Further Properties of Self-assembly by Hairpin Formation

Henning Bordihn[1], Victor Mitrana[2,3,4(✉)], Andrei Păun[3,4], and Mihaela Păun[4,5]

[1] Department of Computer Science, University of Potsdam, Potsdam, Germany
henning@cs.unipotsdam.de
[2] Department of Information Systems, Polytechnic University of Madrid, Madrid, Spain
victor.mitrana@upm.es
[3] Faculty of Mathematics and Computer Science, University of Bucharest, Bucharest, Romania
apaun@fmi.unibuc.ro
[4] National Institute for Research and Development of Biological Science, Bucharest, Romania
[5] Faculty of Administration and Business, University of Bucharest, Bucharest, Romania
mihaela.paun@gmail.com

Abstract. We continue the investigation of three operations on words and languages with motivations coming from DNA biochemistry, namely unbounded and bounded hairpin completion and hairpin lengthening. We first show that each of these operations can be used for replacing the third step, the most laborious one, of the solution to the CNF-SAT reported in [28]. As not all the bounded/unbounded hairpin completion or lengthening of semilinear languages remain semilinear, we study sufficient conditions for semilinear languages to preserve their semilinearity property after applying once either the bounded or unbounded hairpin completion, or lengthening. A similar approach is then started for the iterated variants of the three operations. A few open problems are finally discussed.

Keywords: DNA hairpin formation · Hairpin completion · Bounded hairpin completion · Hairpin lengthening · Semilinearity property

1 Introduction

A *DNA strand* can be abstracted, if its spatial structure is ignored, as a word over the four-letter alphabet $\{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$ where the letters represent the nucleotides

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, project number POC P-37-257. Victor Mitrana has also been supported by the Alexander von Humboldt Foundation.

I. McQuillan and S. Seki (Eds.): UCNC 2019, LNCS 11493, pp. 37–51, 2019.
https://doi.org/10.1007/978-3-030-19311-9_5

Adenine, Cytosine, Guanine, and Thymine, respectively. In the double helix of DNA, the two strands are end-to-end chemically oriented in opposite directions, namely from 5′ to 3′ and from 3′ to 5′, respectively, hence they are anti-parallel, which permits base pairing by the Watson-Crick complementarity, where A is complementary to T and C to G. This base pairing, by the hydrogen bonds between complementary nucleotides under some specific environment conditions, is one of the main properties of DNA which the process of DNA replication is based on.

Throughout this note, we use a bar-notation for the Watson-Crick complement; thus $\overline{\mathsf{A}} = \mathsf{T}$ and $\overline{\mathsf{T}} = \mathsf{A}$ as well as $\overline{\mathsf{C}} = \mathsf{G}$ and $\overline{\mathsf{G}} = \mathsf{C}$. We extend this bar-notation to sequences of nucleotides (words) by $\overline{s_1 \cdots s_n} = \overline{s_n} \cdots \overline{s_1}$.

Due to the base pairing discussed above, a single stranded DNA molecule may produce a hairpin structure as shown in Fig. 1.

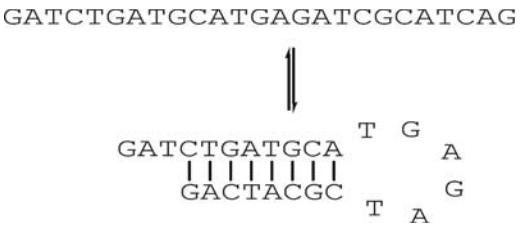

Fig. 1. Hairpin structure.

In many DNA-based algorithms, the single stranded DNA molecules that formed already or might form a hairpin structure cannot be used in the subsequent computations. Hairpin or hairpin-free DNA structures have numerous applications in DNA computing and molecular genetics. In a series of papers, see, e.g., [7,9,10], such structures are discussed in the context of finding sets of DNA sequences which are unlikely to lead to "bad" hybridizations, that is DNA fragments do not anneal to complementary segments in undesired ways. It has been claimed in different places that the potential of DNA molecules to form self-assembly structures might be employed for designing solutions to hard problems, see, e.g., [30]. Thus a first DNA-based solution to the CNF-SAT problem, where one of the main steps was implemented on the basis of hairpin formation by single-stranded DNA molecules was reported in [28]. In this DNA-based solution, the third phase is mainly based on the elimination of hairpin structured molecules. A rather long and complicated lab methodology is discussed for implementing this phase. We propose a modification of this algorithm that may use any of the three operations considered here and could be easily implemented. Different types of hairpin and hairpin-free languages were defined in [3,25], and [15], where they were studied from a language theoretical point of view.

A common lab technique to lengthen DNA or to make copies of regions of DNA is called *polymerase chain reaction* (PCR). This technique which is rather cheap, easy and reliable is widely used in molecular biology but also in

DNA computing. This technique is used to produce a complete double stranded DNA molecule starting from a single stranded molecule as informally follows: the starting single strand (usually called *primer*) is bonded to its $3'$ end with another shorter strand (usually called *template*) by Watson-Crick complementarity. Then a *polymerization buffer* with many copies of the four nucleotides, and a DNA polymerase (an enzyme) will concatenate nucleotides to the primer by complementing the template. A very closely related intramolecular reaction, called whiplash PCR, which employs polymerization stop is also used here.

These principles discussed above have been the source of inspiration for introducing in [5] a new formal operation on words, namely *hairpin completion*. We now informally explain the hairpin completion operation and how it can be related to the aforementioned biological concepts. Let us consider the following hypothetical biological situation: we are given one single stranded DNA molecule z such that either a prefix or a suffix of z is Watson-Crick complementary to a subword of z. Then the prefix or suffix of z and the corresponding subword of z get annealed by complementary base pairing and then z is lengthened by DNA polymerases up to a complete hairpin structure. Finally, the linear structure of the molecule is restored by DNA denaturation such that the whole process described above can be resumed. This is illustrated in Fig. 2, where $z = \gamma\alpha\beta\overline{\alpha}$.

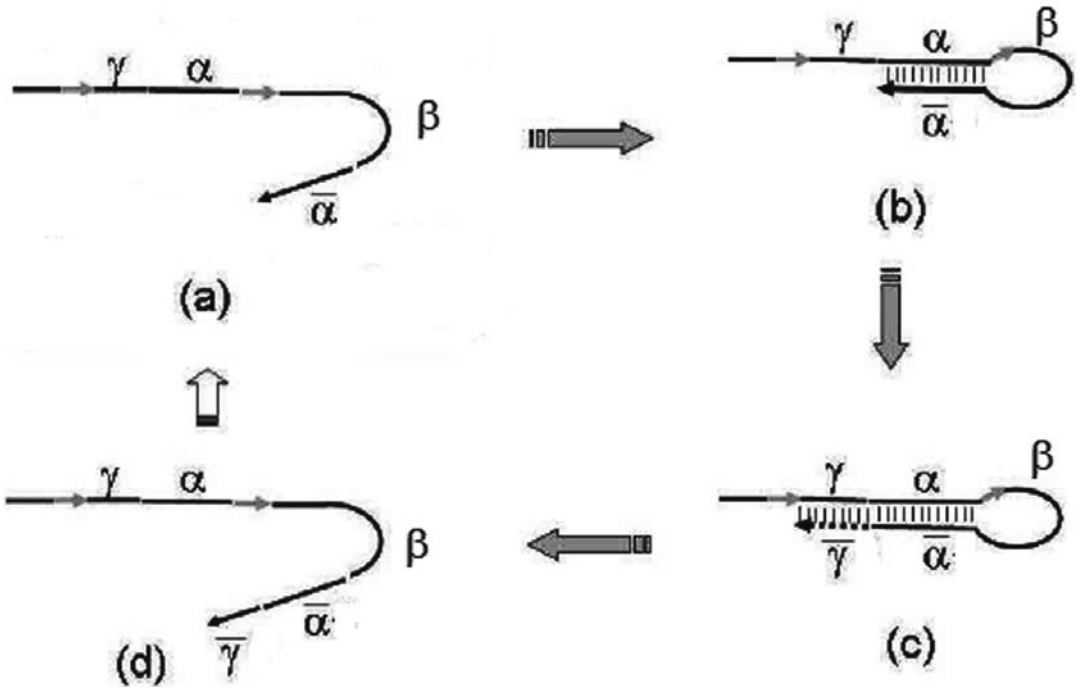

Fig. 2. (a) A single stranded DNA molecule; (b) hairpin formation with part α; (c) polymerization extension of γ; (d) restoring the linear structure.

The mathematical expression of this hypothetical situation defines the hairpin completion operation. This operation is considered in [5] as an abstract operation on formal languages. A series of subsequent works extended the

investigation of this operation and proposed a few variants. Some algorithmic problems regarding the hairpin completion are investigated in [18]. In the aforementioned papers, no restriction is imposed on the length of prefix or suffix added by the hairpin completion. In [14] one considers a restricted variant of the hairpin completion, called bounded hairpin completion. Another operation derived from the biological phenomenon described above, namely hairpin lengthening has been introduced in [21] and further investigated in [22]. Two other operations inspired by these biological phenomena are: *WK-superposition* [2] and [20], *overlap assembly* [6] and [8].

We propose a modification of the solution to the CNF-SAT based on the DNA hairpin formation reported in [28]. This modification, which may use any of the operations considered here, eliminates the third step of that algorithm which seems to be the most laborious one. Our algorithm appears, at least theoretically, to be more easily implemented in a laboratory because our step might be implemented by gel electrophoresis.

A set of tuples of integers constructed as a multi-dimensional arithmetic progression is called linear. A finite union of linear sets is called semilinear. The semilinear sets are exactly the sets definable in Presburger arithmetic [12]. They also can be viewed as a generalization of ultimately periodic sets of natural numbers to any given dimension. Semilinear sets have numerous applications in theoretical computer science and mathematics: the verification of some subclasses of Minsky counter machines, automata and logics over unranked trees with counting, equational Horn clauses, systems of Diophantine equations, etc.

Semilinearity is considered to be a linguistic invariant for natural languages, that is a property which remains robust under slight syntactic modifications. Clearly the class of semilinear languages is not directly useful as it contains undecidable languages, but it might be useful to separate classes of languages depending on the defining formalism. As many researchers have considered DNA as a language, pointing out that genetic code and natural language share a number of units, structures and operations, we may ask whether a bio-inspired operation applied to a semilinear language preserves the semilinearity property of that language, especially if the operation is intended to capture some biological phenomena. Along these lines it is worth mentioning that the input sets decidable by a chemical reaction network are precisely the semilinear sets [4].

The three aforementioned operations are considered in relation to the semilinearity property of defined languages. We first show that not all the bounded or unbounded hairpin completion or hairpin lengthening of semilinear languages remain semilinear, but we give sufficient conditions for semilinear languages to preserve the semilinearity property after applying once the bounded or unbounded hairpin completion or hairpin lengthening. A similar investigation is then done for the iterated variants of the three operations. Finally, we discuss a few directions for further investigation.

2 Basic Definitions

We assume the reader to be familiar with the fundamental concepts of formal language theory and automata theory, see, e.g., [27].

An alphabet is a finite set of letters. For a finite set A we denote by $card(A)$ the cardinality of A. The set of all words over an alphabet V is denoted by V^*. The empty word is denoted by ε; moreover, $V^+ = V^* \setminus \{\varepsilon\}$. Given a word w over an alphabet V, we denote by $|w|$ its length, while $|w|_a$ denotes the number of occurrences of the letter a in w. If $w = xyz$ for some $x, y, z \in V^*$, then x, y, z are called prefix, subword, suffix, respectively, of w. If $0 < |x|, |z| < |w|$, then x and z are called proper prefix and proper suffix, respectively. If x or z is non-empty, then y is called a proper subword of w.

An *involution* over a set S is a bijective mapping $\sigma : S \longrightarrow S$ such that $\sigma = \sigma^{-1}$. In this paper's context, any involution σ over some set S such that $\sigma(a) \neq a$ for all $a \in S$ is called a *Watson-Crick involution*. Despite that this is nothing more than a fixed point-free involution, we prefer this terminology since the hairpin completion defined later is inspired by the DNA biochemistry, where the Watson-Crick base-pair complementarity plays an important role. Let $\bar{\cdot}$ be a Watson-Crick involution over some alphabet V; we extend this involution to an anti-morphism from V^* to V^* in the usual way, namely $\overline{a_1 a_2 \dots a_n} = \overline{a_n} \dots \overline{a_2}\ \overline{a_1}$. We say that the letters a and $\overline{a}$ are complementary to each other. If $\bar{\cdot}$ is a Watson-Crick involution over some alphabet V, then clearly $\overline{V} = \{\overline{a} \mid a \in V\} = V$; such an alphabet is called a Watson-Crick alphabet. If not otherwise stated, all the alphabets in this note are Watson-Crick alphabets and $\bar{\cdot}$ is a fixed involution such that $\overline{a} \neq a$ for any letter a in the alphabet. Remember that the DNA alphabet consists of four letters, $V_{DNA} = \{A, C, G, T\}$, which are abbreviations for the four nucleotides and we have set $\overline{A} = T$, $\overline{C} = G$.

Let V be an alphabet, for any $w \in V^+$ we define the *k-hairpin lengthening* of w, denoted by $HL_k(w)$, for some $k \geq 1$, as follows [21]:

$$\begin{aligned} HLP_k(w) = \{&\overline{\delta} w \mid w = \alpha\beta\overline{\alpha}\gamma, |\alpha| = k,\ \alpha, \beta, \gamma \in V^+, \\ &\text{and } \delta \text{ is a proper prefix of } \gamma\} \\ HLS_k(w) = \{&w\overline{\delta} \mid w = \gamma\alpha\beta\overline{\alpha}, |\alpha| = k,\ \alpha, \beta, \gamma \in V^+, \\ &\text{and } \delta \text{ is a proper suffix of } \gamma\}, \\ HL_k(w) = &\ HLP_k(w) \cup\ HLS_k(w) \end{aligned}$$

If in the above definitions, one replaces "δ is a proper prefix/suffix of γ" by "$\delta = \gamma$", then the operation is called *k-hairpin completion* [5] and it is denoted by HC_k. Furthermore, if $\delta = \gamma$ and $|\gamma| \leq p$, for some positive integer p, then the operation is called *p-bounded k-hairpin completion* [14] and it is denoted by pHC_k. The *hairpin lengthening/completion* and *bounded hairpin completion* of w is defined by

$$H(w) = \bigcup_{k \geq 1} H_k(w),$$

where $H \in \{HL, HC, pHC\}$, respectively. The hairpin lengthening/completion and bounded hairpin completion is naturally extended to languages by

$$H_k(L) = \bigcup_{w \in L} H_k(w) \qquad H(L) = \bigcup_{w \in L} H(w),$$

where $H \in \{HL, HC, pHC\}$, respectively. We want to stress that the biological phenomenon is just a source of inspiration for introducing the operations defined above. First, it is known that a "stable" hairpin structure as above is possible if the subword α is sufficiently long. Second, it is known that DNA polymerase can act continuously only in the $5' \longrightarrow 3'$ due to the greater stability of $3'$ when attaching new nucleotides. As one can see in our definitions, we have allowed polymerase to extend continuously in either end; however, polymerase can also act in the opposite direction, but in short "spurts" (Okazaki fragments).

For a class of languages $\mathcal{F}$ and an integer $k \geq 1$ we denote the class of the hairpin lengthening/completion and bounded hairpin completion of languages in $\mathcal{F}$ by $H_k(\mathcal{F}) = \{H_k(L) \mid L \in \mathcal{F}\}$.

3 A New Solution to the CNF-SAT Using Hairpin Operations

The Conjunctive Normal Form (CNF) - Satisfiability (SAT) problem asks to find, if any, Boolean assignments that satisfy a given formula in conjunctive normal form, that is a formula of the form $C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where each C_i is a clause and $\wedge$ is the logical AND operator. Each clause C_i is of the form $C_i = l_{i_1} \vee l_{i_2} \vee \cdots \vee l_{i_{k_i}}$, where each l_{i_j} is a literal (a variable or its negation) and $\vee$ is the logical OR operation. A DNA-based algorithm for solving CNF-SAT having three main steps was proposed in [28]. We refer to [28] for all unexplained notions.

S1. Generate all the literal strings with respect to the given formula. The literal strings encode conjunctions of literals one per clause. Such a conjunction is

$$l_{1_{r_1}} \wedge l_{2_{r_2}} \wedge \cdots \wedge l_{m_{r_m}},$$

where $1 \leq r_j \leq k_j$ for all $1 \leq j \leq m$. This step is implemented by a routine ligation reaction which concatenates DNA segments encoding literal units to larger strings according to the input formula.

S2. Allow all literal strings formed in step S1 to form hairpins. Note that the hairpin structures formed in this step are not necessarily as that of (b) in Figure 2. More precisely, it is not necessary that one of the segments that get annealed by complementary base pairing be a prefix or a suffix. This step may be implemented by regulating the temperature.

S3. Remove all the molecules that formed a hairpin in step S2. This has been done by two techniques: (i) remove DNA molecules that formed hairpin by enzymatic digestion, and (ii) amplify the population of DNA molecules without hairpin by a variant of PCR, called "exclusive PCR". This is the reason why this step is very laborious.

The correctness of the algorithm follows from the fact that a molecule forms a hairpin in the second step, if and only if it encodes an inconsistent assignment, that is when both a variable and its negation are true. A long and rather complicated procedure based on the two techniques described above is proposed for implementing the third step. We propose the following modification of this algorithm. First, instead of literal units as in the previous algorithm, we use extended literal units, that is literal units as above linked to a unique segment, say α which is newly designed and cannot form hairpin structure with any other segment. As in the first step of the previous algorithm, literal strings are obtained by extending the already produced strings with extended literal units associated with a new clause which was not considered yet. However, before extending the current literal strings with the new extended literal unit, we propose to allow all current literal strings obtained in the previous step to form hairpins and be extended by one of the three hairpin operations. All those literal strings that contain the complement of α will be removed such that they will not be extended with new extended literal units. The algorithm is informally described below. This situation corresponds to the case of using hairpin completion.

S1. Generate all the exteded literal units associated with each clause.
S2. For each clause C that has not been considered yet, do the following:
S2.1. Extend by ligation the current literal strings with the extended literal units associated with C, say α.
S2.2. Allow all current literal strings obtained in the previous step to form hairpins and be extended by one of the three hairpin operations. Let us choose the hairpin completion.
S2.3. Remove all literal strings that contain the complement of α.
S3. All the literal strings at this step encode solutions to the input formula.

As one can see, this modification seems to simplify the implementation of the algorithm because the laborious step 3 in the initial algorithm, which is based on a rather complicated la procedures, namely either enzymatic digestion of the hairpin DNA molecules or exclusive PCR, is avoided and replaced by a sequence of a rather standard lab procedure which removes all single stranded DNA molecules that contain a given sequence. In the case of hairpin lengthening, this lab method could be gel electrophoresis.

4 Non-iterated Hairpin Formation and Semilinearity

A language is semilinear if its Parikh map is a semilinear set [24]. More formally, a subset S of $\mathbb{N}^k$ (k-tuples of natural numbers) is a linear set if there exist vectors $v_0, v_1, \ldots, v_n \in \mathbb{N}^k$ such that $S = \{v_0 + \sum_{i=1}^{n} x_i v_i | x_i \in \mathbb{N}, 1 \leq i \leq n\}$. A finite union of linear sets is called a semilinear set. Let $V = \{a_1, \ldots, a_k\}$ be an alphabet. The Parikh mapping of w is the vector $\psi(w) = (|w|_{a_1}, \ldots, |w|_{a_k})$, which is extended to languages, by $\psi(L) = \{\psi(w) | w \in L\}$. A language is semilinear if its Parikh

mapping is a semilinear set. Two languages are said to be letter-equivalent if for every word in one language there exist at least one anagram of that word in the other language. Clearly, two letter-equivalent languages have the same Parikh mapping. It is known that a language L is semilinear if and only if it is letter-equivalent to a regular language [24]. Furthermore, a language family is called semilinear if all the languages in the family are semilinear. We denote by **SLin** the class of semilinear languages.

Theorem 1

1. **SLin** *is closed neither under* HC_k *nor under* HL_k*, for any* $k \geq 1$.

2. **SLin** *is not closed under* pHC_k*, for any* $k \geq 1$, $p \geq 2$.

Proof. Since our main source of inspiration is the DNA biochemistry, we give counterexamples that use at most 4-letter alphabets.

1. Let $k \geq 1$ and take the semilinear language

$$L_1 = \{ba^{2^n}b^k a^m \overline{b^k} \mid n, m \geq 1\},$$

where both a and $\overline{a}$ are different from b. It is easy to note that $HC_k(L_1) = \{ba^{2^n}b^k a^m \overline{b^k}\overline{a^{2^n}}\overline{b} \mid n, m \geq 1\}$, which is not semilinear. This follows from the fact that semilinear sets are closed under projection, and the projection of $\psi(HC_k(L_1))$ on $\overline{a}$ is not semilinear.
Furthermore,

$$\begin{aligned} HL_k(L_1) =& \{ba^{2^n}b^k a^m \overline{b^k}\overline{a^r} \mid n, m \geq 1, 1 \leq r \leq 2^n\} \cup \\ & \{ba^{2^n}b^k a^m \overline{b^k}\overline{a^{2^k}}\overline{b} \mid n, m \geq 1\}. \end{aligned}$$

By a similar reason as above, $HL_k(L_1)$ cannot be semilinear. We prove that the projection of $\psi(HL_k(L_1))$ on the letters $\overline{a}$ and $\overline{b}$ is not semilinear. This projection is the set

$$X = \{(t, k) \mid t \geq 1\} \cup \{(2^t, k+1) \mid t \geq 1\}.$$

As the class of semilinear sets is closed under intersection [12], if X were semilinear, then the set $X \cap \{(t, k+1) \mid t \geq 1\}$ would be semilinear and this is not the case.

2. We take the semilinear language $L_2 = \{ba^t b^k a^{2^n} \overline{b^k} \mid n, t \geq 1\}$. For every $p \geq 2$, $pHC_k(L_2) = \{ba^r b^k a^{2^n} \overline{b^k}\overline{a^r}\overline{b} \mid n \geq 1, 1 \leq r < p\}$, which is not semilinear. This statement is supported by the fact that the projection of $\psi(pHC_k(L_2))$ on a is $X = \{r + 2^n \mid 1 \leq r < p, n \geq 1\}$, which is not semilinear.

Assume that X is semilinear, hence X is a finite union of linear sets. At least one set in this union, say Y must be infinite. Assume that Y is a linear set with q periods, that is there are the natural numbers $c_1, c_2, \ldots, c_q$, for some $q \geq 1$, such that

$$Y = \{c_o + \sum_{j=1}^{q} x_j c_j \mid x_j \in \mathbb{N}\}.$$

It is known that any linear set is the union of a finite set and a linear set with one period [26]. Therefore, $Y = Z \cup \{c_0 + xy \mid x \in \mathbb{N}\}$. Now, let n_1 be a large number and $1 \leq r_1 < p$ such that $r_1 + 2^{n_1} = c_0 + xy$, for some x. Clearly, there exist $1 \leq r_2 < p$ and n_2 such that $r_2 + 2^{n_2} = c_0 + (x + p)y$. It follows that $2^{n_1}(2^{n_2-n_1} - 1) + r_2 - r_1 = py$, which is a contradiction by the choice of n_1. □

Although the whole class **SLin** is closed under none of the hairpin completion, bounded hairpin completion, and hairpin lengthening, subclasses of **SLin** with further closure properties are closed.

A shuffle of two words is an arbitrary interleaving of subwords of these words such that it contains all letters of both words, like shuffling two decks of cards. This is a well-known language-theoretic operation with a long history in theoretical computer science, in particular within formal languages. Formally, the shuffle operation denoted by $\|\!|\!|$ is defined recursively for two words as follows:

$$x \,\|\!|\!|\, \varepsilon = \varepsilon \,\|\!|\!|\, x = \{x\},\ x \in V^*, \text{ and}$$
$$ax \,\|\!|\!|\, by = a(x \,\|\!|\!|\, by) \cup b(ax \,\|\!|\!|\, y),\ a, b \in V,\ x, y \in V^*.$$

The shuffle of two languages $L_1, L_2 \subseteq V^*$ is denoted by $L_1 \,\|\!|\!|\, L_2$ and is defined as the language consisting of all words that are a shuffle of a word from L_1 and a word from L_2. Thus

$$L_1 \,\|\!|\!|\, L_2 = \{\, w \in u \,\|\!|\!|\, v \mid u \in L_1,\ v \in L_2 \,\}.$$

A restrictive variant of this operation that can be applied to equal length words only, called literal shuffle, is defined by $x \wr\!\wr\, y = a_1 b_1 a_2 b_2 \ldots a_n b_n$, provided that $x = a_1 a_2 \ldots a_n$, $y = b_1 b_2 \ldots b_n$, for some $n \geq 1$, and $a_i, b_i \in V, 1 \leq i \leq n$.

Theorem 2. *Let $\mathcal{F}$ be a family of semilinear languages.*

1. *If $\mathcal{F}$ is closed under intersection and shuffle with regular languages, then both families $HC_k(\mathcal{F})$ and $HL_k(\mathcal{F})$ contain semilinear languages only, for any $k \geq 1$.*
2. *If $\mathcal{F}$ is closed under intersection with regular languages, then all languages in $pHC_k(\mathcal{F})$ are semilinear for all $p, k \geq 1$.*

Proof. 1. The proof of the first two statements is based on a similar idea to that used in [19]. We take a semilinear language $L \subseteq V^*$ in $\mathcal{F}$ and define the next two languages:

$$L_1 = \{(\gamma \wr\!\wr\, \overline{\gamma})\alpha\beta\overline{\alpha}, \alpha, \beta, \gamma \in V^+, |\alpha| = k, \gamma\alpha\beta\overline{\alpha} \in L\},$$
$$L_2 = \{\mu(\delta \wr\!\wr\, \overline{\delta})\alpha\beta\overline{\alpha}, \alpha, \beta \in V^+, |\alpha| = k, \mu\delta\alpha\beta\overline{\alpha} \in L\}.$$

□

Claim 1: $HC_k(L) \in \mathbf{SLin}$.

It is known that the class of semilinear languages is closed under union, hence it suffices to prove that $HCS_k(L)$ is semilinear. Clearly, this language is letter-equivalent to the language L_1. We consider a new alphabet U, disjoint of V, and define a one-to-one mapping $\sigma : V \longrightarrow U$; moreover let $\theta : (V \cup U)^* \longrightarrow V$ be a morphism such that $\theta(a) = \begin{cases} a, & \text{if } a \in V, \\ \overline{b}, & \text{if } a = \sigma(b). \end{cases}$

Now, L_1 can be written as $L_1 = \bigcup_{|\alpha|=k} \theta((L \,|||\, U^*) \cap \{a\sigma(a) \mid a \in V\}^+\{\alpha\}V^+\{\overline{\alpha}\})$.

By the closure properties of $\mathcal{F}$, the language $L \,|||\, U^*$ is in $\mathcal{F}$, moreover, $(L \,|||\, U^*) \cap \{a\sigma(a) \mid a \in V\}^+\{\alpha\}V^+\{\overline{\alpha}\}$ still belongs to $\mathcal{F}$ as each language $\{a\sigma(a) \mid a \in V\}^+\{\alpha\}V^+\{\overline{\alpha}\}$ is regular for any $|\alpha| = k$. For the class of semilinear languages is closed under non-erasing morphisms, it follows that L_1 is a finite union of semilinear languages, which verifies the first claim of the proof.

Claim 2: $HL_k(L) \in \mathbf{SLin}$.

The language $HLS_k(L)$, which is defined analogously to $HCS_k(L)$, is now letter-equivalent to the language L_2. Further on,

$$L_2 = \bigcup_{|\alpha|=k} \theta((L \,|||\, U^*) \cap V^*\{a\sigma(a) \mid a \in V\}^+\{\alpha\}V^+\{\overline{\alpha}\}),$$

which concludes the proof of the second claim.

2. Let $L \subseteq V^*$ be a semilinear language in $\mathcal{F}$, and $p, k \geq 1$. As in the previous case, it suffices to prove that $pHCP_k(L)$ is semilinear. To this aim, for every x with $1 \leq |x| \leq p$, we consider the language $L(x) = L \cap (\bigcup_{|\alpha|=k} \{\alpha\}V^+\{\overline{\alpha}x\})$. Since $\mathcal{F}$ is closed under intersection with regular sets, it follows $L(x) \in \mathcal{F}$ holds. Now, $pHCP_k(L) = \bigcup_{1\leq|x|\leq p} \{\overline{x}\}L(x)$, which implies that $pHCP_k(L)$ is semilinear.

As a direct consequence of this theorem and of Parikh Theorem [24] we have:

Corollary 1. *The hairpin completion, hairpin lengthening, and bounded hairpin completion of every context-free language is semilinear.*

In the proof of the previous theorem, we have used the new alphabet U, hence the construction does not work if one wants to stay within the DNA alphabet with four letters. However, a similar result holds under related requirements. A *generalized sequential machine* (GSM for short) is a device that has finitely many states and each transition is defined as follows: it reads one input letter and outputs 0 or more letters, depending on the current state and the read letter. Every GSM defines a GSM mapping which is a function that associates a finite

set of words with each input word. The GSM mapping of a language is defined in the standard way. A family of languages $\mathcal{F}$ is closed under GSM mapping if the GSM mapping of each language in $\mathcal{F}$ lies in $\mathcal{F}$.

Theorem 3. *If $\mathcal{F}$, a family of semilinear languages, is closed under GSM mappings, then all families $HC_k(\mathcal{F})$, $HL_k(\mathcal{F})$, and $pHC_k(\mathcal{F})$ contain semilinear languages only, for any $k \geq 1$.*

Proof. We give an informal explanation for $HC_k(\mathcal{F})$; the small differences for the other two cases can be easily set by the reader. We define a GSM M that follows the next stages:

Stage 1. For each read letter a, M outputs $a\bar{a}$. Nondeterministically, M passes to the next stage. Note that M reads at least one letter in Stage 1.

Stage 2. This stage is divided into two substages. In the first one, M reads k letters, outputs them and store them as a word into its internal memory. In the second substage, M outputs each read letter. At least one letter is read during the second substage. Nondeterministically, M passes to Stage 3.

Stage 3. M reads k letters, outputs them and checks whether these letters form a word that is the complement of the stored word. If this is the case, M enters a final state otherwise, it enters an error state and halts in both cases.

It is not difficult to note that, given a language L in $\mathcal{F}$, the GSM mapping of L is letter-equivalent to $HCS_k(L)$, which concludes the proof. □

5 Iterated Hairpin Formation and Semilinearity

We consider that a similar investigation of the iterated hairpin completion, hairpin lengthening, and bounded hairpin completion could be of interest for the reader. The iterated version of the hairpin completion is defined as usual by:

$$HC_k^0(w) = \{w\},\ HC_k^{n+1}(w) = HC_k(HC_k^n(w)),\ HC_k^*(w) = \bigcup_{n\geq 0} HC_k^n(w)$$
$$HC^0(w) = \{w\},\ HC^{n+1}(w) = HC(HC^n(w)),\quad HC^*(w) = \bigcup_{n\geq 0} HC^n(w)$$

$$HC_k^*(L) = \bigcup_{w\in L} HC_k^*(w) \qquad HC^*(L) = \bigcup_{w\in L} HC^*(w).$$

In a similar way one can define the iterated hairpin lengthening as well as the bounded hairpin completion. The examples presented in Theorem 1 can be modified to prove the following statement.

Theorem 4. SLin *is closed neither under HC_k^* nor under pHC_k^*, for any $k \geq 3$ and $p \geq 2$.*

Proof. The modifications are as follows:

$$L_1 = \{ba^{2^n}b^k\bar{b}a^m b\overline{b^k} \mid n, m \geq 1\}, \qquad L_2 = \{ba^t b^s (ab)^{2^n}\overline{b^k} \mid n, t \geq 1, s \geq k\}.$$

The proof is based on the fact that the iteration is blocked after the first step. More precisely, $HC_k^*(L_1) = L_1 \cup HC_k(L_1)$ and $pHC_k^*(L_2) = L_2 \cup pHC_k(L_2)$. As $HC_k(L_1) = \{ba^{2^n}b^k\overline{b}a^m b\overline{b^k a^{2^n} b} \mid n, m \geq 1\}$, it follows that $HC_k(HC_k(L_1)) = \emptyset$. The projection of $HC_k^*(L_1)$ on $\overline{a}$ is not semilinear.

On the other hand, as $pHC_k(L_2) = \{ba^t b^s (ab)^{2^n} \overline{b^{k+q}} \overline{a^t b} \mid n \geq 1, 0 \leq q \leq s-k, t+q+1 \leq p\}$. Again, $pHC_k(pHC_k(L_2)) = \emptyset$. If the alphabet of $pHC_k^*(L_2)$ is $\{a, \overline{a}, b, \overline{b}\}$, then the set

$$\psi(pHC_k(pHC_k(L_2))) \cap \{(n_1, n_2, n_3, n_4 \mid n_i \geq 1, 1 \leq i \leq 4\}$$

is not semilinear as its projection on the first component is not semilinear. □

Clearly, the statement is valid for any $k \geq 1$, as soon as alphabets with more than four letters are used. We do not know whether **SLin** is closed under HL_k^*.

We now consider the iterated hairpin completion of singletons. Two out of the three cases have been completely solved. More precisely,

Theorem 5. *Let w be a word and $k, p \geq 1$.*

1. *The iterated k-hairpin lengthening of w is regular, hence semilinear* [21].
2. *The iterated p-bounded k-hairpin completion of w is regular, hence semilinear* [14].

The case of iterated unbounded hairpin completion is open. It is known that the iterated unbounded hairpin completion of a single word can be even non-context-free [17], but we are not aware of any result about the semilinearity of this language. It is worth noting that the iterated unbounded hairpin completion of the word used in [17], namely

$$w = a^k b a^k \overline{a^k} a^k c \overline{a^k},$$

is a semilinear language, though it is not context-free. We strongly conjecture that the iterated unbounded hairpin completion of a single word is always semilinear. It is worth noting that there have been reported necessary and sufficient conditions such that the iterated unbounded hairpin completion of a single word with a special structure, namely a crossing $(2, 2)$-word, is a regular language [29]. The fact that the iterated p-bounded k-hairpin completion of a singleton is semilinear turns out to be useful for proving a more general result.

Theorem 6. *Let $p, k \geq 1$ and $\mathcal{F}$ be a class of semilinear languages closed under intersection with regular languages. Then $pHC_k^*(\mathcal{F}) \in$ **SLin**.*

Proof. We take the language $L \in \mathcal{F}$ over the alphabet V. The language L can be written as the union of two languages L_1 and L_2, where

- L_1 contains all words of L shorter than $2(k+p)+1$.
- L_2 contains all words of L of a length at least $2(k+p)+1$.

The relation $pHC_k^*(L) = pHC_k^*(L_1) \cup pHC_k^*(L_2)$ is immediate. As the language L_1 is finite and the iterated bounded hairpin completion of every singleton is semilinear, it follows that $pHC_k^*(L_1)$ is semilinear. It remains to prove that $pHC_k^*(L_2)$ is also semilinear.

We first note that $L_2 \in \mathcal{F}$ as it can be obtained from L by intersection with a regular language. Second, let u, v be arbitrary words over V of length $k+p$. We set $L_2(u,v) = L_2 \cap \{u\}V^+\{v\}$. By the closure properties of $\mathcal{F}$, each language $L_2(u,v)$ belongs to $\mathcal{F}$ as well. As $L_2 = \bigcup_{|u|=|v|=k+p} L_2(u,v)$ it follows that $pHC_k^*(L_2) = \bigcup_{|u|=|v|=k+p} pHC_k^*(L_2(u,v))$. Every language $pHC_k^*(L_2(u,v))$ can be expressed as $pHC^*(L_2(u,v)) = \sigma(pHC_k^*(uZv))$, where Z is a letter that does not belong to V and σ is a substitution $\sigma : (V \cup \{Z\})^* \longrightarrow 2^{V^*}$ defined by:

$$\sigma(a) = \begin{cases} \{a\}, \text{ if } a \in V, \\ \{z \in V^+ \mid uzv \in L_2(u,v)\}, \text{ if } a = Z. \end{cases}$$

As one can easily see, each substitution σ as above is a substitution with semilinear languages which preserves the semiliniarity [11], therefore each language $pHC_k^*(L_2(u,v))$ is semilinear. In conclusion, $pHC_k^*(L_2)$ is semilinear as well, which concludes the proof. □

As we have mentioned above, the iterated bounded hairpin completion and the iterated hairpin lengthening of every singleton language are semilinear sets. Based on the constructive proofs of these results we may have the following brief discussion.

Let $p, k \geq 1$, w be a word over some alphabet V of cardinality n, and $X \subseteq \mathbb{N}^n$ be a semilinear set. Then the problems:

(i) Does $\psi(HL_k^*(w)) = X$ hold?
(ii) Does $\psi(pHC_k^*(w)) = X$ hold?

are decidable. Both are odd consequences of the following facts. From the proof in [21], it is possible to effectively construct a finite automaton that recognizes $HL_k^*(w)$, hence a regular expression for this language. From the regular expression one can construct the semilinear set $\psi(HL_k^*(w))$ following the original Parikh's proof [24] as well as some later variants [1,13]. A more recent work dealing with the complexity of this transformation is [16]. Thus, if $HL_k^*(w)$ is recognized by a nondeterministic finite automaton with n states, $\psi(HL_k^*(w))$ can be constructed in polynomial time in n (but exponential in $card(V)$). As the equivalence of semilinear sets is decidable, see, e.g., [23], the statement (i) follows. An analogous reasoning works for the second statement as well. Obviously, the algorithm described here has a huge complexity. Is it possible to decrease this complexity by considering another approach?

6 Concluding Remarks

Starting from the observation that the class of semilinear languages is not closed under any of the non-iterated or iterated hairpin operations, we have given some sufficient conditions for subclasses of semilinear languages to preserve this property after applying the hairpin operations. It would be of interest to find other sufficient and/or necessary conditions. Another possible and attractive way to continue this investigation might be to extend the investigation from Sect. 4 to other classes of languages.

References

1. Blattner, M., Latteux, M.: Parikh-bounded languages. In: Even, S., Kariv, O. (eds.) ICALP 1981. LNCS, vol. 115, pp. 316–323. Springer, Heidelberg (1981). https://doi.org/10.1007/3-540-10843-2_26
2. Bottoni, P., Labella, A., Manca, V., Mitrana, V.: Superposition based on Watson-Crick-like complementarity. Theory Comput. Syst. **39**, 503–524 (2006)
3. Castellanos, J., Mitrana, V.: Some Remarks on Hairpin and Loop Languages, Words, Semigroups, and Translations, pp. 47–59. World Scientific, Singapore (2001)
4. Chen, H.-L., Doty, D., Soloveichik, D.: Deterministic function computation with chemical reaction networks. Nat. Comput. **13**, 517–534 (2014)
5. Cheptea, D., Martin-Vide, C., Mitrana, V.: A new operation on words suggested by DNA biochemistry: hairpin completion. In: Proceedings of Transgressive Computing, pp. 216–228 (2006)
6. Csuhaj-Varjú, E., Petre, I., Vaszil, G.: Self-assembly of strings and languages. Theor. Comput. Sci. **374**, 74–81 (2007)
7. Deaton, R., Murphy, R., Garzon, M., Franceschetti, D.R., Stevens, S.E.: Good encodings for DNA-based solutions to combinatorial problems. In: Proceedings of DNA-Based Computers II, DIMACS Series, vol. 44, pp. 247–258 (1998)
8. Enaganti, S.K., Ibarra, O.H., Kari, L., Kopecki, S.: On the overlap assembly of strings and languages. Nat. Comput. **16**, 175–185 (2017)
9. Garzon, M., Deaton, R., Neathery, P., Murphy, R.C., Franceschetti, D.R., Stevens, E.: On the encoding problem for DNA computing. In: Proceedings of Third DIMACS Workshop on DNA-Based Computing, University of Pennsylvania, pp. 230–237 (1997)
10. Garzon, M., Deaton, R., Nino, L.F., Stevens Jr., S.E., Wittner, M.: Genome encoding for DNA computing. In: Proceedings of Third Genetic Programming Conference, Madison, MI, 1998, pp. 684–690 (1998)
11. Ginsburg, S.: AFL with the semilinear property. J. Comput. Syst. Sci. **5**, 365–396 (1971)
12. Ginsburg, S., Spanier, E.H., Henry, E.: Semigroups, Presburger formulas, and languages. Pac. J. Math. **16**, 285–296 (1966)
13. Goldstine, J.: A simplified proof of Parikh's theorem. Discrete Math. **19**, 235–239 (1977)
14. Ito, M., Leupold, P., Manea, F., Mitrana, V.: Bounded hairpin completion. Inf. Comput. **209**, 471–485 (2011)

15. Kari, L., Konstantinidis, S., Sosík, P., Thierrin, G.: On hairpin-free words and languages. In: De Felice, C., Restivo, A. (eds.) DLT 2005. LNCS, vol. 3572, pp. 296–307. Springer, Heidelberg (2005). https://doi.org/10.1007/11505877_26
16. Kopczyński, E., To, A.W. : Parikh images of grammars: complexity and applications. In: Proceedings of 25th Annual IEEE Symposium on Logic in Computer Science (LICS), 2010, pp. 80–89 (2010)
17. Kopecki, S.: On the iterated hairpin completion. Theor. Comput. Sci. **412**, 3629–3638 (2011)
18. Manea, F., Martín-Vide, C., Mitrana, V.: On some algorithmic problems regarding the hairpin completion. Discrete Appl. Math. **157**, 2143–2152 (2009)
19. Manea, F., Mitrana, V., Yokomori, T.: Two complementary operations inspired by the DNA hairpin formation: completion and reduction. Theor. Comput. Sci. **410**, 417–425 (2009)
20. Manea, F., Mitrana, V., Sempere, J.M.: Some remarks on superposition based on Watson-Crick-like complementarity. In: Diekert, V., Nowotka, D. (eds.) DLT 2009. LNCS, vol. 5583, pp. 372–383. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02737-6_30
21. Manea, F., Mercas, R., Mitrana, V.: Hairpin lengthening and shortening of regular languages. In: Bordihn, H., Kutrib, M., Truthe, B. (eds.) Languages Alive. LNCS, vol. 7300, pp. 145–159. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31644-9_10
22. Manea, F., Martín-Vide, C., Mitrana, V.: Hairpin lengthening: language theoretic and algorithmic results. J. Log. Comput. **25**, 987–1009 (2015)
23. Oppen, D.: A $2^{2^{2^{pn}}}$ upper bound on the complexity of Presburger arithmetic. J. Comput. Syst. Sci. **16**, 323–332 (1978)
24. Parikh, R.: On context-free languages. J. ACM **13**, 570–581 (1966)
25. Păun, G., Rozenberg, G., Yokomori, T.: Hairpin languages. Intern. J. Found. Comp. Sci. **12**, 837–847 (2001)
26. Rosales, J.C., García-Sánchez, P.A.: Numerical Semigroups. Springer-Verlag, New-York (2009). https://doi.org/10.1007/978-1-4419-0160-6
27. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages. Springer-Verlag, Berlin (1997). https://doi.org/10.1007/978-3-642-59136-5
28. Sakamoto, K., Gouzu, H., Komiya, K., Kiga, D., Yokoyama, S., Yokomori, T., Hagiya, M.: Molecular computation by DNA hairpin formation. Science **288**, 1223–1226 (2000)
29. Shikishima-Tsuji, K.: Regularity of iterative hairpin completions of crossing $(2, 2)$-words. Int. J. Found. Comput. Sci. **27**, 375–390 (2016)
30. Winfree, E., Yang, X., Seeman, N.C.: Universal computation via self-assembly of DNA: some theory and experiments. In: DNA Bsed Computers II, vol. 44 of DIMACS (1999), pp. 191–213 (1999)