

Preventing Undesirable Bonds Between DNA Codewords

Lila Kari¹, Stavros Konstantinidis², and 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 Dept. 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 input data for DNA computing must be encoded into the form of single or double DNA strands. As complementary parts of single strands can bind together forming a double-stranded DNA sequence, one has to impose restrictions on these sets of DNA words (=languages) to prevent them from interacting in undesirable ways. We recall a list of known properties of DNA languages which are free of certain types of undesirable bonds. Then we introduce a general framework in which we can characterize each of these properties by a solution of a uniform formal language inequation. This characterization allows us among others to construct (i) a uniform algorithm deciding in polynomial time whether a given DNA language possesses any of the studied properties, and (ii) in many cases also an algorithm deciding whether a given DNA language is maximal with respect to the desired property.

1 Introduction

A principle of DNA computing, in a nutshell, is to encode a task into a set of input DNA molecules so that their mutual (and highly parallel) reactions serve as a computational process, and produce a set of output molecules representing a *result* of the computing. Fundamental techniques are the operations of *hybridization (annealing)* and *denaturation (melting)* [16]. Given this framework, [14] and others distinguish two elementary subproblems of the design of sets of molecules for DNA experiments and computing:

- *Positive* design problem: to construct an input set of DNA molecules such that there exists a sequence of reactions to produce a desired final set – the result of the experiment.

* Corresponding author.

- *Negative* design problem: the input set of molecules must not give way to the reactions that produce undesired molecules encoding a false result or blocking the desired reactions.

The negative design problem can be solved on a general basis by construction of a large set of molecules which do not allow undesired mutual reactions. Various subsets of this set are then used for concrete experiments. See e.g. [1, 3, 4, 5, 6, 8, 10, 13] for studies of properties of such sets (called also DNA languages) and methods of their construction.

In this paper we introduce the concept of general (*strictly*) *bond-free DNA language property*, and show that many of the previously studied properties are its special cases. Moreover, we construct a general quadratic-time algorithm deciding whether a given regular set of codewords satisfies any of these special cases of the bond-free property. We note that by the term *algorithm* we always mean a *deterministic* procedure, even if its input might be a nondeterministic formal automaton.

By utilizing and improving recent results in [9] on language inequations, we furthermore show that for many of the bond-free properties the *maximality* problem is decidable. Polynomial-time algorithms are presented for the case of θ -compliant finite sets and θ -non-overlapping regular sets. The proofs of these results are mostly nontrivial and due to page limitations can be found in [11].

2 Undesirable Bonds in DNA Languages

We represent the single-stranded DNA molecules by strings over the *DNA alphabet* $\Delta = \{A, C, T, G\}$. Generally, an *alphabet* Σ is a finite and nonempty set of symbols. The set of all words (over Σ) is denoted by Σ^* , including the *empty word* λ . The length of a word w is denoted by $|w|$. For an $n \geq 0$, w^n denotes the n concatenated copies of w . We denote the mirror image of the word w by w^R .

A language L is a set of words, i.e. a subset of Σ^* . For $n \geq 0$, we denote by L^n the set of all words $w_1 \cdots w_n$ such that each w_i is in L . We also write $L^* = L^0 \cup L^1 \cup L^2 \cup \cdots$, and L^+ for $L^* - \{\lambda\}$. The complement of L is $L^c = \Sigma^* - L$. The mirror image of L is $L^R = \{w^R \mid w \in L\}$.

A nondeterministic finite automaton (*NFA*) 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 $sx \rightarrow t$, for $s, t \in S$, $x \in \Sigma$. If for every two productions $sx_1 \rightarrow t_1$ and $sx_2 \rightarrow t_2$ of an NFA we have that $x_1 \neq x_2$ then the automaton is called a *DFA* (deterministic finite automaton). The language accepted by the automaton A is denoted by $L(A)$. The *size* $|A|$ of the automaton A is the number $|S| + |P|$.

A mapping $\alpha : \Sigma^* \rightarrow \Sigma^*$ is called a *morphism* (*anti-morphism*) of Σ^* if $\alpha(uv) = \alpha(u)\alpha(v)$ (respectively $\alpha(uv) = \alpha(v)\alpha(u)$) for all $u, v \in \Sigma^*$.

An *involution* $\theta : \Sigma \rightarrow \Sigma$ of Σ is a mapping such that $\theta(\theta(x)) = x$ for all $x \in \Sigma$. It follows then that an involution θ is bijective and $\theta = \theta^{-1}$. An involution of Σ can be extended to either a morphism or an antimorphism of

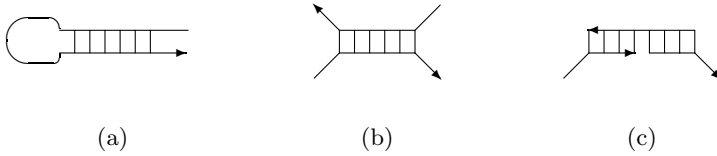


Fig. 1. Types of intramolecular (a) and intermolecular (b), (c) hybridizations

Σ^* . For example, if we extend the identity of Σ to an antimorphism of Σ^* we obtain the mirror-image involution of Σ^* that maps each word u into u^R .

If we consider the DNA-alphabet Δ , then the mapping $\tau : \Delta \rightarrow \Delta$ defined by $\tau(A) = T, \tau(T) = A, \tau(C) = G, \tau(G) = C$ can be extended in the usual way to an antimorphism of Δ^* . Then single strands $w_1, w_2 \in \Delta^*$ are complementary iff $w_1 = \tau(w_2)$. We refer the reader to [17] for further details on automata and formal languages.

Two types of unwanted hybridization are usually considered: *intramolecular*, Fig. 1 (a), and *intermolecular*, (b) and (c). The following properties of a DNA language $L \subseteq \Sigma^+$ preventing such a hybridization have been defined in [4, 8, 10].

- (A) **θ -non-overlapping:** $L \cap \theta(L) = \emptyset$.
- (B) **θ -compliant:** $\forall w \in L, x, y \in \Sigma^*, w, x\theta(w)y \in L \Rightarrow xy = \lambda$.
- (C) **θ - p -compliant:** $\forall w \in L, y \in \Sigma^*, w, \theta(w)y \in L \Rightarrow y = \lambda$.
- (D) **θ - s -compliant:** $\forall w \in L, y \in \Sigma^*, w, y\theta(w) \in L \Rightarrow y = \lambda$.
- (E) **strictly θ -compliant:** both θ -compliant and θ -non-overlapping.
- (F) **θ -free:** $L^2 \cap \Sigma^+\theta(L)\Sigma^+ = \emptyset$.
- (G) **θ -sticky-free:** $\forall w \in \Sigma^+, x, y \in \Sigma^*, wx, y\theta(w) \in L \Rightarrow xy = \lambda$.
- (H) **θ -3'-overhang-free:** $\forall w \in \Sigma^+, x, y \in \Sigma^*, wx, \theta(w)y \in L \Rightarrow xy = \lambda$.
- (I) **θ -5'-overhang-free:** $\forall w \in \Sigma^+, x, y \in \Sigma^*, xw, y\theta(w) \in L \Rightarrow xy = \lambda$.
- (J) **θ -overhang-free:** both θ -3'-overhang-free and θ -5'-overhang-free.

We agree to say that a language L containing the empty word has one of the above properties if $L \setminus \{\lambda\}$ has that property. Observe that (F) corresponds to Fig. 1 (c), while others are special cases of (b).

In [6], a θ -non-overlapping language is called to be *strictly θ* . Generally, if any other property holds in conjunction with (A), we add the qualifier *strictly*. Further properties have been defined in [6]. We denote by $\text{Sub}_k(L)$ the set of all subwords of L of the length k . The following property corresponds to Fig. 1 (a):

- (K) **$\theta(k, m_1, m_2)$ -subword compliant:** $\forall u \in \Sigma^k, \Sigma^*u\Sigma^{m_1}\theta(u)\Sigma^{m_2} \cap L = \emptyset$ for $k > 0, m_1 \leq m \leq m_2$.
- (L) **θ - k -code:** $\text{Sub}_k(L) \cap \text{Sub}_k(\theta(L)) = \emptyset, k > 0$.

The following property is defined for $\theta = I$, the identity relation, in [3]. A language L is called

- (M) **solid** if:
1. $\forall x, y, u \in \Sigma^*, u, xuy \in L \Rightarrow xy = \lambda$, and
 2. $\forall x, y \in \Sigma^*, u \in \Sigma^+, xu, uy \in L \Rightarrow xy = \lambda$.

L is *solid relative to an* $M \subseteq \Sigma^*$ if 1. and 2. above hold only for $w = pxuyq \in M$. L is called *comma-free* if it is solid relative to L^* . Solid languages are also used in [10] as a tool for constructing error-detecting DNA languages that are invariant under bio-operations. We refer to [6, 10, 11] for further examples and for mutual relations between classes of a DNA languages satisfying these properties.

3 Binary Word Operations

Binary word operations on are extensively used in the following sections for representing interaction of DNA molecules. A *binary word operation* is a mapping $\diamond : \Sigma^* \times \Sigma^* \rightarrow 2^{\Sigma^*}$, where 2^{Σ^*} is the set of all subsets of Σ^* . The notion can be extended to languages X and Y as follows:

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

The left and the right inverse \diamond^l and \diamond^r of \diamond , respectively, are defined as

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

Examples of binary word operations are catenation, quotient, insertion, deletion, shuffle etc. See [7, 9, 15] for more details.

Further we introduce word operations on trajectories [2, 12, 15]. 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 an operation \diamond 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 .

Definition 1. *The shuffle of α with β on the trajectory t , denoted by $\alpha \sqcup_t \beta$, is defined as follows:*

$$\alpha \sqcup_t \beta = \{ \alpha_1 \beta_1 \dots \alpha_k \beta_k \mid \alpha = \alpha_1 \dots \alpha_k, \beta = \beta_1 \dots \beta_k, t = 0^{i_1} 1^{j_1} \dots 0^{i_k} 1^{j_k}, \text{ where } |\alpha_m| = i_m \text{ and } |\beta_m| = j_m \text{ for all } m, 1 \leq m \leq k \}.$$

Example 2. Let $\alpha = a_1 a_2 \dots a_8, \beta = b_1 b_2 \dots b_5, t = 0^3 1^2 0^3 1 0 1 0 1$. The shuffle of α and β on the trajectory t is: $\alpha \sqcup_t \beta = \{ a_1 a_2 a_3 b_1 b_2 a_4 a_5 a_6 b_3 a_7 b_4 a_8 b_5 \}$.

Definition 3. *The deletion of β from α on the trajectory t is the following binary word operation:*

$$\alpha \rightsquigarrow_t \beta = \{ \alpha_1 \dots \alpha_k \mid \alpha = \alpha_1 \beta_1 \dots \alpha_k \beta_k, \beta = \beta_1 \dots \beta_k, t = 0^{i_1} 1^{j_1} \dots 0^{i_k} 1^{j_k}, \text{ where } |\alpha_m| = i_m \text{ and } |\beta_m| = j_m \text{ for all } m, 1 \leq m \leq k \}.$$

Example 4. Let $\alpha = babaab$, $\beta = bb$ and assume that $t = 001001$. The deletion of β from α on the trajectory t is: $\alpha \rightsquigarrow_t \beta = \{baaa\}$.

The operations can be extended to sets of trajectories as follows: $\alpha \diamond_T \beta = \bigcup_{t \in T} \alpha \diamond_t \beta$, where \diamond stands for \sqcup or \rightsquigarrow , respectively. The operations \sqcup_T and \rightsquigarrow_T generalize further to languages due to (1).

4 Bond-Free DNA Languages

The notion of DNA language property from Section 2 can be formalized as follows: a property \mathcal{P} is a mapping $\mathcal{P} : 2^{\Sigma^*} \rightarrow \{\text{true}, \text{false}\}$. We say that a language $L \subseteq \Sigma^*$ has (or satisfies) the property \mathcal{P} if $\mathcal{P}(L) = \text{true}$.

Definition 5. A language property \mathcal{P} is called a bond-free property of degree 2 if there exist binary word operations \diamond_{lo} , \diamond_{up} such that for an arbitrary $L \subseteq \Sigma^*$, $\mathcal{P}(L) = \text{true}$ iff







$$\forall w \in \Sigma^+, x, y \in \Sigma^*, (w \diamond_{\text{lo}} x \cap L \neq \emptyset, w \diamond_{\text{up}} y \cap \theta(L) \neq \emptyset) \Rightarrow xy = \lambda. \quad (2)$$

The phrase *degree 2* is used to stress the fact that the property describes bonds of two single DNA strands. In the remainder of this paper we write simply *bond-free property* for a bond-free property of degree two.

Furthermore, in this and the following section we assume that $\diamond_{\text{lo}} = \sqcup_{T_{\text{lo}}}$ and $\diamond_{\text{up}} = \sqcup_{T_{\text{up}}}$ for some trajectory sets $T_{\text{lo}}, T_{\text{up}} \subseteq V^*$.

Theorem 6. The language properties (B), (C), (D), (G), (H), (I), (M.1), (M.2) are bond-free properties. Moreover, the associated sets of trajectories $T_{\text{lo}}, T_{\text{up}}$ are regular.

Proof. Assume that θ is an antimorphism and define the sets of trajectories $T_{\text{lo}}, T_{\text{up}}$ as follows.

- (B) θ -compliant: $T_{\text{lo}} = 0^+$, $T_{\text{up}} = 1^*0^+1^*$. 
- (C) θ -p-compliant: $T_{\text{lo}} = 0^+$, $T_{\text{up}} = 1^*0^+$. 
- (D) θ -s-compliant: $T_{\text{lo}} = 0^+$, $T_{\text{up}} = 0^+1^*$. 
- (G) θ -sticky-free: $T_{\text{lo}} = 0^+1^*$, $T_{\text{up}} = 0^+1^*$. 
- (H) θ -3'-overhang-free: $T_{\text{lo}} = 0^+1^*$, $T_{\text{up}} = 1^*0^+$. 
- (I) θ -5'-overhang-free: $T_{\text{lo}} = 1^*0^+$, $T_{\text{up}} = 0^+1^*$. 

Consider e.g. the property (H), θ -3'-overhang-freedom. Then $w \sqcup_{T_{\text{lo}}} x = \{wx\}$ and $w \sqcup_{T_{\text{up}}} y = \{yw\}$. The relations in (2) adopt the form $wx \in L$, $yw \in \theta(L)$. This is equivalent to $wx \in L$, $\theta(w)\theta(y) \in L$. As $xy = \lambda$ iff $x\theta(y) = \lambda$,

(2) corresponds to the definition of (H) in Section 2. The proofs of the other mentioned properties are analogous.

If θ is a morphism, then all the sets of trajectories T_{up} must be replaced by the reversed sets T_{up}^R , the proof technique remaining unchanged. \square

Observe that $T_{\text{lo}}, T_{\text{up}}$ for a certain property corresponds to the “shape” of the bonds prohibited in languages satisfying the property, see attached figures.

The main reason for introducing Definition 5 is the characterization of bond-free properties via language inequations.

Theorem 7. *For each bond-free property \mathcal{P} there are regular sets of trajectories T_1, T_2 and a binary word operation $\Xi_{\mathcal{P}}$ defined as*

$$x \Xi_{\mathcal{P}} y = ((x \sqcup_{(01)^*} T_{\text{lo}}) \sqcup_{T_1} (\theta(y) \sqcup_{(01)^*} T_{\text{up}})) \rightsquigarrow_{T_2} K_1, \quad (3)$$

such that $\mathcal{P}(L) = \text{true}$ for an $L \subseteq \Sigma^*$ iff $L \Xi_{\mathcal{P}} L \subseteq K_2$.

This characterization allows us to answer decidability questions “Is $\mathcal{P}(L) = \text{true}$ for a given language L and a bond-free property \mathcal{P} ?”

Theorem 8. *Let \mathcal{P} be a bond-free property associated with regular sets of trajectories $T_{\text{lo}}, T_{\text{up}}$. The following problem is decidable in quadratic time w.r.t $|A|$:*

Input: an NFA A .

Output: Y/N depending on whether $L(A)$ satisfies \mathcal{P} .

Proof. Can be found in [11] using known results about word operations on trajectories in [2, 12, 15]. \square

In [4] the decidability of the properties (D) and (F) for regular sets of words was shown. In [3] the decidability of (M) in quadratic time is proven. In [5] an algorithm deciding (F) in quadratic time for finite sets of codewords is presented. The following corollary extends and generalizes these previous results for the case of regular sets of DNA codewords.

Corollary 9. *The following problem is decidable in quadratic time w.r.t. $|A|$:*

Input: an NFA A .

Output: Y/N depending on whether $L(A)$ satisfies any of the properties (B), (C), (D), (G), (H), (I), (J) (M).

On the other hand, it is known [4] that for some bond-free properties, e.g. (B) and (F), there is no such algorithm in the case of context-free DNA languages.

5 Maximal Bond-Free Languages

The approach used in the previous section can be applied also to maximality problems (“Is $L \subseteq M$ maximal w.r.t. a bond-free property \mathcal{P} ?”). The symbol $M \subseteq \Sigma^*$ represents the set of all applicable/constructible DNA strands in a case at hand. The following theorem is based on nontrivial results concerning language inequations in [9].

Theorem 10. *Let \mathcal{P} be a bond-free property and $M \subseteq \Sigma^+$ a set of words. For a language $L \subseteq M$ satisfying \mathcal{P} , denote*

$$R = M - (L \cup L \boxminus_{\mathcal{P}}^r K_2^c \cup K_2^c \boxminus_{\mathcal{P}}^l L), \tag{4}$$

$$Q = \{z \in \Sigma^* \mid z \boxminus_{\mathcal{P}} z \cap K_2^c \neq \emptyset\}, \tag{5}$$

where $\boxminus_{\mathcal{P}}$ is defined by (3) and $K_2 = (\Sigma \cup V)^*0(\Sigma \cup V)^* \cup \{\lambda\}$. Then L is a maximal subset of M satisfying \mathcal{P} iff $R - Q = \emptyset$.

After calculating the inverses of the operation $\boxminus_{\mathcal{P}}$, we obtain a decision algorithm concerning maximal bond-free languages.

Theorem 11. *Consider a fixed involution θ . Let \mathcal{P} be one of the properties (B), (C), (D), (G) if θ is an antimorphism, and one of (B), (C), (D), (H), (I) if θ is a morphism.*

Let $M \subseteq \Sigma^+$ be a regular set of words, and $L \subseteq M$ a regular language satisfying \mathcal{P} . Then there is an algorithm deciding whether L is a maximal subset of M satisfying \mathcal{P} .

Let A be a DFA accepting L . The algorithms described in the above theorem may require an exponential number of steps w.r.t. $|A|$ in the worst case. But one can obtain a polynomial-time algorithm at least for finite languages.

Theorem 12. *The following problem is decidable in time $O(\|L\|^3|A|)$, where $\|L\|$ is the quantity $\sum_{w \in L} |w|$.*

Input: DFA A and a finite language L such that $L \subseteq L(A)$ and L satisfies the property (B).

Output: Y/N, depending on whether L is a maximal subset of $L(A)$ satisfying (B).

The language inequation approach can be used also for characterization of supersets of non-maximal bond-free DNA languages without breaking the given bond-free property, see [11] for details. However, to construct such a superset may require an exponential number of steps w.r.t. $|A|$.

6 Strictly Bond-Free Languages

In this section we focus mostly on the strict versions of the DNA language properties (B)–(L), i.e. their conjunctions with (A). As one can easily observe, the property (E) is equal to strictly (B), hence we do not refer to (E) in the sequel. The following concept of a *strictly bond-free* property generalizes these properties. However, (non-strictly) (L) is also a special case of the strictly bond-free property.

Definition 13. *A language property \mathcal{P} is called the strictly bond-free property of degree 2 if there are binary word operations \diamond_{lo} , \diamond_{up} and an involution θ such that for an arbitrary $L \subseteq \Sigma^*$, $\mathcal{P}(L) = \text{true}$ iff*

$$\forall w, x, y \in \Sigma^* (w \diamond_{\text{lo}} x \cap L \neq \emptyset, w \diamond_{\text{up}} y \cap \theta(L) \neq \emptyset) \Rightarrow w = \lambda. \tag{6}$$

Again, in the remainder of this paper we write simply *strictly bond-free property* for the strictly bond-free property of degree two.

Theorem 14. *The language properties (A), strictly (B)–(D), strictly (G)–(I), (L), strictly (L) are strictly bond-free properties.*

Proof. Let $\diamond_{\text{lo}} = \sqcup\sqcup_{T_{\text{lo}}}$ and $\diamond_{\text{up}} = \sqcup\sqcup_{T_{\text{up}}}$, where T_{lo} and T_{up} are the sets of trajectories used in the proof of Theorem 6. The rest of the proof relies on similar techniques. \square

Similarly as in Theorem 7, one can characterize strictly bond-free properties via language (in)equations.

Theorem 15. *For each strictly bond-free property \mathcal{P} there is a binary word operation $\square_{\mathcal{P}}$ defined as*

$$x \square_{\mathcal{P}} y = (x \diamond_{\text{lo}}^l \Sigma^*) \rightsquigarrow_{1+} (\theta(y) \diamond_{\text{up}}^l \Sigma^*), \quad (7)$$

such that for a language $L \subseteq \Sigma^$, $\mathcal{P}(L) = \text{true}$ iff $L \square_{\mathcal{P}} L = \emptyset$.*

As a consequence one can derive a quadratic time decision algorithm for regular DNA languages.

Theorem 16. *Let \mathcal{P} be a strictly bond-free property associated with operations $\diamond_{\text{lo}} = \sqcup\sqcup_{T_{\text{lo}}}$, $\diamond_{\text{up}} = \sqcup\sqcup_{T_{\text{up}}}$, with regular sets of trajectories $T_{\text{lo}}, T_{\text{up}}$. Then the following problem is decidable in quadratic time w.r.t. $|A|$:*

Input: an NFA A .

Output: Y/N depending on whether $L(A)$ satisfies \mathcal{P} .

Corollary 17. *Let \mathcal{P} be any of the properties (A), strictly (B) – strictly (D), strictly (G) – strictly (J), (L), strictly (L). The following problem is decidable in quadratic time w.r.t. $|A|$:*

Input: an NFA A .

Output: Y/N depending on whether $L(A)$ satisfies \mathcal{P} .

On the other hand, one can easily show [11] that e.g. for the property (A), there is no decision algorithm in the context-free case.

7 Maximal Strictly Bond-Free Languages

We focus on maximality problems (see section 5) w.r.t. a strictly bond-free property \mathcal{P} . For the case of the θ -non-overlapping regular languages, the problem is decidable in polynomial time, for some other properties the polynomial-time algorithm for regular languages is not known.

Theorem 18. *The following problem is decidable in time $O((|A| \cdot |A_{\theta}| \cdot |A_M|)^3)$.*

Input: DFA's A , A_{θ} and an NFA A_M such that $L(A) = \theta(L(A_{\theta})) \subseteq L(A_M)$ and $L(A)$ is θ -non-overlapping.

Output: Y/N, depending on whether $L(A)$ is a maximal θ -non-overlapping subset of $L(A_M)$.

Theorem 19. Consider a fixed involution θ . Let \mathcal{P} be any of the properties strictly (B) – strictly (D), strictly (G), (L), strictly (L) if θ is an antimorphism. Let \mathcal{P} be any of strictly (B) – strictly (D), strictly (H), strictly (I), (L), strictly (L) if θ is a morphism.

Let $M \subseteq \Sigma^+$ be a regular set of words and $L \subseteq M$ a regular language satisfying \mathcal{P} . Then there is an algorithm deciding whether L is a maximal subset of M satisfying \mathcal{P} .

Similarly as in Section 5, supersets of non-maximal languages satisfying a certain strictly bond-free property can be also characterized.

8 Summary

We proposed a sequence of algorithms solving decision problems of DNA languages without undesirable bonds. The results are summarized in Tables 1 and 2. The abbreviations *REG* and *CF* denote the classes of regular and context-free languages, respectively. In the column θ , the symbol A denotes antimorphism and M denotes morphism, * stands for an arbitrary involution. In the columns corresponding to particular properties (B)–(M), D stands for decidable, Q for quadratic-time decidable, P for polynomial-time decidable, U for undecidable and ? for an open problem.

Furthermore we presented a polynomial-time algorithm deciding maximality of a *finite* DNA language w.r.t. the property (B).

Among major open questions we mention the study of fast algorithms for construction of finite bond-free languages, methods preventing imperfect bonds (with bulges, non-complementary pairs etc.) between DNA strands, and study of influence of the secondary DNA structure and free energy of single strands.

Table 1. Decision problems of non-strict DNA language properties

Problem	Class	θ	Properties									
			(B)	(C)	(D)	(G)	(H)	(I)	(J)	(L)	(M)	
Does a given language satisfy the property \mathcal{P} ?	<i>REG</i>	*	Q	Q	Q	Q	Q	Q	Q	Q	Q	
	<i>CF</i>	*	U	?	?	?	?	?	?	?	?	
Is a given language maximal w.r.t. \mathcal{P} ?	<i>REG</i>	A	D	D	D	D	?	?	?	D	-	
	<i>REG</i>	M	D	D	D	?	D	D	?	D	?	

Table 2. Decision problems of strict DNA language properties

Problem	Class	θ	Properties									
			(A)	(B)	(C)	(D)	(G)	(H)	(I)	(J)	(L)	
Does a given language satisfy the property \mathcal{P} ?	<i>REG</i>	*	Q	Q	Q	Q	Q	Q	Q	Q	Q	
	<i>CF</i>	*	U	?	?	?	?	?	?	?	?	
Is a given language maximal w.r.t. \mathcal{P} ?	<i>REG</i>	A	P	D	D	D	D	?	?	?	D	
	<i>REG</i>	M	P	D	D	D	?	D	D	?	D	

Acknowledgements

Research was partially supported by the Canada Research Chair Grant to L.K., NSERC Discovery Grants R2824A01 to L.K. and R220259 to S.K., and by the Grant Agency of Czech Republic, Grant 201/02/P079 to P.S.

References

1. M. Arita, S. Kobayashi, DNA sequence design using templates. *New Generation Computing* **20** (2002), 263–277.
2. M. Domaratzki, *Deletion Along Trajectories*. Tech. Report 464-2003, School of Computing, Queen’s University, 2003, and submitted for publication.
3. 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.
4. S. Hussini, L. Kari, S. Konstantinidis, Coding properties of DNA languages. *theoretical Computer Science* **290/3** (2002), 1557-1579.
5. N. Jonoska, D. Kephart, K. Mahalingam, Generating DNA code words. *Congressus Numerantium* **156** (2002), 99–110.
6. 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.
7. L. Kari, On insertion and deletion in formal languages, *PhD thesis*, University of Turku, Finland, 1991.
8. 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.
9. L. Kari, S. Konstantinidis, Language equations, maximality and error detection. Submitted for publication.
10. L. Kari, S. Konstantinidis, E. Losseva, G. Wozniak, Sticky-free and overhang-free DNA languages. *Acta Informatica* **40** (2003), 119–157.
11. L. Kari, S. Konstantinidis, P. Sosík, *On Properties of Bond-Free DNA Languages*. Dept. of Computer Science Tech. Report No. 609, Univ. of Western Ontario, 2003, and submitted for publication.
12. L. Kari, P. Sosík, *Language deletion on trajectories*. Dept. of Computer Science Technical Report No. 606, University of Western Ontario, London, 2003.
13. A. Marathe, A.E. Condon, R.M. Corn, On combinatorial DNA words design. *J. Computational Biology*, **8**:3, 2001.
14. G. Mauri, C. Ferretti, Word Design for Molecular Computing: A Survey. In J. Chen and J.H. Reif (Eds.), *DNA Computing, 9th International Workshop on DNA Based Computers, Lecture Notes in Computer Science* **2943** (2004), 37–46.
15. 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.
16. G. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, Springer-Verlag, Berlin, 1998.
17. G. Rozenberg, A. Salomaa (Eds.), *Handbook of Formal Languages*, Springer-Verlag, Berlin, 1997.