Complexity Results and the Growths of Hairpin Completions of Regular Languages (Extended Abstract)

Volker Diekert and Steffen Kopecki

Universität Stuttgart, FMI
Universitätsstr. 38, 70569 Stuttgart, Germany
{diekert,kopecki}@fmi.uni-stuttgart.de

Abstract. The hairpin completion is a natural operation on formal languages which has been inspired by molecular phenomena in biology and by DNA-computing. In 2009 we presented in [6] a (polynomial time) decision algorithm to decide regularity of the hairpin completion. In this paper we provide four new results: 1.) We show that the decision problem is **NL**-complete. 2.) There is a polynomial time decision algorithm which runs in time $\mathcal{O}(n^8)$, this improves [6], which provided $\mathcal{O}(n^{20})$. 3.) For the one-sided case (which is closer to DNA computing) the time is $\mathcal{O}(n^2)$, only. 4.) The hairpin completion is unambiguous linear context-free. This result allows to compute the growth (generating function) of the hairpin completion and to compare it with the growth of the underlying regular language.

1 Introduction

The hairpin completion is a natural operation of formal languages which has been inspired by molecular phenomena in biology and by DNA-computing. An intramolecular base pairing, known as a *hairpin*, is a pattern that can occur in single-stranded DNA and, more commonly, in RNA. Hairpin or hairpin-free structures have numerous applications to DNA computing and molecular genetics, see [5, 8, 9, 13, 14] and the references within. For example, an instance of 3-Sat has been solved with a DNA-algorithm and one of the main concepts was to eliminate all molecules with a hairpin structure, see [18].

In this paper we study the hairpin completion from a purely formal language viewpoint. The hairpin completion of a formal language was first defined in [4]; here we use a slightly more general definition which was introduced in [6]. The formal operation of the hairpin completion on words is best explained in Fig. 1. In that picture as in the rest of the paper we mean by putting a *bar* on a word (like $\overline{\alpha}$) to read it from right-to-left in addition to replacing a with the *Watson-Crick complement* $\overline{a}$ for letters. The hairpin completion of a regular language is linear context-free [4]. For some time it was not known whether regularity of the hairpin completion is decidable. It was only in 2009 when we presented in [6] a decision algorithm. The runtime of that algorithm is in $\mathcal{O}(n^{20})$, hence polynomial.

M. Domaratzki and K. Salomaa (Eds.): CIAA 2010, LNCS 6482, pp. 105–114, 2011.

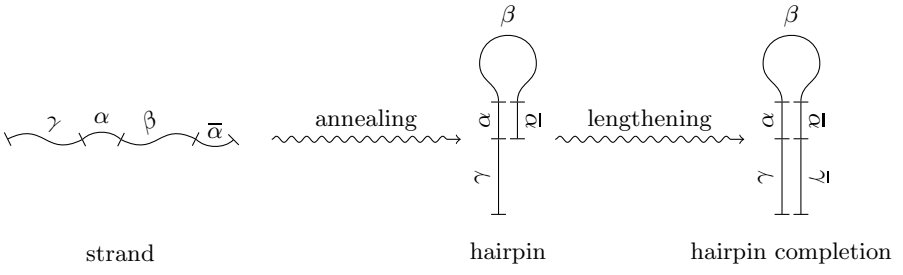

Fig. 1. Hairpin completion of a DNA-strand (or a word)

Here we present a modified approach to solve the decision problem. The new approach leads to improved complexity results and a new structure theorem. We show that the decision problem is **NL**-complete (Thm. 1). We show that there is a polynomial time decision algorithm which runs in time $\mathcal{O}(n^8)$ (Thm. 2, i.). So, the improvement is from $\mathcal{O}(n^{20})$ down to $\mathcal{O}(n^8)$. Moreover, in the biological model the one-sided hairpin completion is of particular interest, and in that special case we need quadratic time, only (Thm. 2, iii.). We also argue why the time bounds might be optimal in the worst case.

A byproduct of the method yields that the hairpin completion of a regular language is unambiguous linear context-free (Thm. 4). The result about unambiguity allows to compute the growth (generating function) of the hairpin completion and to compare it with the growth of the underlying regular language (Thm. 3 and Cor. 1).

This takes us back to a challenging open problem in formal languages. Regularity of linear context-free languages is undecidable in general [1, 11]. But the situation for unambiguous context-free languages is open for more than 40 years. Hence, we have now a positive result within the classical context of deciding regularity within a class of unambiguous (linear) context-free languages.

Due to page limitation, some of the proofs have been removed. The missing proofs can be found in the technical report [7].

2 Preliminaries and Notation

We assume the reader to be familiar with the fundamental concepts of formal language theory, automata theory, and complexity theory, see [12,17]. By **NL** we mean the complexity class NLOGSPACE, which contains the problems which can be decided with a non-deterministic algorithm using $\mathcal{O}(\log n)$ space. We heavily rely on the well-known result that **NL** is closed under complementation. We also use the fact that if L can be reduced to L' via some single-valued non-deterministic transduction in $\mathcal{O}(\log n)$ space and $L' \in \mathbf{NL}$, then we have $L \in \mathbf{NL}$, too. This reduction is performed by a non-deterministic log-space Turing machine. In case the machine stops on input w, the output is always the same, independently of non-deterministic moves during the computation. So, we can

call the output $r(w)$. The reduction property tells us $w \in L$ if and only if both, the machine sometimes stops on input w and $r(w) \in L'$.

By Σ we denote a finite alphabet with at least two letters which is equipped with an *involution* $\overline{\ } : \Sigma \to \Sigma$. An involution for a set is a bijection such that $\overline{\overline{a}} = a$. We extend the involution to words $a_1 \cdots a_n$ by $\overline{a_1 \cdots a_n} = \overline{a_n} \cdots \overline{a_1}$. (Just like taking inverses in groups.) For languages $\overline{L}$ denotes the set $\{\overline{w} \mid w \in L\}$. The set of words over Σ is denoted Σ^*; and the *empty word* is denoted by 1. Given a word w, we denote by $|w|$ its length and $w(m) \in \Sigma$ its m-th letter. By $\Sigma^{\leq m}$ we mean the set of all words with length at most m. If $w = xyz$ for some $x, y, z \in \Sigma^*$, then x and z are called *prefix* and *suffix*, respectively. The prefix relation between words x and w is denoted by $x \leq w$.

Throughout the paper L_1, L_2 mean two regular languages in Σ^* and by k we mean a (small) constant, say $k = 10$. We define the *hairpin completion* $\mathcal{H}_k(L_1, L_2)$ by

$$\mathcal{H}_k(L_1, L_2) = \{\gamma\alpha\beta\overline{\alpha}\overline{\gamma} \mid (\gamma\alpha\beta\overline{\alpha} \in L_1 \vee \alpha\beta\overline{\alpha}\overline{\gamma} \in L_2) \wedge |\alpha| = k\}\,.$$

Three cases are of main interest: 1.) $L_1 = L_2$, 2.) $L_1 = \overline{L_2}$, and 3.) $L_1 = \emptyset$ or $L_2 = \emptyset$. Compared to the definition of the hairpin completion in [4, 16] case 1.) corresponds to the the two-sided hairpin completion and case 3.) to the one-sided hairpin completion. Since we have better time complexities for 2.) and 3.) than for 1.) or in the general case we make the time bounds rather precise.

Regular languages can be specified by non-deterministic finite automata (NFA) $\mathcal{A} = (\mathcal{Q}, \Sigma, E, \mathcal{I}, \mathcal{F})$, where $\mathcal{Q}$ is the finite set of *states*, $\mathcal{I} \subseteq \mathcal{Q}$ is the set of *initial states*, and $\mathcal{F} \subseteq \mathcal{Q}$ is the set of *final states*. The set E contains labeled *edges* (or *arcs*), it is a subset of $\mathcal{Q} \times \Sigma \times \mathcal{Q}$. For a word $u \in \Sigma^*$ we write $p \xrightarrow{u} q$, if there is a path from state p to q which is labeled by the word u. Thus, the accepted language becomes

$$L(\mathcal{A}) = \left\{ u \in \Sigma^* \;\middle|\; \exists p \in \mathcal{I},\, \exists q \in \mathcal{F} :\; p \xrightarrow{u} q \right\}.$$

Later it will be crucial to use also paths which avoid final states. For this we introduce a special notation. First remove all arcs (p, a, q) where $q \in \mathcal{F}$ is a final state. Thus, final states do not have incoming arcs anymore in this reduced automaton. Let us write $p \overset{u}{\Longrightarrow} q$, if there is a path in this reduced automaton from state p to q which is labeled by the word u. Note that for such a path $p \overset{u}{\Longrightarrow} q$ we allow $p \in \mathcal{F}$, but on the path we never meet any final state again.

An NFA is called a deterministic finite automaton (DFA), if it has one initial state and for every state $p \in \mathcal{Q}$ and every letter $a \in \Sigma$ there is exactly one arc $(p, a, q) \in E$. In particular, a DFA in this paper is always complete, thus we can read every word to its end. We also write $p \cdot u = q$, if $p \xrightarrow{u} q$. This yields a (totally defined) function $\mathcal{Q} \times \Sigma^* \to \mathcal{Q}$, which defines an action of Σ^* on $\mathcal{Q}$ on the right.

In the following we need a DFA accepting L_1 as well as a DFA accepting L_2, but the DFA for L_2 has to work from right-to-left. Instead of introducing this concept we use a DFA (working as usual from left-to-right), which accepts $\overline{L_2}$.

This automaton has the same number of states (and is structurally isomorphic to) as a DFA accepting the *reversal language* of L_2.

As input we assume that the regular languages L_1 and $\overline{L_2}$ are specified by DFAs with state set $\mathcal{Q}_i$, state $q_{0i} \in \mathcal{Q}_i$ as initial state, and $\mathcal{F}_i \subseteq \mathcal{Q}_i$ as final states. We fix $n_i = |\mathcal{Q}_i|$ to be the number of states, $i = 1, 2$. By n we mean $\max\{n_1, n_2\}$. The input size is therefore the number n.

We also need the usual product DFA with state space

$$\mathcal{Q} = \{(p_1, p_2) \in \mathcal{Q}_1 \times \mathcal{Q}_2 \mid \exists w \in \Sigma^* : (p_1, p_2) = (q_{01} \cdot w,\ q_{02} \cdot w)\}\,.$$

The action is given by $(p_1, p_2) \cdot a = (p_1 \cdot a,\ p_2 \cdot a)$. We let $n_{12} = |\mathcal{Q}|$. Hence, $n \le n_{12} \le n_1 \cdot n_2 \le n^2$, and $n = n_1 = n_{12}$ if $L_2 = \emptyset$ or $L_1 = \overline{L_2}$. In the following we work simultaneously in all three automata defined so far. Moreover, in $\mathcal{Q}_1$ and $\mathcal{Q}_2$ we have to work backwards. This leads to nondeterminism. Our first new construction concerns a special NFA in Section 3.1.

3 Main Results

The complexity results of this paper are the following:

Theorem 1. *The decision problem whether the hairpin completion $\mathcal{H}_k(L_1, L_2)$ is regular is* **NL***-complete.*

Theorem 2. *i.) The problem whether the hairpin completion $\mathcal{H}_k(L_1, L_2)$ is regular can be decided in time $\mathcal{O}(n^8)$.*
ii.) For $L_1 = \overline{L_2}$ it can be decided in time $\mathcal{O}(n^6)$.
iii.) For $L_2 = \emptyset$ it can be decided in time $\mathcal{O}(n^2)$.

An algorithm solving this problem is sketched in Section 3.3. For a proof of the strict time bounds and a proof of the **NL**-hardness we refer to [7].

The *growth* or *generating function* g_L of a formal language L is defined as: $g_L(z) = \sum_{m \ge 0} \left|L \cap \Sigma^{\le m}\right| z^m$. We can view g_L as a formal power series or as an analytic function in one complex variable where the radius of convergence is strictly positive. The radius of convergence is at least $1/|\Sigma|$.

It is well-known that the growth of a regular language L is effectively rational, i.e., a quotient of two polynomials. The same is true for unambiguous linear context-free languages. In particular, the growth is either polynomial or exponential. If the growth is exponential, then we find an algebraic number $\rho \in \mathbb{R}$ such that $\left|L \cap \Sigma^{\le m}\right|$ behaves essentially as ρ^m, see [3, 10, 2].

It was shown in [4] that $\mathcal{H}_k(L_1, L_2)$ is an linear context-free language. As a byproduct to our techniques to prove the complexity results above we find that $\mathcal{H}_k(L_1, L_2)$ is unambiguous, and hence its growth (i.e., generating function) is a rational function, see e.g. [15] for this well-known fact. We obtain:

Theorem 3. *The hairpin completion $\mathcal{H}_k(L_1, L_2)$ is an unambiguous linear context-free language with an effectively computable rational growth function.*

This result is proved in Section 3.2.

3.1 The NFA $\mathcal{A}$

In this section we define a certain NFA which is called simply $\mathcal{A}$. Almost all further results are done by exploring properties of this NFA. The NFA is a sort of product automaton over $\mathcal{Q} \times \mathcal{Q}_1 \times \mathcal{Q}_2 \subseteq \mathcal{Q}_1 \times \mathcal{Q}_2 \times \mathcal{Q}_1 \times \mathcal{Q}_2$ where $\mathcal{Q}_1, \mathcal{Q}_2$ and $\mathcal{Q}$ are defined as in Section 2. The size of this automaton is $\mathcal{O}(n^4)$ in the worst case, and our decision algorithm will take into account all pairs of states in this NFA. Hence, $\mathcal{O}(n^8)$ might be an optimal time bound and the decision algorithm is not worse than quadratic in the size of the NFA $\mathcal{A}$.

For every quadruple $(p_1, p_2, q_1, q_2) \in \mathcal{Q}_1 \times \mathcal{Q}_2 \times \mathcal{Q}_1 \times \mathcal{Q}_2$ we define a regular language $B(p_1, p_2, q_1, q_2)$ as follows:

$$B(p_1, p_2, q_1, q_2) = \{w \in \Sigma^* \mid p_1 \cdot w = q_1 \wedge p_2 \cdot \overline{w} = q_2\}.$$

We say that (p_1, p_2, q_1, q_2) is a *basic bridge* if $B(p_1, p_2, q_1, q_2) \neq \emptyset$. The idea behind of this notation is that $B(p_1, p_2, q_1, q_2)$ closes a gap between pairs (p_1, p_2) and (q_1, q_2) (which are on different sides). For a letter $a \in \Sigma$ we call (p_1, p_2, q_1, q_2) an *a-bridge* if $B(p_1, p_2, q_1, q_2) \cap a\Sigma^* \neq \emptyset$.

Lemma 1. *The number of basic bridges and a-bridges is bounded by $\mathcal{O}(n_1^2 n_2^2)$. A table containing all these bridges can be computed in time $\mathcal{O}(n_1^2 n_2^2) \subseteq \mathcal{O}(n^4)$, and there is a single-valued non-deterministic transduction working in $\mathcal{O}(\log n)$ space which outputs this table.*

Proof. To compute the basic bridges amounts to compute the transitive closure in some graph where the number of nodes and edges is in $\mathcal{O}(n_1 n_2)$. This gives the time bound. Once we have the bridges we can compute the a-bridges in time $\mathcal{O}(n_1^2 n_2^2)$.

If (p_1, p_2, q_1, q_2) is a basic bridge, we can verify this property in **NL**. Since **NL** is closed under complementation, we can output the whole table by a single-valued non-deterministic transduction in $\mathcal{O}(\log n)$ space.

We also need *levels* for $0 \leq \ell \leq k$, hence there are $k+1$ levels. By $[k]$ we denote in this paper the set $\{0, \ldots, k\}$. Define

$$\{((p_1, p_2), q_1, q_2, \ell) \in \mathcal{Q} \times \mathcal{Q}_1 \times \mathcal{Q}_2 \times [k] \mid (p_1, p_2, q_1, q_2) \text{ is a basic bridge}\}$$

as the state space of the NFA $\mathcal{A}$. Its size is bounded by $N \cdot (k+1) \in \mathcal{O}(N) \subseteq \mathcal{O}(n^4)$, where $N = n_{12} n_1 n_2$. We have $N = n^2$ for $L_2 = \emptyset$, and $N = n^3$ for $L_2 = \overline{L_1}$.

By a (slight) abuse of languages we call a state $((p_1, p_2), q_1, q_2, \ell)$ a *bridge*, and we keep in mind that there exists a word w such that $p_1 \cdot w = q_1$ and $p_2 \cdot \overline{w} = q_2$. Bridges are frequently denoted by (P, q_1, q_2, ℓ) with $P = (p_1, p_2) \in \mathcal{Q}$, $q_i \in Q_i$, $i = 1, 2$, and $\ell \in [k]$. Bridges are a central concept in the following.

The a-transitions in the NFA for $a \in \Sigma$ are given by the following arcs:

$$(P,\ q_1 \cdot \overline{a},\ q_2 \cdot \overline{a}, 0) \xrightarrow{a} (P \cdot a,\ q_1, q_2, 0) \qquad \text{for } q_i \cdot \overline{a} \notin \mathcal{F}_i,\ i = 1, 2,$$

$$(P,\ q_1 \cdot \overline{a},\ q_2 \cdot \overline{a}, 0) \xrightarrow{a} (P \cdot a,\ q_1, q_2, 1) \qquad \text{for } q_1 \cdot \overline{a} \in \mathcal{F}_1 \text{ or } q_2 \cdot \overline{a} \in \mathcal{F}_2,$$

$$(P,\ q_1 \cdot \overline{a},\ q_2 \cdot \overline{a}, \ell) \xrightarrow{a} (P \cdot a,\ q_1, q_2, \ell + 1) \qquad \text{for } 1 \leq \ell < k.$$

Observe that no state of the form $(P, q_1, q_2, 0)$ with $q_1 \in \mathcal{F}_1$ or $q_2 \in \mathcal{F}_2$ has an outgoing arc to level zero; we must switch to level one. There are no outgoing arcs on level k, and for each $(a, P, q_1, q_2, \ell) \in \Sigma \times \mathcal{Q} \times \mathcal{Q}_1 \times \mathcal{Q}_2 \times [k-1]$ there exists at most one arc $(P, q_1', q_2', \ell) \xrightarrow{a} (P \cdot a, q_1, q_2, \ell')$. Indeed, the triple (q_1', q_2', ℓ') is determined by (q_1, q_2, ℓ) and the letter a. Not all arcs exist because (P, q_1', q_2', ℓ) can be a bridge whereas $(P \cdot a, q_1, q_2, \ell')$ is not. Thus, there are at most $|\Sigma| \cdot N \cdot k \in \mathcal{O}(N)$ arcs in the NFA.

The set of initial states $\mathcal{I}$ contains all bridges of the form $(Q_0, q_1', q_2', 0)$ with $Q_0 = (q_{01}, q_{02})$. The set of final states $\mathcal{F}$ is given by all bridges (P, q_1, q_2, k) on level k.

Remark 1. The NFA $\mathcal{A}$ can be computed by Lemma 1 in time $\mathcal{O}(n_1^2 n_2^2)$ and by a single-valued non-deterministic transduction in $\mathcal{O}(\log n)$ space. Thus for both the polynomial time and the **NL** algorithm we can have direct access to $\mathcal{A}$ and we can assume that $\mathcal{A}$ is written on the input tape.

The next result shows the unambiguity of paths in the automaton $\mathcal{A}$.

Lemma 2. *Let $w \in \Sigma^*$ be the label of a path in $\mathcal{A}$ from a bridge $A = (P, p_1, p_2, \ell)$ to $A' = (P', p_1', p_2', \ell')$, then the path is unique. This means that $B = B'$ whenever $w = uv$ and*

$$A \xrightarrow{u} B \xrightarrow{v} A', \qquad\qquad A \xrightarrow{u} B' \xrightarrow{v} A'.$$

Proof. It is enough to consider $u = a \in \Sigma$. Let $B = (Q, q_1, q_2, m)$. Then we have $Q = P \cdot a$ and $q_i = p_i' \cdot \overline{v}$. If $\ell = 0$ and $p_i \notin \mathcal{F}_i$ for $i = 1, 2$, then $m = 0$, too. Otherwise $m = \ell + 1$. Thus, B is defined by A, A', and u, v. We conclude $B = B'$.

3.2 Structure Theorem and Rational Growth

For languages U and V we define the language V^U as follows:

$$V^U = \{uv\overline{u} \mid u \in U, v \in V\}.$$

Clearly, if U and V are regular, then V^U is linear context-free. We are interested in a disjoint union of languages V^U where for $w \in V^U$ the factorization $w = uv\overline{u}$ with $u \in U$ and $v \in V$ is unambiguous.

Theorem 4. *Let $T = \mathcal{I} \times \mathcal{F}$. For each $\tau = (I, F) \in T$ with $F = ((d_1, d_2), e_1, e_2, k)$ let R_τ be the (regular) set of words which label a path from the initial bridge I to the final bridge F and let $B_\tau = B(d_1, d_2, e_1, e_2)$. The hairpin completion is a disjoint union*

$$\mathcal{H}_k(L_1, L_2) = \bigcup_{\tau \in T} B_\tau^{R_\tau}.$$

Moreover, for each word in some $w \in B_\tau^{R_\tau}$ there is a unique factorization $w = \rho\beta\overline{\rho}$ with $\rho \in R_\tau$ and $\beta \in B_\tau$.

Corollary 1. *The hairpin completion $\mathcal{H}_k(L_1, L_2)$ is an unambiguous linear context-free language and it has a rational growth function. The growth can be directly calculated by the growth of the regular languages R_τ and B_τ.*

Corollary 1 allows to compare the growth of L_1 and L_2 with the growth of their hairpin completion $\mathcal{H}_k(L_1, L_2)$. It is also a slightly more precise version of Theorem 3.

3.3 Complexity for Testing the Regularity of $\mathcal{H}_k(L_1, L_2)$

First Test. The automaton $\mathcal{A}$ accepts the union of the languages R_τ as defined in Theorem 4. If the accepted language is finite then all R_τ are finite and hence all $B_\tau^{R_\tau}$ are regular. This leads to the following result:

Proposition 1. *i.) If the accepted language of the NFA $\mathcal{A}$ is finite, then the hairpin completion $\mathcal{H}_k(L_1, L_2)$ is regular.*
ii.) If L_1 or L_2 is finite, but the accepted language of $\mathcal{A}$ is infinite, then the hairpin completion $\mathcal{H}_k(L_1, L_2)$ is not regular.

Test 1: Check either by some **NL**-algorithm or in time $\mathcal{O}(N) \subseteq \mathcal{O}(n^4)$ (in time $\mathcal{O}(n^2)$, if L_1 or L_2 is empty) whether the accepted language of the NFA $\mathcal{A}$ is finite.

If "yes" (=$L(\mathcal{A})$ is finite), then output that $\mathcal{H}_k(L_1, L_2)$ is regular. If "no", but L_1 or L_2 is finite, then output that $\mathcal{H}_k(L_1, L_2)$ is not regular.

Second Test. From now on we may assume that the automaton $\mathcal{A}$ accepts an infinite language and both L_1 and L_2 are infinite as well. We assume that all states are reachable from initial bridges and lead to some final bridges. (Recall that graph reachability can be checked in **NL**.)

Let K be the set of non-trivial strongly connected components of the automaton $\mathcal{A}$ (read as a directed graph). For $\kappa \in K$ let $N_\kappa = |\kappa|$ the number of states in the component κ. Let us choose some $A_\kappa \in \kappa$ and some shortest non-empty word $v_\kappa \in \Sigma^+$ such that there is a path in $\mathcal{A}$ labeled by v_κ from A_κ to A_κ.

The next lemma tells us that for a regular hairpin completion $\mathcal{H}_k(L_1, L_2)$ the word v_κ is uniquely defined by A_κ, its length is N_κ, and its conjugacy class depends only on κ.

Lemma 3. *Assume that the hairpin completion $\mathcal{H}_k(L_1, L_2)$ is regular.*
1.) Let $A_\kappa \xrightarrow{v_\kappa} A_\kappa$ as above and $A_\kappa \xrightarrow{w} C$ be a path in $\mathcal{A}$ to some final bridge. Then the word w is a prefix of some word in v_κ^+.
2.) The word v_κ and the loop $A_\kappa \xrightarrow{v_\kappa} A_\kappa$ are uniquely defined by the state A_κ and we have $|v_\kappa| = N_\kappa$.
3.) The loop $A_\kappa \xrightarrow{v_\kappa} A_\kappa$ visits every other state $B \in \kappa$ exactly once. Thus, the loop defines an Hamiltonian cycle of κ.

Remark 2. We decompose the automaton $\mathcal{A}$ in its strongly connected components by the algorithm of Tarjan in time $\mathcal{O}(N)$. (Note that we have $K \neq \emptyset$

since $|L(\mathcal{A})|$ is infinite.) This is also possible by some single-valued non-deterministic transduction. Putting some linear order on the set of bridges, we can assign to each $\kappa \in K$ the least $A_\kappa \in \kappa$. If $\mathcal{H}_k(L_1, L_2)$ is regular, then (by Lemma 3) we can output the uniquely defined words v_κ for all $\kappa \in K$. We observe that $\sum_{\kappa\in K} |v_\kappa| = \sum_{\kappa\in K} N_\kappa \le N$. So, the list of all v_κ is computable in time $\mathcal{O}(N)$ and also by some single-valued non-deterministic transduction, in case $\mathcal{H}_k(L_1, L_2)$ is regular.

Test 2: It has two parts. Part I: For each strongly connected component $\kappa \in K$ compute a shortest word v with $0 < |v| \le N_\kappa$ such that $A_\kappa \xrightarrow{v} A_\kappa$ is a loop in the automaton $\mathcal{A}$. If $|v| \ne N_\kappa$, then **stop** and output that $\mathcal{H}_k(L_1, L_2)$ is not regular. Part II: If $|v| = N_\kappa$ for all κ, then let L_κ be the accepted language of $\mathcal{A}$ when the bridge A_κ is used as initial state. Let $\mathrm{Pref}(v^+)$ be the language of prefixes of words in v^+. (Note that a DFA for the complement of $\mathrm{Pref}(v^+)$ has $N_\kappa + 1$ states.) If we do not find $L_\kappa \subseteq \mathrm{Pref}(v^+)$, then **stop** and output that $\mathcal{H}_k(L_1, L_2)$ is not regular.

Part I can be done in time $\mathcal{O}(\sum_{\kappa\in K} N_\kappa) \subseteq \mathcal{O}(N)$ or in **NL**. Part II can be done in time $\mathcal{O}(\sum_{\kappa\in K} N_\kappa \cdot N) \subseteq \mathcal{O}(N^2) \subseteq \mathcal{O}(n^8)$. The **NL**-algorithm for Part II is based on the fact that we can guess a position m where the m-th letter of $w \in L_\kappa$ differs from the $(m \bmod N_\kappa)$-th letter of v_κ.

Remark 3. Henceforth we may assume that Test 2 was successful and following Remark 2 we assume that the list of all words v_κ is available. Thus, we can think that the list $(v_\kappa; \kappa \in K)$ is written on the input tape. For the **NL**-algorithm we perform another single-valued non-deterministic transduction to achieve this.

Third and Fourth Test. We fix a strongly connected component $\kappa \in K$ of $\mathcal{A}$. We let $A = A_\kappa = ((p_1, p_2), q_1, q_2, 0)$ and $v = v_\kappa$ as above. By u we denote some word leading from an initial bridge $((q_{01}, q_{02}), q_1', q_2', 0)$ to A. (The following tests do not rely on the choice of u.) The main idea is to investigate runs through the DFAs for L_1 and $\overline{L_2}$ where $s, t \ge n$.

$$L_1: \quad q_{01} \xrightarrow{u} p_1 \xrightarrow{v^s} p_1 \xrightarrow{xy} c_1 \xrightarrow{z} d_1 \xrightarrow{\overline{x}} e_1 \xRightarrow{\overline{v}^{n_1}} q_1 \xRightarrow{\overline{v}^*} q_1 \xRightarrow{\overline{u}} q_1'$$

$$\overline{L_2}: \quad q_{02} \xrightarrow{u} p_2 \xrightarrow{v^t} p_2 \xrightarrow{x} c_2 \xrightarrow{\overline{z}} d_2 \xrightarrow{\overline{y}\overline{x}} e_2 \xRightarrow{\overline{v}^{n_2}} q_2 \xRightarrow{\overline{v}^*} q_2 \xRightarrow{\overline{u}} q_2'$$

We investigate the case where $uv^s xyz\overline{x}\overline{v}^t\overline{u} \in \mathcal{H}_k(L_1, L_2)$ for all $s \ge t$ and where (by symmetry) this property is due to the longest prefix belonging to L_1.

The following lemma is the most technical one in our paper.

Lemma 4. *Let $x, y, z \in \Sigma^*$ be words and $(d_1, d_2) \in \mathcal{Q}_1 \times \mathcal{Q}_2$ with the following properties:*

1.) *$k \le |x| < |v| + k$ and x is a prefix of some word in v^+.*
2.) *$0 \le |y| < |v|$ and xy is the longest common prefix of xyz and some word in v^+.*
3.) *$z \in B(c_1, c_2, d_1, d_2)$, where $c_1 = p_1 \cdot xy$ and $c_2 = p_2 \cdot x$.*

4.) $q_1 = d_1 \cdot \overline{x}\overline{v}^{n_1}$ *and during the computation of* $d_1 \cdot \overline{x}\overline{v}^{n_1}$ *we see after exactly* k *steps a final state in* $\mathcal{F}_1$ *and then never again.*

5.) $q_2 = d_2 \cdot \overline{y}\overline{x}\overline{v}^{n_2}$ *and, let* $e_2 = d_2 \cdot \overline{y}\overline{x}$*, during the computation of* $e_2 \cdot \overline{v}^{n_2}$ *we do not see a final state in* $\mathcal{F}_2$*.*

If $\mathcal{H}_k(L_1, L_2)$ *is regular, then* $xyz\overline{x}\overline{v} = \mu\delta\beta\overline{\delta}\overline{\mu}$ *where* $|\delta| = k$ *and* $\delta\beta\overline{\delta}\overline{\mu}\overline{u} \in L_2$*.*

Lemma 5. *The existence of words* $x, y, z \in \Sigma^*$ *and states* $(d_1, d_2) \in \mathcal{Q}_1 \times \mathcal{Q}_2$ *satisfying 1.) to 5.) of Lemma 4, but where for all factorizations* $xyz\overline{x}\overline{v} = \mu\delta\beta\overline{\delta}\overline{\mu}$ *we have* $p_2 \cdot \mu\delta\overline{\beta}\overline{\delta} \notin \mathcal{F}_2$ *(and accordingly* $\delta\beta\overline{\delta}\overline{\mu}\overline{u} \notin L_2$*), can be decided in time* $\mathcal{O}(n_{12}^2 n_1^2 n_2^2) \subseteq \mathcal{O}(n^8)$ *and in* **NL***.*

Proof. It is enough to perform Tests 3, 4 below and to prove the complexity.

The tests distinguish whether the word z is non-empty or empty.

Test 3: Decide the existence of words $x, y, z \in \Sigma^*$ with $z \neq 1$ and states $(d_1, d_2) \in \mathcal{Q}_1 \times \mathcal{Q}_2$ satisfying 1.) to 5.) of Lemma 4, but where for all factorizations $xyz\overline{x}\overline{v} = \mu\delta\beta\overline{\delta}\overline{\mu}$ we have $p_2 \cdot \mu\delta\overline{\beta}\overline{\delta} \notin \mathcal{F}_2$. If we find such a situation, then **stop** and output that $\mathcal{H}_k(L_1, L_2)$ is not regular.

Test 4: Decide the existence of words $x, y \in \Sigma^*$ and states $(d_1, d_2) \in \mathcal{Q}_1 \times \mathcal{Q}_2$ satisfying 1.) to 5.) of Lemma 4 with $z = 1$, but where for all factorizations $xy\overline{x}\overline{v} = \mu\delta\beta\overline{\delta}\overline{\mu}$ we have $p_2 \cdot \mu\delta\overline{\beta}\overline{\delta} \notin \mathcal{F}_2$. If we find such a situation, then **stop** and output that $\mathcal{H}_k(L_1, L_2)$ is not regular.

The correctness of both tests follows by Lemma 4, but even termination of Test 3 is not completely obvious. Termination is due to condition that xy is the longest common prefix of xyz and some word in v^+. This means, if $z \neq 1$, then there exists a letter a such that $z \in a\Sigma^*$ and xya is no prefix of any word in v^+. Now $|y| < |v|$, hence we see that $xyz\overline{x}\overline{v} = \mu\delta\beta\overline{\delta}\overline{\mu}$ implies $\mu\delta \leq xy$.

Thus it is enough to check the computation starting in state $d_2 \in \mathcal{Q}_2$ when reading the word $\overline{y}\overline{x}$. Test 3 is successful if we find such a computation which after more than $k-1$ steps does not meet any final state in $\mathcal{F}_2$. We do not need the word z, we just have to know that (c_1, c_2, d_1, d_2) is in the precomputed table of a-bridges (cf. Lemma 1) where a is a letter such that xya is no prefix of any word in v^+. It is obvious that Test 3 can be performed in polynomial time as well as in **NL**.

Test 4 is for $z = 1$, so in any case the number of factorizations $xy\overline{x}\overline{v} = \mu\delta\beta\overline{\delta}\overline{\mu}$ is polynomial. It is again obvious that Test 4 can be performed polynomial time as well as in **NL**.

The following lemma completes the proof of Theorem 1 and 2.

Lemma 6. *Suppose no outcome of Tests 1, 2, 3, and 4 is "not regular". Then the hairpin completion* $\mathcal{H}_k(L_1, L_2)$ *is regular.*

Acknowledgement

We thank the anonymous referees for many useful remarks and hints.

References

1. Baker, B.S., Book, R.V.: Reversal-bounded multi-pushdown machines. In: Annual IEEE Symposium on Foundations of Computer Science, pp. 207–211 (1972)
2. Berstel, J., Reutenauer, C.: Rational series and their languages. Springer, New York (1988)
3. Ceccherini-Silberstein, T.: On the growth of linear languages. Advances in Applied Mathematics 35(3), 243–253 (2005)
4. Cheptea, D., Martin-Vide, C., Mitrana, V.: A new operation on words suggested by DNA biochemistry: Hairpin completion. Transgressive Computing, 216–228 (2006)
5. Deaton, R., Murphy, R., Garzon, M., Franceschetti, D., Stevens, S.: Good encodings for DNA-based solutions to combinatorial problems. In: Proc. of DNA-Based computers DIMACS Series, vol. 44, pp. 247–258 (1998)
6. Diekert, V., Kopecki, S., Mitrana, V.: On the hairpin completion of regular languages. In: Leucker, M., Morgan, C. (eds.) ICTAC 2009. LNCS, vol. 5684, pp. 170–184. Springer, Heidelberg (2009)
7. Diekert, V., Kopecki, S.: Complexity Result and the Growths of Hairpin Completions of Regular Languages. Technical Report Computer Science 2010/04, University of Stuttgart (June 2010)
8. Garzon, M., Deaton, R., Neathery, P., Murphy, R., Franceschetti, D., Stevens, E.: On the encoding problem for DNA computing. In: The Third DIMACS Workshop on DNA-Based Computing, pp. 230–237 (1997)
9. Garzon, M., Deaton, R., Nino, L., Stevens Jr., S., Wittner, M.: Genome encoding for DNA computing. In: Proc. Third Genetic Programming Conference, pp. 684–690 (1998)
10. Gawrychowski, P., Krieger, D., Rampersad, N., Shallit, J.: Finding the growth rate of a regular or context-free language in polynomial time. In: Ito, M., Toyama, M. (eds.) DLT 2008. LNCS, vol. 5257, pp. 339–358. Springer, Heidelberg (2008)
11. Greibach, S.A.: A note on undecidable properties of formal languages. Mathematical Systems Theory 2(1), 1–6 (1968)
12. Hopcroft, J.E., Ulman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Reading (1979)
13. Kari, L., Konstantinidis, S., Losseva, E., Sosík, P., Thierrin, G.: Hairpin structures in DNA words. In: Carbone, A., Pierce, N.A. (eds.) DNA 2005. LNCS, vol. 3892, pp. 158–170. Springer, Heidelberg (2006)
14. Kari, L., Mahalingam, K., Thierrin, G.: The syntactic monoid of hairpin-free languages. Acta Inf. 44(3-4), 153–166 (2007)
15. Kuich, W.: On the entropy of context-free languages. Information and Control 16, 173–200 (1970)
16. Manea, F., Mitrana, V., Yokomori, T.: Two complementary operations inspired by the DNA hairpin formation: Completion and reduction. Theor. Comput. Sci. 410(4-5), 417–425 (2009)
17. Papadimitriou, C.H.: Computatational Complexity. Addison Wesley, Reading (1994)
18. Sakamoto, K., Gouzu, H., Komiya, K., Kiga, D., Yokoyama, S., Yokomori, T., Hagiya, M.: Molecular Computation by DNA Hairpin Formation. Science 288(5469), 1223–1226 (2000)