x \cdot E$ have a 50% GC-ratio. We note that many of the templates listed in [2] satisfy (12).

We introduce now another direct construction method. In this method, the bond-free language is again of the form K^+, where K is a code of fixed-length. Moreover there is a set I of positions in which the codeword symbols are always in $\{A, C\}$. The method is described more formally in the next theorem. The notation $k \,\%\, 2$ stands for the remainder of the integer division $k/2$, and $v[i]$ stands for the symbol of the word v at position i.

Theorem 4.2 *Let I be a nonempty subset of $\{1, \ldots, k\}$ of cardinality $\lfloor k/2 \rfloor + 1 + \lfloor (d + k \,\%\, 2)/2 \rfloor$. Then the language K^+ is $(\tau, H_{d,k})$-bond-free, where*

$$K = \{v \in \Sigma^k \mid \text{ if } i \in I \text{ then } v[i] \in \{A, C\}\}.$$

Proof. Let r be the quantity $\lfloor (d + k \,\%\, 2)/2 \rfloor$. One can verify that the condition $d < k$ is equivalent to $\lfloor k/2 \rfloor + 1 + r \leq k$. Now note that every word of length k of the form sp, with s being a suffix of K and p a prefix of K, contains at least $\lfloor k/2 \rfloor + 1 + r$ symbols in $\{A, C\}$. Hence, $\tau(sp)$ contains at least $\lfloor k/2 \rfloor + 1 + r$ symbols in $\{G, T\}$. Assume that K^+ is not $(\tau, H_{d,k})$-bond-free; then $H(s_1 p_1, \tau(s_2 p_2)) \leq d$ for some words $s_1 p_1$ and $s_2 p_2$ of the above form. This implies that $s_1 p_1$ contains at least $\lfloor k/2 \rfloor + 1 + r - d$ symbols in $\{T, G\}$. At the same time this word contains at least $\lfloor k/2 \rfloor + 1 + r$ symbols in $\{A, C\}$. Hence,

$$|s_1 p_1| \geq \lfloor k/2 \rfloor + 1 + r - d + \lfloor k/2 \rfloor + 1 + r = 2r + 2 + 2\lfloor k/2 \rfloor - d.$$

Using the facts that $k = 2\lfloor k/2 \rfloor + k \,\%\, 2$ and $2r + 2 > d + k \,\%\, 2$, it follows that $|s_1 p_1| > k$; a contradiction. Hence, K^+ must be $(\tau, H_{d,k})$-bond-free. □

Let l be the quantity $\lfloor k/2 \rfloor + 1 + \lfloor (d + k \,\%\, 2)/2 \rfloor$ that appears in the above theorem – recall from the proof of the theorem that $l \leq k$. The size of the code K is $2^l 4^{k-l}$. On the other hand the method of k-templates produces codes K of size 2^k. Obviously, $2^l 4^{k-l} \geq 2^k$. Moreover, one can verify that $k = l$ iff d is in $\{k-2, k-3\}$. An advantage of the method of Theorem 4.2 is that we can construct $(\tau, H_{d,k})$-bond-free languages with a large ratio d/k. On the other hand, in the case of $d = 0$ we have that $\lfloor (d + k \,\%\, 2)/2 \rfloor = 0$ and, therefore,

$$|K| = 2^{\lfloor k/2 \rfloor + 1} 4^{\lceil k/2 \rceil - 1}. \tag{13}$$

Another advantage of some codes K defined in the previous theorem is that one can encode and decode information in linear time. More specifically, consider the following instance of the code K

$$K = \Sigma^{k-l}\{A, C\}^l$$

such that l is even – this holds, for instance, if d and k are even and $k + d + 2$ is a multiple of 4. Let n be the quantity $k - l/2$. Every word $a_1 \cdots a_n$ in Σ^n can be encoded with a codeword in K as follows. Each symbol a_i is encoded as a_i, for $i = 1, \ldots, k - l$, and each symbol a_j is encoded as

$$\begin{cases} AA, & \text{if } a_j = A; \\ AC, & \text{if } a_j = C; \\ CA, & \text{if } a_j = G; \\ CC, & \text{if } a_j = T; \end{cases}$$

for $j = k - l + 1, \ldots, n$. Clearly, this process can be done in time $O(n)$. For example, if $k = 12$ and $d = 6$ then $l = 10$ and $n = 7$ and the word $AGTTCAG$ will be encoded as $AGCCCCACAACA$. On the other hand it is easy to see that each codeword $b_1 \cdots b_k$ in K can be decoded in time $O(k)$.

4.2 Methods Based on the Catenation Closure

The main idea here is that the catenation closure of Q^{d+1}, that is the language $(Q^{d+1})^+$, is $(\tau, H_{d,k})$-bond-free if Q is of length q with the property that Q^+ is $(\tau, H_{0,q})$-bond-free. The correctness of the method is based on the following theorem.

Theorem 4.3 *Let j and q be positive integers and let L be a subset of $\Sigma^{jq}\Sigma^*$. If L is $(\tau, H_{t,q})$-bond-free, for some integer $t \geq 0$, then it is also $(\tau, H_{d,k})$-bond-free, where $d = j(t + 1) - 1$ and $k = jq$.*

Proof. The claim is trivial for $j = 1$. So assume that $j \geq 2$ and consider any subwords w and w' of L of length $k = jq$. These can be written in the form $v_1 \cdots v_j$ and $v'_1 \cdots v'_j$, respectively, where each v_i and v'_i is of length q. For each index i, the words v_i and v'_{j-i+1} are subwords of L, which implies that $H(v_i, \tau(v'_{j-i+1})) > t$ and, therefore,

$$H(w, \tau(w')) = \sum_{i=1}^{j} H(v_i, \tau(v'_{j-i+1})) \geq j(t + 1).$$

Hence, L is $(\tau, H_{d,k})$-bond-free. □

Observe that for $t = 0$, the above theorem says that nearly every language that is $(\tau, H_{0,q})$-bond-free is *inherently* $(\tau, H_{d,k})$-bond-free for any $d > 0$ and any $k \geq q(d+1)$. This is a connection between the properties **P2** and **P4** considered in Section 1.

With the notation of the above theorem, let Q be a code of length q such that the language Q^+ is $(\tau, H_{t,q})$-bond-free. Let $K = Q^j$ and let $k = jq$. The code Q could be defined by some direct method, or by brute force for small values of q and t. In either case, the language K^+ is $(\tau, H_{d,k})$-bond-free.

In the case of $t = 0$, we have that $j = d + 1$ and the cardinality of the code K is $|Q|^{d+1}$, which can be larger than the cardinality of the codes defined in Theorem 4.2 with the same parameters. For example, consider the code

$$Q = \{A, C\}^2\Sigma \cup \{A, C\}\{G, T\}\{A, C\},$$

which consists of $2^2 \cdot 4 + 2^3 = 24$ codewords of length 3. Then the language Q^+ is $(\tau, H_{0,q})$-bond-free, where $q = 3$.[1] Moreover, according to the above, for any integer $d > 0$, the language $(Q^{d+1})^+$ is $(\tau, H_{d,k})$-bond-free, where $k = 3(d+1)$, and the code $K = Q^{d+1}$ consists of 24^{d+1} codewords. On the other hand, if the code K is defined using Theorem 4.2 for $k = 3(d+1)$) then the quantity $\lfloor k/2 \rfloor + 1 + \lfloor (d + k \,\%\, 2)/2 \rfloor$ is equal to $(k+d+1)/2 = 2(d+1)$, which implies that the cardinality of K is equal to 16^{d+1}.

The following observation can be viewed as a converse type of Theorem 4.3.

Theorem 4.4 *Let K be any set of words such that the language K^+ is $(\tau, H_{d,k})$-bond-free, for some integers $d \geq 0$ and $k \geq 1$. Then the language K^+ is also $(\tau, H_{0,k-d})$-bond-free.*

Proof. Let x and y be any two subwords of K^+ of length $k-d$. We need to show that $x \neq \tau(y)$. It is easy to see that one can always find subwords of K^+ of the form xu and vy whose length is equal to k. The assumption about K^+ implies that $H(xu, \tau(y)\tau(v)) > d$. Moreover, as $H(u, \tau(v) \leq d$, it follows that $H(x, \tau(y)) > 0$ as required. □

We close this section with a connection between the properties **P2** and **P4** considered in Section 1, which is analogous to the connection between **P1** and **P5**.

Theorem 4.5 *Let Q be any code of fixed length q such that Q^+ satisfies **P2**[q]. For any positive integer j, the language $(Q^j)^+$ satisfies **P4**[$j-1, jq$].*

Proof. The proof is similar to that of Theorem 4.3, for $t = 0$, and is left to the reader. □

4.3 All Maximal (Hamming) Bond-free Languages

With the results of Section 3 in mind, we understand that the languages of the form K^+ obtained by the preceding methods are not necessarily maximal. In what follows we discuss methods of obtaining new bond-free languages, possibly maximal, from old ones using the subword closure operation $\otimes$. We need the following, slightly restricted, version of the subword closure of S, where S is any code of fixed length k,

$$S^{\oplus} \triangleq S^{\otimes} \cap (\Sigma^k)^+.$$

We call $S^{\oplus}$ the *block closure* of S. Given a trie T accepting S, one can use a product construction between the DFA $T^{\otimes}$ of Lemma 3.10 and a DFA accepting $(\Sigma^k)^+$ to construct a DFA $T^{\oplus}$ of size $O(k|T|)$ accepting the block closure of S. Using Lemma 3.5, one can verify that

$$S_1 \subseteq S \text{ iff } {S_1}^{\oplus} \subseteq S^{\oplus},$$

for all subsets S and S_1 of Σ^k. This implies that if $S_1 \neq S$ then ${S_1}^{\oplus} \neq S^{\oplus}$.

Theorem 4.6 *Let S be a set of words of fixed length k. Then each of the languages $S^{\otimes}$ and $S^{\oplus}$ is $(\tau, H_{d,k})$-bond-free iff*

$$\tau(S) \cap H_d(S) = \emptyset. \tag{14}$$

[1] To see this, represent with 0 any element in $\{A, C\}$, with 1 any element in $\{G, T\}$, and with $*$ any element in Σ, and observe that (i) the representation of any subword of length 3 of Q^+ either contains two 0's or is equal to $*01$, and (ii) the representation of any subword of length 3 of $\tau(Q^+)$ either contains two 1's or is equal to $01*$.

Proof. The statement follows easily from (11) and the fact that $S = \mathrm{Sub}_k(S^\otimes) = \mathrm{Sub}_k(S^\oplus)$ – see Lemma 3.2. □

Using the above observation we can extend $(\tau, H_{d,k})$-bond-free languages of the form K^+, such as those constructed earlier, as follows – we assume the words of K are of fixed length k. Let $S = \mathrm{Sub}_k(K^2) = \mathrm{Sub}_k(K^+)$. Then S satisfies (14) and, therefore, the language $S^\otimes$ is a $(\tau, H_{d,k})$-bond-free language that includes K^+.

Next consider any code K of length k satisfying property **P3**$[d, k]$ – recall from Section 1 that such codes have been studied in [23] and [31]. Using again the above theorem it follows that $K^\otimes$ is a $(\tau, H_{d,k})$-bond-free language.

The question that arises now is when the bond-free languages of Theorem 4.6 are maximal. The following result addresses this question. In fact we show a complete characterization of all maximal $(\tau, H_{d,k})$-bond-free subsets of $(\Sigma^k)^+$ and $\Sigma^k\Sigma^*$.

Theorem 4.7 *The class of all maximal $(\tau, H_{d,k})$-bond-free subsets of $(\Sigma^k)^+$ is finite and equal to*

$$\{S^\oplus \mid S \subseteq \Sigma^k \text{ and } S \text{ is maximal satisfying } \tau(S) \cap H_d(S) = \emptyset\}.$$

In particular, if $d = 0$ then this class is equal to

$$\{S^\oplus \mid S \cup \tau(S) = \{v \in \Sigma^k \mid v \neq \tau(v)\},\ \ \tau(S) \cap S = \emptyset\}$$

and has cardinality

$$\begin{cases} 2^{4^k/2}, & \text{if } k \text{ is odd;} \\ 2^{(4^k - 4^{k/2})/2}, & \text{if } k \text{ is even.} \end{cases}$$

Proof. Let $D_{d,k}$ be the set $\{v \in \Sigma^k \mid H(v, \tau(v)) > d\}$. It is evident that if a subset S' of Σ^k satisfies (14) then $S' \subseteq D_{d,k}$. The main tool of this proof is the following claim.

- For any subset S of Σ^k that satisfies (14) we have that S is maximal with this property iff S satisfies
$$D_{d,k} \subseteq S \cup \tau(H_d(S)). \tag{15}$$

For the 'if' part of the claim assume that S satisfies (14) and (15), but suppose that there is a word v in $\Sigma^k - S$ such that the set $S \cup \{v\}$ satisfies (14) as well. As $v \in D_{d,k}$ and $v \notin S$ we have that $v \in \tau(H_d(S))$, or equivalently $\tau(v) \in H_d(S)$ which implies that $S \cup \{v\}$ does not satisfy (14); a contradiction. For the 'only if' part assume that S is maximal with the property (14), but suppose there is a word v in $D_{d,k}$ such that v is not in $S \cup \tau(H_d(S))$. One can verify that the set $S \cup \{v\}$ satisfies (14) as well, contradicting the maximality of S.

We now turn to the general statement of the theorem where $d \geq 0$. We need to prove the following two claims.

1. If a subset S of Σ^k satisfies (14) and (15) then $S^\oplus$ is a maximal $(\tau, H_{d,k})$-bond-free subset of $(\Sigma^k)^+$.

2. If L is any maximal $(\tau, H_{d,k})$-bond-free subset of $(\Sigma^k)^+$ then $L = S^\oplus$, where $S = \mathrm{Sub}_k(L)$, such that S satisfies (14) and (15).

For the first claim we note firstly that (14) and the fact $S = \mathrm{Sub}_k(S^\oplus)$ imply that $S^\oplus$ is a $(\tau, H_{d,k})$-bond-free subset of $(\Sigma^k)^+$. Now consider the sets

$$X = ((\Sigma^k)^+ - S^\oplus) \cap D_{d,k} \cap \tau(H_d(S))^c \ \text{ and } \ Y = ((\Sigma^k)^+ - S^\oplus) \cap S^\otimes.$$

It is easy to see that $X = D_{d,k} - (S \cup \tau(H_d(S)))$. As both of X and Y are empty when (15) holds, Lemma 3.6 implies that $S^{\oplus}$ is a maximal $(\tau, H_{d,k})$-bond-free subset of $(\Sigma^k)^+$, as required.

For the second claim, let L be any maximal $(\tau, H_{d,k})$-bond-free subset of $(\Sigma^k)^+$. Then S satisfies (14) and, by Lemma 3.6, S satisfies (15) as well. Moreover, L is a subset of $S^{\oplus}$ – see Lemma 3.5. It remains to show that $S^{\oplus}$ is a subset of L. For this, assume there is a word w in $S^{\oplus} - L$; then the set $((\Sigma^k)^+ - L) \cap S^{\otimes}$ is nonempty and Lemma 3.6 implies that L is not maximal. Hence, $S^{\oplus} = L$ as required.

It is evident that the class of all maximal $(\tau, H_{d,k})$-bond-free subsets of $(\Sigma^k)^+$ is a finite set. The statement for the case of $d = 0$ follows when we note that, for any subset S of Σ^k, we have that $H_d(S) = S$ and, if S satisfies (14), then $\tau(S) \cup S \subseteq D_{0,k}$. Moreover, the claim about the cardinality of the class follows easily when we note that the partition $\{\{v, \tau(v)\} \mid v \in D_{0,k}\}$ of $D_{0,k}$ contains $|D_{0,k}|/2$ different subsets and that exactly one of the two elements of each subset can be included in a particular S. We also recall that the cardinality of $D_{0,k}$ is 4^k if k is odd, or $4^k - 4^{k/2}$ if k is even [3]. □

Note: *The above theorem holds also for subsets of $\Sigma^k\Sigma^*$ if we replace $S^{\oplus}$ with $S^{\otimes}$.*

According to Theorem 4.7, if K is a maximal subset of Σ^k satisfying $\tau(K) \cap H_d(K)$ then the language $K^{\oplus}$ is a maximal $(\tau, H_{d,k})$-bond-free subset of $(\Sigma^k)^+$. In the case of $d = 0$ the characterization of the maximal bond-free languages is quite explicit: Define any partition $\{S, \tau(S)\}$ of the set $\{v \in \Sigma^k \mid v \neq \tau(v)\}$ and then compute $S^{\oplus}$; this language will be maximal. Note that the language L_2 considered in Example 3.7 is a particular instance of this construction.

The above theorem implies that every k-block $(\tau, H_{d,k})$-bond-free language L is included in a *regular* maximal such language. Statements of this type with L being regular have been obtained for various code-related properties and are of particular interest in the theory of codes [5], [14]. In our case it is also interesting to note that the language L is not necessarily regular.

Corollary 4.8 *Let M be one of $(\Sigma^k)^+$ or $\Sigma^k\Sigma^*$. Every $(\tau, H_{d,k})$-bond-free subset of M is included in a regular maximal $(\tau, H_{d,k})$-bond-free subset of M.*

5 Discussion

We have considered the problem of undesirable bonds and proposed the property of (θ, sim)-bond-freedom for DNA languages, which addresses this problem when bonds between imperfect complements of DNA molecules are permitted. Using recent language theoretic tools, we were able to establish various decidability results about (θ, sim)-bond-freedom. The case where sim is the Hamming similarity has been considered by many authors. In this case, we have demonstrated interesting connections between our property and those of other authors, and have identified general construction *methods*. In particular, we have identified all DNA languages that are maximal with respect to the new property. This result is also applicable to the case of the θ-k-code property of [12].

Directions for future research include the following.

- Derive a methodology for defining properties of DNA languages that would be able to address the uniqueness problem – called positive design problem in [25] – as independently of the application as possible.

- Elaborate on the proposed construction methods to obtain concrete constructions of languages that, in addition to being bond-free, they satisfy additional properties such as uniqueness and fixed GC-ratio.

- Explore further the subword closure operation from a theoretical at least point of view.

References

[1] M. Andronescu, D. Dees, L. Slaybaugh, Y. Zhao, A. Condon, B. Cohen, S. Skiena, Algorithms for testing that sets of DNA words concatenate without secondary structure. In [8], 182-195.

[2] M. Arita, S. Kobayashi, DNA sequence design using templates. *New Generation Computing* **20** (2002), 263–277.

[3] E. Baum, DNA sequences useful for computation. *Proc. 2nd DIMACS Workshop on DNA-based computers*, 1996, 122–127.

[4] M. Domaratzki, *Deletion Along Trajectories.* Tech. report 464-2003, School of Computing, Queen's University, 2003, and submitted for publication.

[5] A. Ehrenfeucht, G. Rozenberg. Each regular code is included in a maximal regular code. *RAIRO Inform. Ther. Appl.* **20** (1985), 89–96.

[6] U. Feldkamp, S. Saghafi, H. Rauhe. *DNASequenceGenerator - A program for the construction of DNA sequences.* In [13], 179–189.

[7] M. Garzon, P. Neathery, R. Deaton, R. C. Murphy, D. R. Franceschetti, S. E. Stevens Jr., A new metric for DNA computing. In: J. R. Koza, K. Deb, M. Dorigo, D. B. Vogel, M. Garzon, H. Iba, R. L. Riolo, (eds) *Proc. 2nd Annual Genetic Programming Conference, Stanford U.*, 1997, 472–478.

[8] M. Hagiya, A. Ohuchi (eds), *Proc. 8th Workshop on DNA-Based Computers, Saporo, Japan, 2002. LNCS* **2568**, 2002.

[9] T. Head, Relativised code concepts and multi-tube DNA dictionaries. In C.S. Calude, G. Păun, *Finite Versus Infinite: Contributions to an Eternal Dilemma*, Springer-Verlag, London, 2000, 175–186.

[10] J.E. Hopcroft, R. Motwani, J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Boston, 2001.

[11] S. Hussini, L. Kari, S. Konstantinidis, Coding properties of DNA languages. In [13], 107-118.

[12] N. Jonoska, K. Mahalingam, Languages of DNA based code words. In J. Chen, J. Reif (eds.), Preproceedings of DNA9, June 1–4, 2003, Madison, Wisconsin, pp. 58–68.

[13] N. Jonoska, N. C. Seeman (eds), *Proc. 7th Workshop on DNA-Based Computers, Tampa, Florida, 2001. LNCS* **2340**, 2002.

[14] H. Jürgensen, S. Konstantinidis. Codes. In [28], 511–607.

[15] L. Kari, On insertion and deletion in formal languages, *PhD thesis*, University of Turku, Finland, 1991.

[16] L. Kari, On language equations with invertible operations, *Theoretical Computer Science* **132** (1994), 129–150.

[17] L. Kari, R. Kitto, G. Thierrin, Codes, involutions and DNA encoding. In W. Brauer, H. Ehrig, J. Karhumäki, A. Salomaa (eds.), *Lecture Notes in Computer Science* **2300** (2002), 376–393.

[18] L. Kari, S. Konstantinidis, Language equations, maximality and error detection. Submitted for publication.

[19] L. Kari, S. Konstantinidis, E. Losseva, G. Wozniak, Sticky-free and overhang-free DNA languages. *Acta Informatica* **40** (2003), 119–157.

[20] L. Kari, S. Konstantinidis, P. Sosík, On properties of bond-free DNA languages. *Tech. report 609, Univ. Western Ontario, Dept. Computer Science*, 2003.

[21] L. Kari, P. Sosík, *Language deletion on trajectories.* Dept. of Computer Science technical report No. 606, University of Western Ontario, London, 2003.

[22] S. Konstantinidis, Transducers and the properties of error detection, error correction and finite-delay decodability. *J. Universal Comp. Science* **8** (2002), 278–291.

[23] A. Marathe, A.E. Condon, R.M. Corn, On combinatorial DNA word design. *J. Computational Biology*, **8**:3, 2001.

[24] A. Mateescu, G. Rozenberg, A. Salomaa, Shuffle on trajectories: syntactic constraints, TUCS technical report No. 41, Turku Centre for Computer Science, 1996, and *Theoretical Computer Science* **197** (1998), 1–56.

[25] G. Mauri, C. Ferretti, *Word Design for Molecular Computing: A Survey.* Invited lecture at DNA9, June 1–4, 2003, Madison, Wisconsin, U.S.

[26] G. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms* (Springer-Verlag, Berlin, 1998).

[27] J. H. Reif, T. H. LaBean, M. Pirrung, V. S. Rana, B. Guo, C. Kingsford, G. S. Wickham, Experimental construction of very large scale DNA databases with associative search capability. In [13], 231–247.

[28] G. Rozenberg, A. Salomaa (eds). *Handbook of Formal Languages, vol. I.* Berlin: Springer 1997.

[29] A. Salomaa, *Formal Languages,* Academic press, London, 1973.

[30] F. Tanaka, M. Nakatsugawa, M. Yamamoto, T. Shiba, A. Ohuchi, Developing support system for sequence design in DNA computing. In [13], 129–137.

[31] D. C. Tulpan, H. H. Hoos, A. E. Condon, Stochastic Local Search Algorithms for DNA Word Design. In [8], 229–241.

[32] D. Wood. *Theory of Computation,* Harper & Row, New York, 1987.

[33] S. Yu, Regular languages. In [28], pp. 41–110.