Hairpin Structures in DNA Words

Lila Kari[1], Stavros Konstantinidis[2], Elena Losseva[1],
Petr Sosík[3,4,⋆], and Gabriel Thierrin[5]

[1] Department of Computer Science,
The University of Western Ontario, London, ON, N6A 5B7 Canada
{lila, elena}@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] Facultad de Informática, Universidad Politécnica de Madrid,
Campus de Montegancedo s/n, Boadilla del Monte 28660,
Madrid, Spain
[4] Institute of Computer Science, Silesian University, Opava, Czech Republic
petr.sosik@fpf.slu.cz
[5] Department of Mathematics,
The University of Western Ontario, London, ON, N6A 5B7 Canada
thierrin@uwo.ca

Abstract. We formalize the notion of a DNA hairpin secondary structure, examining its mathematical properties. Two related secondary structures are also investigated, taking into the account imperfect bonds (bulges, mismatches) and multiple hairpins. We characterize maximal sets of hairpin-forming DNA sequences, as well as hairpin-free ones. We study their algebraic properties and their computational complexity. Related polynomial-time algorithms deciding hairpin-freedom of regular sets are presented. Finally, effective methods for design of long hairpin-free DNA words are given.

1 Introduction

A single strand of deoxyribonucleic acid (DNA) consists of a sugar-phosphate backbone and a sequence of nucleotides attached to it. There are four types of nucleotides denoted by A, C, T, and G. Two single strands can bind to each other if they have opposite polarity (strand's orientation in space) and are pairwise Watson-Crick complementary: A is complementary to T, and C to G. The binding of two strands is also called annealing. The ability of DNA strands to anneal to each other allows for creation of various secondary structures. A DNA hairpin is a particular type of secondary structure investigated in this paper. An example of a DNA hairpin structure is shown in Figure 1.

The reader is referred to [1, 16] for an overview of the DNA computing paradigm. The study of DNA secondary structures such as hairpin loops is motivated by finding reliable encodings for DNA computing techniques. These

⋆ Corresponding author.

A. Carbone and N.A. Pierce (Eds.): DNA11, LNCS 3892, pp. 158–170, 2006.

Fig. 1. An example of a DNA hairpin loop

techniques usually rely on a certain set of DNA bonds and secondary structures, while other types of bonds and structures are undesirable. Various approaches to the design of DNA encodings without undesirable bonds and secondary structures are summarized in [14] and [11]. For more details we refer the reader e.g. to [5, 12, 13]. Here we apply the formal language approach which has been used in [2, 7, 8, 10, 11] and others.

Hairpin-like secondary structures are of special importance for DNA computing. For instance, they play an important role in insertion/deletion operations with DNA. Hairpins are the main tool used in the Whiplash PCR computing techniques [18]. In [20] hairpins serve as a binary information medium for DNA RAM. Last, but not least, hairpins are basic components of "smart drugs" [3].

The paper is organized as follows. Section 2 introduces basic definitions, Section 3 presents results on hairpins, and in Section 4 we study two important variants of the hairpin definition. The first one takes into the account imperfect DNA bonds (mismatches, bulges), the second one is related to hairpin-based nanomachines. We study algebraic properties of (maximal) hairpin-free languages. The hairpin-freedom problem and the problem of maximal hairpin-free sets are both shown to be decidable in polynomial time for both regular and context-free languages. The last section provides methods of constructing long hairpin-free words.

2 Preliminary Definitions

We denote by X a finite alphabet and by X^* its corresponding free monoid. The cardinality of the alphabet X is denoted by $|X|$. The empty word is denoted by 1, and $X^+ = X^* - \{1\}$. A *language* is an arbitrary subset of X^*. For a word $w \in X^*$ and $k \geq 0$, we denote by w^k the word obtained as catenation of k copies of w. Similarly, X^k is the set of all words from X^* of length k. By convention, $w^0 = 1$ and $X^0 = \{1\}$. We also denote $X^{\leq k} = X^0 \cup X^1 \cup \ldots \cup X^k$. A *uniform*, or *block*, *code* is a language all the words of which are of the same length k, for some $k \geq 0$, and is therefore contained in X^k.

A mapping $\psi : X^* \to X^*$ is called a *morphism* (*anti-morphism*) of X^* if $\psi(uv) = \psi(u)\psi(v)$ (respectively $\psi(uv) = \psi(v)\psi(u)$) for all $u, v \in X^*$, and $\psi(1) = 1$. See Chapter 7 in [19] for a general overview of morphisms. An involution $\theta : X \longrightarrow X$ is defined as a map such that θ^2 is the identity function. An involution θ can be extended to a morphism or an antimorphism over X^*. In both cases θ^2 is the identity over X^* and $\theta^{-1} = \theta$. The simplest involution is

the identity function ϵ. A *mirror involution*μ is an antimorphic involution which maps each letter of the alphabet to itself.

We shall refer to the DNA alphabet $\Delta = \{A, C, T, G\}$, over which two involutions of interest are defined. The DNA *complementarity involution* γ is a morphism given by $\gamma(A) = T$, $\gamma(T) = A$, $\gamma(C) = G$, $\gamma(G) = C$. For example, $ACGCTG = \mu(GTCGCA) = \gamma(TGCGAC)$.

The antimorphic involution $\tau = \mu\gamma$ (the composite function of μ and γ, which is also equal to $\gamma\mu$), called the *Watson-Crick involution*, corresponds to the DNA bond formation of two single strands. If for two strings $u, v \in \Delta^*$ it is the case that $\tau(u) = v$, then the two DNA strands represented by u, v anneal as Watson-Crick complementary sequences.

A nondeterministic finite automaton (*NFA*) is a quintuple $A = (S, X, s_0, F, P)$, where 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 $sx \to t$, for $s, t \in S$, $x \in X$. If for every two productions $sx_1 \to t_1$ and $sx_2 \to t_2$ 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 $|S| + |P|$.

Analogously we define a *pushdown automaton (PDA)* and a *deterministic pushdown automaton (DPDA).* We refer the reader to [6, 19] for detailed definitions and basics of formal language theory.

3 Hairpins

Definition 1. *If θ is a morphic or antimorphic involution of X^* and $k > 0$, then a word $u \in X^*$ is said to be θ-k-hairpin-free or simply hp(θ,k)-free if $u = xvy\theta(v)z$ for some $x, v, y, z \in X^*$ implies $|v| < k$.*

Notice that words of length less than $2k$ are hp(θ,k)-free. If we interpret this definition for the DNA alphabet Δ and the Watson-Crick involution τ, then a hairpin structure with the length of bond at least k is a word that is not hp(θ,k)-free.

Definition 2. *Denote by $hpf(\theta, k)$ the set of all hp(θ,k)-free words in X^*. The complement of $hpf(\theta, k)$ is $hp(\theta, k) = X^* - hpf(\theta, k)$.*

Notice that $hp(\theta, k+1) \subseteq hp(\theta, k)$ for all $k > 0$.

Definition 3. *A language L is called θ-k-hairpin-free or simply $hp(\theta, k)$-free if $L \subseteq hpf(\theta, k)$.*

It follows by definition that a language L is hp(θ, k)-free *iff* $X^*vX^*\theta(v)X^* \cap L = \emptyset$ for all $|v| \geq k$. An analogous definition was given in [7], where a θ-k-hairpin-free language is called θ-subword-k-code. The authors focused on their coding properties and relations to other types of codes. They consider also the restriction on the length of the hairpin, namely that $1 \leq |y| \leq m$ for some $m \geq 1$. The

reader can verify that many of the results given in this paper remain valid if we apply this additional condition.

Example. Recall that γ is the DNA complementary involution over Δ^*, then:

$$hpf(\gamma, 1) = \{A, C\}^* \cup \{A, G\}^* \cup \{T, C\}^* \cup \{T, G\}^*$$

We give the necessary and sufficient conditions for finiteness of the languages $hpf(\theta, k)$, $k \geq 1$. Proofs of the following results can be found in [9]. Recall that $hpf(\mu, k)$ is the set of all words which do not contain any two non-overlapping mirror parts of length at least k.

Proposition 4. *Let X be a binary alphabet. For every word $w \in X^*$ in $hpf(\mu, 4)$ we have that $|w| \leq 31$. Moreover the following word of length 31 is in $hpf(\mu, 4)$:*

$$a^7ba^3bababab^2ab^2a^2b^7.$$

Proposition 5. *Consider a binary alphabet X. Then $hpf(\mu, k)$ is finite if and only if $k \leq 4$.*

Proposition 6. *Let θ be a morphic or antimorphic involution. The language $hpf(\theta, k)$ over a non-singleton alphabet X is finite if and only if one of the following holds:*

(a) $\theta = \epsilon$, the identity involution;
(b) $\theta = \mu$, the mirror involution, and either $k = 1$ or $|X| = 2$ and $k \leq 4$.

3.1 Properties of hp$(\theta, 1)$-Free Languages

Recall the definition of an embedding order: $u \leq_e v$ if and only if $u = u_1u_2 \cdots u_n$, $v = v_1u_1v_2u_2 \cdots \cdots v_nu_nv_{n+1}$ for some integer n with $u_i, v_j \in X^*$.

A language L is called *right $\leq_e$-convex* [21] if $u \leq_e w$, $u \in L$ implies $w \in L$. The following result is well known: *All languages (over a finite alphabet) that are right $\leq_e$-convex are regular.*

Proposition 7. *The language $hp(\theta, 1)$ is right $\leq_e$-convex (and hence regular).*

Proof. Observe that if $u = u_1u_2 \in hp(\theta, 1)$ and $w \in X^*$ then $u_1wu_2 \in hp(\theta, 1)$. Hence, for $u \in hp(\theta, 1)$, $u \leq_e v$ implies $v \in hp(\theta, 1)$.

Let $L \subseteq X^*$ be a nonempty language and let $S(L) = \{w \in X^* | u \leq_e w, u \in L\}$. Recall further that a set H with $\emptyset \neq H \subseteq X^+$ is called a *hypercode* over X^* *iff* $x \leq_e y$ and $x, y \in H$ imply $x = y$. That is, a hypercode is an independent set with respect to the embedding order.

Proposition 8. *Let θ be a morphic or antimorphic involution. Then there exists a unique hypercode H such that $hp(\theta, 1) = S(H)$.*

Proof. Let $H = \bigcup_{a \in X} a\theta(a)$, then $S(H) = \bigcup_{a \in X} X^*aX^*\theta(a)X^* = hp(\theta, 1)$. The uniqueness of H is immediate.

3.2 Properties of hp(θ, k)-Free Languages

Proposition 7, true for the case $k = 1$, cannot in general be extended to the case $k > 1$. Consider, for example, $X = \{a, b\}$ and a morphism $\theta(a) = b$, $\theta(b) = a$. If $u = a^2b^2$, then $u = a^2\theta(a^2)$ and hence $u \in hp(\theta, 2)$. But $u \leq_e w$ for $w = abab^2$, and $w \notin hp(\theta, 2)$. Therefore, the language $hp(\theta, 2)$ is not $\leq_e$-convex. However, the following weaker result is proven in [9].

Proposition 9. *The languages $hp(\theta, k)$ and $hpf(\theta, k)$, $k \geq 1$, are regular.*

Proposition 9 suggests an existence of fast algorithms solving some problems important from the practical point of view. We investigate two such problems now. Let θ be a fixed morphic or antimorphic involution and let $k \geq 1$ be an arbitrary but fixed integer.

Hairpin-Freedom Problem.

Input: A nondeterministic automaton M.
Output: Yes/No depending on whether $L(M)$ is hp(θ, k)-free.

Maximal Hairpin-Freedom Problem.

Input: A deterministic automaton M_1 accepting a hairpin-free language, and a NFA M_2.
Output: Yes/No depending on whether there is a word $w \in L(M_2) - L(M_1)$ such that $L(M_1) \cup \{w\}$ is hp(θ, k)-free.

We assume that M and M_1 are finite automata in the case of regular languages, and pushdown automata in the case of context-free languages.

Proposition 10. *The hairpin-freedom problem for regular languages is decidable in linear time (w.r.t. $|M|$).*

Proof. By definition, $L(M)$ is hp(θ, k)-free *iff* $L(M) \subseteq hpf(\theta, k)$ *iff* $L(M) \cap hp(\theta, k) = \emptyset$. This problem is solvable in time $\mathcal{O}(|M_k| \cdot |M|)$ for regular languages, where M_k is a NFA accepting $hp(\theta, k)$. The automaton M_k is fixed for a chosen k.

Proposition 11. *The maximal hairpin-freedom problem for regular languages is decidable in time $\mathcal{O}(|M_1| \cdot |M_2|)$.*

Proof. We want to determine whether there exists a word $w \in hpf(\theta, k)$ such that $w \notin L(M_1)$, but $w \in L(M_2)$. It is decidable in time $\mathcal{O}(|M_1| \cdot |M_2| \cdot |M'_k|)$ whether $(hpf(\theta, k) \cap L(M_2)) - L(M_1) = \emptyset$. The size of an NFA accepting $hpf(\theta, k)$ is denoted by $|M'_k|$. The automaton M'_k is fixed for a chosen k.

As an immediate consequence, for a given block code K of length l it is decidable in linear time with respect to $|K| \cdot l$, whether there is a word $w \in X^l - K$ such that $K \cup \{w\}$ is hp(θ, k)-free. This is of particular interest since the lab sets of DNA molecules form often a block code.

Notice also that for a finite set S of DNA sequences (which is the case of practical interest) the size of the automaton M (or M_1) is in the worst case proportional to the total length of all sequences in S.

Proposition 12. *The hairpin-freedom problem for context-free languages is decidable in cubic time (w.r.t. $|M|$).*

Proposition 13. *The maximal hairpin-freedom problem for deterministic context-free languages is decidable in time $\mathcal{O}((|M_1| \cdot |M_2|)^3)$.*

Proof. We want to determine if $\exists w \in hpf(\theta, k)$ such that $w \notin L(M_1)$, but $w \in L(M_2)$. Denote $M_1 = (Q_1, X, \Gamma, q_1, Z_0, F_1, P_1)$, and let $M_2' = (Q_2, X, q_2, F_2, P_2)$ be a NFA accepting the language $hpf(\theta, k) \cap L(M_2)$. Consider the PDA $M = (Q, X, \Gamma, q_0, Z_0, F, P)$, where $Q = Q_1 \times Q_2$, $q_0 = (q_1, q_2)$. For $p \in Q_1, q \in Q_2$, and $Z \in \Gamma$ we define:

(1) $(p, q)aZ \underset{P}{\rightarrow} (p', q')\alpha$ iif $paZ \underset{P_1}{\rightarrow} p'\alpha$ and $qa \underset{P_2}{\rightarrow} q'$,

(2) $(p, q)1Z \underset{P}{\rightarrow} (p', q)\alpha$ iif $p1Z \underset{P_1}{\rightarrow} p'\alpha$

Let $F = \{(p, q) | p \notin F_1 \text{ and } q \in F_2\}$. Then $L(M) = (hpf(\theta, k) \cap L(M_2)) - L(M_1)$, and the size of M is $\mathcal{O}(|M_1| \cdot |M_2|)$. Let G be a CFG such that $L(G) = L(M)$. Note that the construction of G takes cubic time w.r.t. $|M|$, see Theorem 7.31 of [6]. Finally, it is possible to decide in linear time w.r.t. $|G|$ (see Section 7.4.3 of [6]) whether $L(G) = \emptyset$ or not.

The time complexity of the above mentioned algorithms is furthermore proportional to the (constant) size of a NFA accepting the language $hp(\theta, k)$ or $hpf(\theta, k)$, respectively. Therefore we recall results from [9] characterizing the minimal size of these automata.

Proposition 14. *The number of states of a minimal NFA accepting the language $hp(\theta, k)$, $k \geq 1$, over an alphabet X with the cardinality ℓ, is between ℓ^k and $3\ell^k$. Its size is at most $3(\ell^k + \ell^{k+1})$.*

Proposition 15. *Let there be distinct letters $a, b \in X$ such that $a = \theta(b)$. Then the size of a minimal NFA accepting $hpf(\theta, k)$, $k \geq 1$, over an alphabet X with the cardinality ℓ, is at least $2^{(\ell-2)^k/2}$.*

Corollary 16. *Consider the DNA alphabet $\Delta = \{A, C, T, G\}$ and the Watson-Crick involution τ.*

(i) The size of a minimal NFA accepting $hp(\tau, k)$ is at most $15 \cdot 4^k$. The number of its states is between 4^k and $3 \cdot 4^k$.
(ii) The number of states of either a minimal DFA or an NFA accepting $hpf(\tau, k)$ is between $2^{2^{k-1}}$ and $2^{3 \cdot 2^{2k}}$.

The above results show that the size of a minimal NFA for $hp(\tau, k)$ grows exponentially w.r.t. k. However, one should recall that k is the *minimal* length of bond allowing for a stable hairpin. Therefore k is rather low in practical applications and the construction of the mentioned automaton remains computationally tractable.

4 Variants of Hairpins

4.1 Scattered Hairpins

It is a known fact that parts of two DNA molecules could form a stable bond even if they are not exact mutual Watson-Crick complements. They may contain some mismatches and even may have different lengths. Hybridizations of this type are addressed e.g. in [2] and [11]. Motivated by this observation, we consider now a generalization of hairpins.

Definition 17. *Let θ be an involution of X^* and let k be a positive integer. A word $u = wy \in X^*$ is θ-k-scattered-hairpin-free or simply shp(θ, k)-free if for all $t \in X^*$, $t \leq_e w$, $\theta(t) \leq_e y$ implies $|t| < k$.*

Fig. 2. An example of a scattered hairpin – a word in $shp(\tau, 11)$

Definition 18. *We denote by $shpf(\theta, k)$ the set of all shp(θ, k)-free words in X^*, and by $shp(\theta, k)$ its complement $X^* - shpf(\theta, k)$.*

Definition 19. *A language L is called θ-k-scattered-hairpin-free or simply shp(θ,k)-free if $L \subseteq shpf(\theta, k)$.*

Lemma 20. $shp(\theta, k) = S\left(\bigcup_{w \in X^k} w\theta(w) \right)$.

Based on the above immediate result, analogous statements as in Section 3 hold also for scattered hairpins. Proofs are straightforward and left to the reader.

Proposition 21. *(i) The language $shp(\theta, k)$ is right $\leq_e$ -convex.*
(ii) The languages $shp(\theta, k)$ and $shpf(\theta, k)$ are regular.
(iii) There exists a unique hypercode H such that $shp(\theta, k) = S(H)$.

Analogously as in Section 3 we can also define the *scattered-hairpin-freedom problem* and *maximal scattered-hairpin-freedom problem.* Then we easily obtain the following results whose proofs are analogous to those in Section 3.

Corollary 22. *(i) The scattered-hairpin-freedom problem is decidable in linear time for regular languages and in cubic time for context-free languages.*
(ii) The maximal scattered-hairpin-freedom problem is decidable in time $\mathcal{O}(|M_1| \cdot |M_2|)$ for regular languages and in time $\mathcal{O}((|M_1| \cdot |M_2|)^3)$ for deterministic context-free languages.

Also the size of the minimal automaton accepting the language $shp(\theta, k)$ is similar to the case of $hp(\theta, k)$ in Section 3.2.

For the proof of the next proposition we recall the following technical tools from [4].

Definition 23. *A set of pairs of strings* $\{(x_i, y_i) \mid i = 1, 2, \ldots, n\}$ *is called a* fooling set *for a language* L *if for any* i, j *in* $\{1, 2, \ldots, n\}$,

(1) $x_i y_i \in L$, *and*
(2) if $i \neq j$ *then* $x_i y_j \notin L$ *or* $x_j y_i \notin L$.

Lemma 24. *Let* F *be a fooling set of a cardinality* n *for a regular language* L. *Then any NFA accepting* L *needs at least* n *states.*

Proposition 25. *The number of states of a minimal NFA accepting the language* $shp(\theta, k)$, $k \geq 1$, *over an alphabet* X *with the cardinality* ℓ, *is between* ℓ^k *and* $3\ell^k$, *its size is at most* $7\ell^k + 3\ell^{k+1}$.

Proof. Let $M_k = (S, X, s_1, F, P)$ be an NFA accepting $shp(\theta, k)$. The statement is trivial for the cases $\ell = 1$ or $k = 1$. Assume for the rest of the proof that $k \geq 2$ and $\ell \geq 2$.

(i) The reader can easily verify that the set $F = \{(w, \theta(w)) | w \in X^k\}$ is a fooling set for $hp(\theta, k)$. Therefore $|S| \geq \ell^k$.

(ii) Let

$$S = \{s_w, p_w \mid w \in X^{\leq k-1}\} \cup \{q_w \mid w \in X^k\}.$$

Let further $F = \{p_1\}$. The set of productions P is defined as follows:

$$\begin{array}{ll}
s_v a \to s_w \text{ iif } va = w, & \text{for each } v \in X^{\leq k-2},\ a \in X;\\
s_v a \to q_w \text{ iif } va = w, & \text{for each } v \in X^{k-1},\ a \in X;\\
q_w a \to p_v \text{ iif } \theta(av) = w, & \text{for each } v \in X^{k-1},\ a \in X;\\
p_w a \to p_v \text{ iif } av = w, & \text{for each } v \in X^{\leq k-2},\ a \in X.\\
ra \to r & \text{for all } r \in S,\ a \in X.
\end{array}$$

The reader can verify that $L(M_k) = shp(\theta, k)$, and that $|S| \leq 3\ell^k$, $|P| \leq 4\ell^k + 3\ell^{k+1}$, therefore $|M_k| \leq 7\ell^k + 3\ell^{k+1}$.

Note: An example of a similar automaton accepting the language $hp(\theta, k)$ can be found in [9].

4.2 Hairpin Frames

In this section we point out the following two facts. First, long DNA and RNA molecules can form complicated secondary structures as that shown in Figure 3. Second, simple hairpins can be useful in various DNA computing techniques and nanotechnologies, as in [3, 18, 20] and others. Hence it may be desirable to design DNA strands forming simple hairpins but avoiding more complex structures. This motivates another extension of the results from Section 3.

Definition 26. *The pair* $(v, \theta(v))$ *of a word* u *in the form* $u = xvy\theta(v)z$, *for* $x, v, y, z \in X^*$, *is called an* hp-pair *of* u. *The sequence of hp-pairs* $(v_1, \theta(v_1))$, $(v_2, \theta(v_2))$, $\cdots$, $(v_j, \theta(v_j))$ *of the word* u *in the form:*

$$u = x_1 v_1 y_1 \theta(v_1) z_1 x_2 v_2 y_2 \theta(v_2) z_2 \cdots x_j v_j y_j \theta(v_j) z_j$$

is called an hp-frame of degree j *of* u *or simply an* hp(j)-frame *of* u.

An hp-pair is an hp-frame of degree 1. The definition of hairpin frames characterizes secondary structures containing several complementary sequences such as that in Fig. 3.

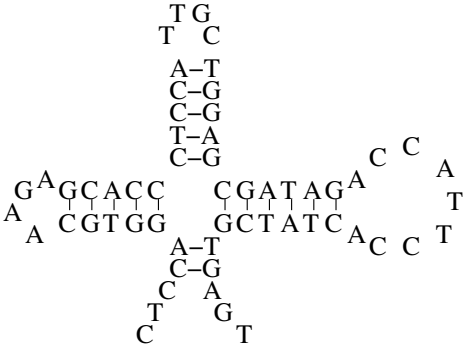

Fig. 3. An example of a hairpin frame – a word in $hp(\tau, fr, 3)$

A word u is said to be an *hp(fr,j)-word* if it contains at least one hp-frame of degree j. Observe that there may be more ways of finding hp-pairs in u, resulting in hp-frames of various degrees. Obviously, any hp(fr,j)-word is also hp(fr,i) for all $1 \leq i \leq j$.

Definition 27. *For an involution* θ *we denote by* $hp(\theta, fr, j)$ *the set of all hp(fr,j)-words* $u \in X^*$, *and by* $hpf(\theta, fr, j)$ *its complement in* X^*.

The results in Section 3, concerning the languages $hp(\theta, 1)$ and $hpf(\theta, 1)$, can easily be extended for the case of hairpin frames. Proofs are left to the reader.

Lemma 28. $hp(\theta, fr, j) = hp(\theta, 1)^j = \left(\bigcup_{a \in X} X^* a X^* \theta(a) X^* \right)^j$.

Proposition 29. *(i) The language* $hp(\theta, fr, j)$ *is right* $\leq_e$ *-convex.*
(ii) The languages $hp(\theta, fr, j)$ *and* $hpf(\theta, fr, j)$ *are regular.*
(iii) There exists a unique hypercode H *such that* $hp(\theta, fr, j) = S(H)$.

Corollary 30. *(i) The* hp(fr,j)-freedom *problem is decidable in linear time for regular languages and in cubic time for context-free languages.*

(ii) The maximal hp(fr,j)-freedom *problem is decidable in time* $\mathcal{O}(|M_1|\cdot|M_2|)$ *for regular languages and in time* $\mathcal{O}((|M_1|\cdot|M_2|)^3)$ *for deterministic context-free languages.*

Proposition 31. *The size of a minimal NFA accepting the language* $hp(\theta, fr, j)$, $j \geq 1$, *over an alphabet* X *with the cardinality* ℓ, *is at most* $4\ell j + 2j + 1$.

Proof. The statement follows by the construction of an NFA $M = (S, X, s_1, F, P)$ accepting the language $hp(\theta, fr, j)$. Let

$$S = \{s_0, s_1, \ldots, s_j\} \cup \{p_i^k \mid 1 \leq i \leq j,\ 1 \leq k \leq \ell\}.$$

Let further $F = \{s_j\}$, and denote $X = \{a_1, \ldots, a_\ell\}$. The set of productions P is defined as follows:

$$\begin{array}{ll} s_{i-1}a_k \to p_i^k,\ p_i^k\theta(a_k) \to s_i & \text{for all } 1 \leq i \leq j,\ 1 \leq k \leq \ell; \\ sa \to s & \text{for all } s \in S,\ a \in X. \end{array}$$

The reader can verify that $L(M_k) = hp(\theta, fr, j)$, and that $|M| = 4\ell j + 2j + 1$.

Unlike the cases of hairpins or scattered hairpins, the size of the minimal NFA accepting $hp(\theta, fr, j)$ is $\mathcal{O}(j\ell)$. However, if we considered also a minimal length k of the hairpin bonds, we would obtain the same exponential size of the automaton as in Section 3.2, but multiplied by j.

5 Construction of Long Hairpin-Free Words

In this section we discuss the problem of constructing long hp(θ, k)-free words for the cases where θ is the Watson-Crick involution and $\theta = \epsilon$. This question is relevant to various encoding problems of DNA computing. For example, in [20] the authors consider n-bit memory elements that are represented by DNA words of the form

$$u_1v_1w_1\theta(v_1)\cdots u_nv_nw_n\theta(v_n)u_{n+1},$$

such that (i) all the u's and v's have length 20 and the w's have length 7, and (ii) the only bonds permitted in a word of this form are the bonds between v_i and $\theta(v_i)$ for all $i = 1, \ldots, n$. This encoding problem can be solved if we first construct a long hp(θ, k)-free word w of length $(20 + 20 + 7)n + 20 = 47n + 20$. Then w can be written in the form

$$u_1v_1w_1 \cdots u_nv_nw_nu_{n+1}$$

and is such that no bonds can occur between any two subwords of length k of w. Here k is the parameter that represents the smallest length of a block of nucleotides that can form a stable bond with a corresponding block of complementary nucleotides – see also the relevant discussion in [11].

For the case where θ is the Watson-Crick involution we consider the method of [11] for constructing $(\theta, H_{0,k})$-bond-free languages L. Such a language L has the

property that, for any two subwords u and v of L of length k, one has that $u \neq \theta(v)$. Note that each word of L is a hp(θ, k)-free word. Moreover, if L is infinite then it contains arbitrarily long words, hence, also words of length $47n + 20$, for any n, as required in the encoding problem discussed in the beginning of this section. We also note that if L is (θ, k)-bond-free then it is (θ, k')-bond-free for any $k' \geq k$. The method of [11] is based on the *subword closure* language operation $\otimes$: Let S be a set of words of length k. Then $S^{\otimes}$ is the set of all words w of length at least k such that any subword of w of length k belongs to S. We note that given the set S one can construct a deterministic finite automaton accepting $S^{\otimes}$ in linear time [11]. The method is as follows. Let S be any set of words of length k such that $S \cap \theta(S) = \emptyset$. Then $S^{\otimes}$ is a $(\theta, H_{0,k})$-bond-free language. In our case, we wish to choose S such that $S^{\otimes}$ is infinite. For example, let S_2 be the set $\{AA, AC, CA, CC, AG, GA\}$. In [11] the authors show an automaton accepting $S_2^{\otimes}$. As $S_2^{\otimes}$ contains the set $(ACCAGAC)^+$ it follows that $S_2^{\otimes}$ is infinite as well.

For the case of $\theta = \epsilon$, we consider a totally different approach. Let $H(K)$ denote the minimum Hamming distance between any two different codewords of a code K. A language K is said to be a *solid code* if (i) no word of K is a subword of another word of K, and (ii) a proper and nonempty prefix of K cannot be a suffix of K. See [17] or Chapter 8 in [19] for background information on codes.

Proposition 32. *Let $k \geq 2$ and let K be a uniform solid code of length k. If $H(K) > \lfloor k/2 \rfloor$, or $H(K) = \lfloor k/2 \rfloor$ and there are no different codewords with a common prefix of length $\lfloor k/2 \rfloor$, then the word $w_1 ... w_n$ is hp(θ, k)-free for all $n \leq \text{card}(K)$ and for all pairwise different codewords $w_1, ..., w_n$.*

Proof. Assume there is $v \in X^k$ such that $w_1 ... w_n = xvyvz$ for some words x, y, z. If $|x|$ is a multiple of k then $v = w_j$ for some $j \geq 1$. As the w_i's are different, $|y|$ cannot be a multiple of k. Hence, $v = s_t p_{t+1}$, where $t > j$ and s_t is a proper and nonempty suffix of w_t and p_{t+1} is a proper and nonempty prefix of w_{t+1}; a contradiction. Now suppose $|x|$ is not a multiple of k. Then, $v = s_j p_{j+1}$ for some nonempty suffix s_j and prefix p_{j+1}. Again, the second occurrence of v cannot be in K. Hence, $v = s_t p_{t+1}$ for some $t \geq j$. Hence, $s_j p_{j+1} = s_t p_{t+1}$. If $|s_j| \neq |s_t|$, say $|s_j| > |s_t|$, then a prefix of p_{t+1} is also a suffix of s_j; which is impossible. Hence, $s_j = s_t$ and $p_{j+1} = p_{t+1}$.

Note that $H(K) \geq \lfloor k/2 \rfloor$ and, therefore, $\lfloor k/2 \rfloor \leq H(p_{j+1}s_{j+1}, p_{t+1}s_{t+1}) = H(s_{j+1}, s_{t+1}) \leq |s_{j+1}| = k - |p_{j+1}|$. Hence, $|p_{j+1}| \leq \lceil k/2 \rceil$. Similarly, $|s_j| \leq \lceil k/2 \rceil$. Also, as $k = |s_j| + |p_{j+1}|$, one has that $|s_j|, |p_{j+1}| \in \{\lfloor k/2 \rfloor, \lceil k/2 \rceil\}$. If $H(K) = \lfloor k/2 \rfloor$ then $p_{j+1} = p_{t+1}$ implies that w_{j+1} and w_{t+1} have a common prefix of length $\lfloor k/2 \rfloor$; a contradiction. If $H(K) > \lfloor k/2 \rfloor$ then both p_{j+1} and s_j are shorter than $\lceil k/2 \rceil$ which contradicts with $k = |s_j| + |p_{j+1}|$.

Suppose the alphabet size $|X|$ is $l > 2$. We can choose any symbol $a \in X$ and consider the alphabet $X_1 = X - \{a\}$. Then for any uniform code $F \subseteq X_1^{k-1}$ it follows that the code Fa is a uniform solid code of length $k : Fa \subseteq X^k$. We are interested in cases where the code F is a linear code of type $[k-1, m, d]$. That is, F is of length $k-1$, cardinality $(l-1)^m$, and $H(F) = d$, and there is an m

by $k-1-m$ matrix G over X_1 such that $F = \{w * [I_m|G] : w \in X_1^m\}$, where I_m is the identity m by m matrix and $*$ is the multiplication operation between a 1 by m vector and an m by m matrix. Thus, $u \in F$ *iff* $u = wx$ for some $w \in X_1^m$ and $x \in X_1^{k-1-m}$ and $x = wG$.

Proposition 33. *Let F be a linear code over X_1 of type $[k-1, m, \lfloor k/2 \rfloor]$. If $m \leq \lfloor k/2 \rfloor$ or k is even then the word $w_1..w_n$ is hp(θ, k)-free for all $n \leq \mathrm{card}(F)$ and for all pairwise different codewords $w_1, ..., w_n$ in Fa.*

Proof. It is sufficient to show that $H(Fa) = \lfloor k/2 \rfloor$ and there are no different words in Fa with a common prefix of length $\lfloor k/2 \rfloor$. Obviously $H(Fa) = H(F) = \lfloor k/2 \rfloor$. As F is generated by a matrix $[I_m|G]$, where G is a matrix in $X_1^{m \times (k-1-m)}$, it follows that there can be no different words in F with a common prefix of length m. If $m \leq \lfloor k/2 \rfloor$ then there can be no different words in Fa with a common prefix of length $\lfloor k/2 \rfloor$. If k is even, consider the well known bound on $|F|$: $|F| \leq |X_1|^{k-1-\lfloor k/2 \rfloor+1}$. Hence, $|X_1|^m \leq |X_1|^{\lfloor k/2 \rfloor}$ which gives $m \leq \lfloor k/2 \rfloor$. Hence, again, we are done.

By the above one can construct an hp(θ, k)-free word of length nk, for some $n \leq \mathrm{card}(F)$, for every choice of n different words in Fa. It is interesting that, for $k = 13$ and $|X| = 4$, the famous Golay code G_{12} of type $[12, 6, 6]$ satisfies the premises of the above Proposition.

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 201/06/0567 to P.S.

References

1. M. Amos, *Theoretical and Experimental DNA Computations.* Springer-Verlag, Berlin, 2005.
2. 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 *Proc. 8th Workshop on DNA-Based Computers*, M. Hagiya, A. Ohuchi, Eds., *LNCS* 2568 (2002), 182–195.
3. Y. Benenson, B. Gil, U. Ben-Dor, R. Adar, E. Shapiro, An autonomous molecular computer for logical control of gene expression. *Nature* 429 (2004), 423–429.
4. J.C. Birget, Intersection and union of regular languages and state complexity. *Information Processing Letters* 43 (1992), 185–190.
5. J. Chen, R. Deaton, M. Garzon, J.W. Kim, D. Wood, H. Bi, D. Carpenter, Y.-Z. Wang, Characterization of non-crosshybridizing DNA oligonucleotides manufactured *in vitro*. In [15], 132–141.
6. J. Hopcroft, J. Ullman, R. Motwani, Introduction to Automata Theory, Languages, and Computation, 2nd ed., Addison-Wesley, 2001.

7. N. Jonoska, D. Kephart, K. Mahalingam, Generating DNA code words. *Congressus Numerantium* 156 (2002), 99–110.
8. N. Jonoska, K. Mahalingam, Languages of DNA based code words. In *DNA Computing, 9th International Workshop on DNA Based Computers*, J. Chen and J.H. Reif, Eds., *LNCS* 2943 (2004), 61–73.
9. L. Kari, S. Konstantinidis, P. Sosík, G. Thierrin, On hairpin-free words and languages. In *Developments in Language Theory, 9th Int. Conf.*, C. de Felice and A. Restivo, Eds., *LNCS* 3572 (2005), 296–307.
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, Bond-free languages: formalizations, maximality and construction methods. In [15], 16–25.
12. S. Kobayashi, Testing Structure Freeness of Regular Sets of Biomolecular Sequences. In [15], 395–404.
13. A. Marathe, A. Condon, R. Corn, On combinatorial DNA word design. *DNA based Computers V*, DIMACS Series, E.Winfree, D.Gifford Eds., AMS Press, 2000, 75–89.
14. G. Mauri, C. Ferretti, Word Design for Molecular Computing: A Survey. In *DNA Computing, 9th International Workshop on DNA Based Computers*, J. Chen and J.H. Reif, Eds., LNCS 2943 (2004), 37–46.
15. G. Mauri, C. Ferretti, Eds., *DNA 10, Tenth International Meeting on DNA Computing.* Preliminary proceedings, University of Milano-Bicocca, 2004.
16. G. Paun, G. Rozenberg, A. Salomaa, *DNA Computing: New Computing Paradigms*, Springer Verlag, Berlin, 1998.
17. S. Roman, *Coding and Information Theory,* Springer-Verlag, New York, 1992.
18. J. A. Rose, R. J. Deaton, M. Hagiya, A. Suyama, PNA-mediated Whiplash PCR. In *DNA Computing, 7th International Workshop on DNA Based Computers*, N. Jonoska and N. C. Seeman, Eds., LNCS 2340 (2002), 104–116.
19. G. Rozenberg, A. Salomaa, Eds., *Handbook of Formal Languages*, Vol. 1, Springer Verlag, Berlin, 1997.
20. N. Takahashi, A. Kameda, M. Yamamoto, A. Ohuchi, Aqueous computing with DNA hairpin-based RAM. In [15], 50–59.
21. G. Thierrin, Convex languages. *Proc. IRIA Symp. North Holland* 1972, 481–492.