Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Theoretical Computer Science 306 (2003) 19–38

Theoretical Computer Science

www.elsevier.com/locate/tcs

Closure and decidability properties of some language classes with respect to ciliate bio-operations[☆]

Mark Daley[a], Oscar H. Ibarra[b,*], Lila Kari[a]

[a] *Department of Computer Science, University of Western Ontario, London, Canada ON N6A 5B7*
[b] *Department of Computer Science, University of California, Santa Barbara, CA 93106, USA*

Received 12 October 2002; received in revised form 8 December 2002; accepted 20 January 2003
Communicated by G. Rozenberg

Abstract

The process of gene unscrambling in ciliates (a type of unicellular protozoa), which accomplishes the difficult task of re-arranging gene segments in the correct order and deleting non-coding sequences from an "encrypted" version of a DNA strand, has been modeled and studied so far from the point of view of the computational power of the DNA bio-operations involved. Here we concentrate on a different aspect of the process, by considering only the linear version of the bio-operations, that do not involve thus any circular strands, and by studying the resulting formal operations from a purely language-theoretic point of view. We investigate closure properties of language families under the mentioned bio-operations and study language equations involving them. We also study the decidability of the existence of solutions to equations of the form $L\diamond Y = R$, $X\diamond L = R$ where L and R are given languages, X and Y are unknowns, and $\diamond$ signifies one of the defined bio-operations.
© 2003 Elsevier B.V. All rights reserved.

1. Introduction

The stichotrichous ciliates are a group of primitive single-celled organisms which have generated a great deal of interest due to their unique genetic mechanisms. Most

[☆] Some of the results in this paper, in less general form, were reported in [2]. Research of Mark Daley and Lila Kari has been supported by Natural Sciences and Engineering Council of Canada Grants. Research of Oscar H. Ibarra has been supported by NSF Grants IIS-0101134 and CCR02-08595.
[*] Corresponding author. Tel.: +1-805-893-4171; fax: +1-805-893-8553.
E-mail address: ibarra@cs.ucsb.edu (O.H. Ibarra).

organisms store their genomic DNA in a linear sequence consisting of coding regions interspersed with non-coding regions. Several ciliate genes, however, are stored in a scrambled form. For example, if a functional copy of a gene consists of the coding regions arranged in the order 1-2-3-4-5, it may appear in the order 3-5-4-1-2 in the genome. This presents an interesting problem for the organism, who must somehow descramble these genes in order to generate functional proteins required for its continued existence. The sizes of the descrambling problems in vivo are nontrivial. The gene encoding *DNA polymerase* α in the ciliate *Stylonichia Lemnae* is broken into more than 48 discreet pieces that must be reassembled in the correct order to produce a functional gene [13].

The details of the biological mechanism underlying the unscrambling process are still unknown. For further information on the biology of the descrambling process in ciliates the reader is referred to [15,16,17]. The two existing formal models for gene unscrambling, by Kari and Landweber [11,12], respectively Ehrenfeucht, Harju, Petre, Prescott and Rozenberg [14,4,18] are consistent with the existing biological data. Each proposes a set of two, respectively three, atomic operations the combination of which can lead to the unscrambling of an arbitrarily scrambled gene. The bio-operations proposed by the first model are circular insertions and deletions, i.e. insertions and deletions of circular strands into/from linear strands, guided by the presence of certain pointers [11,12]. The second model focuses more on the properties of pointers and proposes three operations named based on the biological processes which they describe: hi (hairpin loop with inverted pointers) which reverses a substring between two pointer sequences, ld(loop with direct pointers)-excision which deletes a substring between two pointers and dlad(double loop with alternating direct pointers)-excision/reinsertion which swaps two substrings marked by pointer-pairs. In both cases, the operations presented are based on real biological events that can occur and change a DNA molecule and are capable of descrambling an arbitrarily scrambled ciliate gene. The significance of the differences in the two models serves to underscore that the actual biological process of gene descrambling is only now beginning to be understood. It is thus possible for two relatively disparate models to be consistent with currently available data. This paper does not address the biological aspects and implications of the proposed operations. Instead, we continue in the style of Dassow et al's work on properties of operations inspired by general DNA recombination events [3] and focus on some of their properties as word/language operations. A related study considering generalizations of the operations proposed by Ehrenfeucht, Harju, Petre, Prescott and Rozenberg (different from the ones proposed here) has been undertaken in [5]. We will consider here closure properties of various families of languages under the defined operations. In particular, with the goal of identifying large families of recursive languages that are closed under the bio-operations, we show that the family, NCM, of languages defined by nondeterministic finite automata augmented with reversal-bounded counters (i.e., the counters can be incremented/decremented by 1 and tested for zero but the number of alternations between nondecreasing mode and nonincreasing mode and vice versa is bounded by a fixed constant) [9] is closed under all ciliate bio-operations. It is known [7] that NCM is the smallest class of languages containing the regular sets that is closed under homomorphism, intersection, and the shuffle operation (Section 3 gives the precise definition). We also look at generalizations of NCM (e.g., adding an unrestricted counter or a pushdown stack), and languages defined by space- and time-bounded Turing machines. Our results give as corollaries the closure/non-closure properties of the Chomsky families (regular, context-free, context-sensitive, recursively enumerable) under the ciliate bio-operations. Next, we consider language equations of the type $L \diamond Y = R$, $X \diamond L = R$, where L and R are given languages and X and Y are the unknowns. We study the decidability of the existence of solutions to these equations as well as the existence of singleton solutions.

The paper has five sections in addition to this section. Section 2 formally defines the ciliate bio-operations: synchronized insertion, synchronized deletion, hairpin inversion, and synchronized bi-deletion. Section 3 looks at the closure properties of NCM and its generalizations. Section 4 considers the closure properties of languages defined by space-bounded and time-bounded Turing machines. Section 5 investigates language equations and decision questions involving the ciliate bio-operations. Section 6 is a brief conclusion.

The notations used in the paper are summarized as follows. An alphabet Σ is a finite non-empty set. A word w over Σ is an element of the free semigroup (denoted Σ^+) generated by the letters of Σ and the catenation operation. The length of a word, written $|w|$, is equal to the number of letters in the word. In the free monoid Σ^* we also allow the empty word λ where $|\lambda| = 0$. A language L is a, possibly infinite, set of words over a given alphabet. The complement of a language L is written L^c and is defined as $L^c = \Sigma^* \backslash L$.

For further details on basic formal language theory, the reader is referred to [19].

2. Ciliate Bio-Operations

In this section, we define the basic operations, [2], which will be studied in the remainder of the paper. Two basic operations have been defined in [11,12] that model the processes of intramolecular respectively intermolecular DNA recombinations that are thought to accomplish gene unscrambling.

The operation modeling the intermolecular recombination accomplishes the insertion of a circular sequence vx in a linear string uxw, resulting in a string $uxvxw$. If we ignore the fact that the inserted string is circular, the operation, called *synchronized insertion* (the word "synchronized" points out that insertion can only happen if the sequence x is present in both input strands), is formally defined as follows.

Definition 1. Let α, β be two nonempty words in Σ^*. The synchronized insertion of β into α is defined as: $\alpha \oplus \beta = \{uxvxw \mid \alpha = uxw,\ \beta = vx, x \in \Sigma^+, u, v, w \in \Sigma^*\}$.

The operation modeling intramolecular recombination accomplishes the deletion of a sequence vx from the original strand $uxvxw$, in the form of a circular strand. Ignoring the differences between the linear and circular strands, the resulting operation, that of *synchronized deletion*, is defined as follows.

Definition 2. Let α, β be two nonempty words in Σ^*. The synchronized deletion of β from α is defined as: $\alpha \ominus \beta = \{uxw \mid \alpha = uxvxw, \beta = vx, x \in \Sigma^+, u, v, w \in \Sigma^*\}$.

The operations differ from the original ones defined in [12,11] in that no circular strands are present here. The above two definitions can be extended to languages in the natural way.

We now show that we can consider, without loss of generality, only one-symbol contexts instead of arbitrarily sized contexts for synchronized insertion and deletion.

Lemma 1. *For any* $\alpha, \beta \in \Sigma^+$, $\alpha \oplus \beta = \{u'av'aw' \mid \alpha = u'aw', \beta = v'a, a \in \Sigma, u', v', w' \in \Sigma^*\}$.

Proof. "$\subseteq$" Consider some word $\gamma \in \alpha \oplus \beta$. By definition, $\gamma = uxvxw$ for some $x \in \Sigma^+$, $u, v, w \in \Sigma^*$. As $x \in \Sigma^+$ we can write x as $x = x'a, a \in \Sigma, x' \in \Sigma^*$. We then have $\gamma = ux'avx'aw$ which we can rewrite as $u'av'aw'$ where $u' = ux'$, $v' = vx'$, $w' = w$. Thus, $\gamma \in \{u'av'aw' \mid \alpha = u'aw', \beta = v'a\}$.

"$\supseteq$" Let γ be a word in $\{u'av'aw' \mid \alpha = u'aw', \beta = v'a\}$. Then $\gamma \in \alpha \oplus \beta$, as we can take $x = a$, which gives $\gamma \in \{uxvxw \mid \alpha = uxw, \beta = vx, x \in \Sigma^+\}$. □

Similarly, we can show that:

Lemma 2. *For any* $\alpha, \beta \in \Sigma^+ \alpha \ominus \beta = \{u'aw \mid \alpha = u'av'aw, \beta = v'a, a \in \Sigma, u', v', w \in \Sigma^*\}$.

Although it does not directly model a bio-operation, the operation of synchronized bi-deletion is introduced as it will be needed to investigate the decidability of solution existence for language equations involving synchronized insertion and deletion.

Definition 3. Let $u, v \in \Sigma^+$. The synchronized bi-deletion of v from u is defined as

$$u \boxminus v = \{w \mid u = xayaz,\ v = xaz,\ w = ya,\ a \in \Sigma,\ x, y, z \in \Sigma^*\}.$$

We will also consider a generalization of the hairpin inversion operation *hi* defined in [14]. The name of the operation reflects the fact that it models the process of a DNA strand forming a hairpin, having the end of the hairpin cleaved and then re-attached with the sticky ends switched. This results in the sequence of the cleaved and re-attached region now being the mirror image of what it was prior to the operation.

If $w = a_1a_2 \cdots a_n$, $a_i \in \Sigma$, $1 \leqslant i \leqslant n$ is a word in Σ^+ the *reverse* or *mirror image* of w is denoted by $\mathrm{mi}(w)$ and defined as $\mathrm{mi}(w) = a_n \cdots a_2a_1$.

Definition 4. Let α be a word in Σ^+. The hairpin inverse of α, denoted by $hi(\alpha)$ is defined as $hi(\alpha) = \{xp\mathrm{mi}(y)\mathrm{mi}(p)z \mid \alpha = xpy\mathrm{mi}(p)z$ and $x, y, z \in \Sigma^*,\ p \in \Sigma^+\}$.

This definition can be extended to languages in Σ^+ in the natural way. Similarly to Lemma 1, we again show that it is enough to consider pointers of length one only.

Lemma 3. *If* $\alpha \in \Sigma^+$, $hi(\alpha) = \{xa\mathrm{mi}(y)az \mid \alpha = xayaz, a \in \Sigma, x, y, z \in \Sigma^*\}$.

The mirror image operation has the property that $\mathrm{mi}(\mathrm{mi}(L)) = L$. The hairpin inversion operation is a variation of the mirror image operation in that it inverts subwords inside words of a language. The following lemma answers the question of whether or not applying hairpin inversion twice to a language yields the original language. As it turns out, while the hairpin invertible words of a language L are included in $hi(hi(L))$, the reverse does not hold.

Lemma 4. *If $L \subseteq \Sigma^+$, then for all $a \in \Sigma$, $L \cap \Sigma^* a \Sigma^* a \Sigma^* \subseteq hi(hi(L))$ while $hi(hi(L))$ is not necessarily included in $L \cap \Sigma^* a \Sigma^* a \Sigma^*$.*

Proof. We first prove the inclusion. Suppose $\alpha \in L \cap \Sigma^* a \Sigma^* a \Sigma^*$, then $\alpha = xayaz \in L$ where $a \in \Sigma, x, y, z \in \Sigma^*$ and $xa\,\mathrm{mi}(y)az \in hi(L)$ yielding xa $\mathrm{mi}(\mathrm{mi}(y))az = xayaz \in hi$ $(hi(L))$. That the reverse inclusion does not hold can be seen by considering the language $L = \{gdeacbafdh\}$ over the alphabet $\{a, b, c, d, e, f, g, h\}$. $hi(L) = \{gdeabcafdh, gdfabcaedh\}$ and $hi(hi(L)) = \{gdeacbafdh, gdfacbaedh\} = hi(hi(L)) \cap \Sigma^* a \Sigma^* a \Sigma^*$ and $gdfacbaedh \notin L$. $\square$

3. Closure properties of NCM, NFCM, and NPCM

In this section, we investigate the closure properties of various families (or classes) of languages under the ciliate bio-operations. Our objective is to identify large classes of languages, other than the regular sets and recursively enumerable sets, that are closed under these bio-operations. We will use the terms "family" and "class" interchangeably in the paper.

We recall the definition of reversal-bounded multicounter machines in Ibarra [9]. For $k \geqslant 0$, let NCM(k) be the class of nondeterministic one-way finite automata augmented with k reversal-bounded counters, and NCM be the union of such classes over all k's (see [9] for details). Thus, at every step, a counter can be incremented by 1, decremented by 1, or not changed, and it can be tested for zero. It is reversal-bounded in that in any computation, the number of alternations between nondecreasing mode and nonincreasing mode and vice-versa is bounded by a given constant. For notational convenience we also use NCM(k) and NCM to denote the respective families of accepted languages, and use the same notation to refer to a machine in the classes. Clearly, NCM defines a large class of languages (all regular sets, some non-context-free languages, etc.) Note than an NCM(0) has *no* counter and is an ordinary finite automaton. Hence, the class NCM(0) = REG = regular sets.

Let NPCM be an NCM augmented with an unrestricted pushdown stack. An NFCM is an NCM augmented with a free (i.e., unrestricted) counter. An NPCM (NFCM) with k reversal-bounded counters will be denoted by NPCM(k) (NFCM(k)). Thus, an NPCM(0) (NFCM(0)) is just an ordinary pushdown (one-counter) automaton. Hence, NPCM(0) = CFL = context-free languages, and NFCM(0) = one-counter languages.

There is a nice characterization of NCM (respectively, NPCM, NFCM) in terms of regular sets (respectively, context-free languages, one-counter languages). The *shuffle*

$u \amalg v$ of two words $u, v \in \Sigma^*$ is a finite set consisting of the words $u_1v_1 \ldots u_kv_k$, where $u = u_1u_2 \ldots u_k$ and $v = v_1v_2 \ldots v_k$ for some $u_i, v_i \in \Sigma^*$. If L_1 and L_2 are two languages, their *shuffle* is the language

$$L_1 \amalg L_2 = \bigcup_{u \in L_1, v \in L_2} u \amalg v.$$

A *simple shuffle language* is a language of the form $\Sigma^* \amalg \{a^nb^n \mid n \geqslant 0\}$, for some alphabet Σ and distinct symbols a, b.

It is known [7] that NCM (respectively, NPCM, NFCM) is the smallest class of languages containing the regular sets (respectively, context-free languages, one-counter languages) that is closed under homomorphism and intersection with simple shuffle languages. In particular, it follows that NCM is the smallest class containing the regular sets that is closed under homomorphism, intersection, and the shuffle operation.

The classes NCM, NPCM, NFCM, NPCM(0) NCM(0), NFCM(1), ... share many AFL closure properties (e.g. homomorphism, inverse homomorphism, intersection with regular sets, concatenation, $*$, etc.). But they do not share the same closure properties under the ciliate bio-operations. For example, as we will see, although NCM is closed under all the bio-operations,

- NFCM(0) and NCM(0) are not closed under synchronized insertion (although NPCM(0) is).
- NFCM(0), NPCM(0), NFCM, NPCM are not closed under synchronized deletion and synchronized bi-deletion.

Although NFCM and NCM are closed under hairpin inversion,

- NCM(1), NFCM(0), and NPCM(0) are not closed under hairpin inversion.

For languages defined by space-bounded and time-bounded TM classes, here, too some of the classes are closed under some bio-operations, but yet others are not. Hence, it does not seem possible to deal with all the families considered in this paper in a uniform way, and we will be dealing with them separately.

We will need the following proposition which is easily verified using standard constructions:

Proposition 1. *NCM(k), NPCM(k), and NFCM(k) are closed under intersection with regular sets, homomorphism, and inverse homomorphism.*

We will also need the following result.

Theorem 1. *NCM, NPCM, and NFCM are closed under reversal.*

Proof. The proof uses ideas in [9]. Let M be an NPCM with one pushdown stack and k reversal-bounded counters. We may assume that the machine starts with all counters zero and accepts with all counters zero. We may also assume that each counter makes exactly one reversal in any accepting computation, since a counter making s reversals can be simulated by $(s+1)/2$ counters, each making exactly 1 reversal.

First consider the case when there is only one 1-reversal counter. Let Σ be the input alphabet of M and a, b be two new symbols. We construct an NPCM(0), i.e., an ordinary pushdown automaton M_1 with input alphabet $\Sigma \cup \{a, b\}$ such that any string w accepted by M_1 has the following properties:

1. All occurrences of a's in w precede all occurrences of b's.
2. If the number of a's in w is equal to the the number of b's, then w with the a's and b's deleted, which we denote by $h(w)$, is in $L(M)$.
3. If x is in $L(M)$, then there is a string w accepted by M_1 such that w has the same number of a's and b's, and $h(w) = x$.

The construction of M_1 is straightforward. M_1 on input w, simulates the computation of M treating each symbol a as an increment of 1 to the counter, and each b as a decrement of 1 to the counter. M_1 also makes sure that all occurrences of a's precede all occurrences of b's. M_1 guesses at some point during the simulation that the number of b's seen equals the number of a's (this corresponds to the counter becoming zero), and will no longer see b's in the remainder of the input. M_1 accepts if M accepts.

Since the family CFL (= context-free languages) is closed under reversal, $\mathrm{mi}(L(M_1))$ is also accepted by a pushdown automaton M_2. Finally, we construct an NPCM(1) M_3 accepting $\mathrm{mi}(L(M))$ from M_2. Note that an input x to M_3 no longer contains a's and b's. M_3 has one 1-reversal counter (in addition to the stack). On input x, M_3 simulates the computation of M_2, guessing the occurrences (i.e., reading) of a's as increments to the counter and the occurrences of b's as decrements to the counter. At some time during the computation, M_3 guesses that the counter reverses (and hence will no longer simulate reading a's). M_3 continues simulating M_2, where a decrement of the counter by 1 corresponds to the reading of symbol b by M_2. When the counter becomes zero, M_3 no longer simulates the reading b's. Clearly, M_3 accepts $\mathrm{mi}(L(M))$.

The construction above can easily be generalized when M has k 1-reversal counters. In this case, we need $2k$ new symbols $a_1, b_1, \ldots, a_k, b_k$ for the k counters. We leave the details to the reader.

The cases NCM and NFCM are handled as above, using the fact that regular sets and NFCM(0) (i.e., one-counter languages) are closed under reversal. □

Theorem 2. *NCM is closed under synchronized insertion, synchronized deletion, and synchronized bi-deletion.*

Proof. Let M_1 and M_2 be two NCMs with k_1 and k_2 counters, respectively. We describe the operation of an NCM M accepting the synchronized insertion of $L(M_1)$ with $L(M_2)$. M will have $k_1 + k_2$ counters. M assumes that the input z is of the form $u'av'aw'$. M first simulates M_1 (using the first k_1 counters). At some point, chosen nondeterministically, after it has processed some pointer symbol a, it remembers this symbol and the state, temporarily discontinues the simulation of M_1, and simulates the computation of M_2 on the substring $v'a$ (using the remaining k_2 counters). When M_2 accepts, M resumes the simulation of M_1 on the suffix string w' of the input and accepts if M_1 accepts.

For synchronized deletion, we construct an NCM M accepting the synchronized deletion of $L(M_1)$ with $L(M_2)$ as follows. M, when given input $z = u'aw$, first simulates M_1 (with k_1 counters) until at some time (chosen nondeterministically) when it has processed some symbol a. M then, without moving its input head (i.e., without reading), guesses symbol-by-symbol a string $v'a$ and continues the simulation of M_1 on this string. In parallel, M also simulates the computation of M_2 (using k_2 counters) on $v'a$. When M_2 accepts, M reads the remaining input w and continues the simulation of M_1 and accepts if M_1 accepts.

For synchronized bi-deletion, we construct an NCM M accepting the synchronized bi-deletion of $L(M_1)$ with $L(M_2)$. M when given input $w = ya$, without reading the input, nondeterministically guesses the substring xa (symbol-by-symbol) of $u = xayaz$ and simulates the computation of M_1 on this substring. Thus M also guesses the pointer symbol a and where it occurs (which need not be the first occurrence in xa) after which ya begins. In parallel, M also simulates the computation of M_2 on xa. Then M temporarily discontinues the simulation of M_2 but continues the simulation of M_1 on the input $w = ya$. When M sees pointer symbol a, (again, this need not be its first occurrence), it guesses that it is the last symbol of the input. Without moving on the input, M guesses the symbols comprising the word z and continues the simulation of M_1 on z. In parallel, M also resumes the simulation of M_2 on z. M accepts if both M_1 and M_2 accept. □

Clearly, if in the above theorem M_1 and M_2 are finite automata (i.e., have no counters), then M would also be a finite automaton. Thus,

Corollary 1. *NCM*(0) (= class *of regular sets*) *is closed under synchronized insertion, synchronized deletion, and synchronized bi-deletion.*

Corollary 2. *NPCM and NFCM are closed under synchronized insertion.*

Proof. Given NPCMs M_1 and M_2, we construct an NPCM M accepting the synchronized insertion of $L(M_1)$ with $L(M_2)$ as in the proof of Part 1 of Theorem 2. We note that when M temporarily discontinues the simulation of M_1 and simulates the computation of M_2, M marks the top of the stack before simulating M_2, so that it can use the same stack in the simulation of M_2. After the simulation of M_2, M goes back to the top of the stack of M_1 to resume its simulation.

Given NFCMs M_1 and M_2, the construction of an NFCM M accepting the synchronized insertion is similar. Again, when M temporarily discontinues the simulation of M_1 and simulates the computation of M_2, M stores the value of the free counter into an auxiliary reversal-bounded counter, so that it can use the free counter in the simulation of M_2. After simulating an accepting computation of M_2, M restores the value of the free counter of M_1 using the auxiliary reversal-bounded counter and then continues the simulation of M_1. Thus M will have one free counter and $k_1 + k_2 + 1$ reversal-bounded counters. □

Again, in the above corollary, if M_1 and M_2 are NPCM(0), i.e., ordinary pushdown automata, then M would also be a pushdown automaton. Hence,

Corollary 3. *NPCM*(0) (= *class of context-free languages*) *is closed under synchronized insertion.*

We note that the “ $+$ 1” in k_1+k_2+1 in the second part of the proof of Corollary 2 is necessary since, as we show next, NFCM(0) is not closed under synchronized insertion:

Proposition 2. *NFCM*(0) (= *class of one-counter languages*) *and NCM*(1) *are not closed under synchronized insertion.*

Proof. Consider the languages $L_1=\{a^i pb^i \mid i\geqslant 1\}$ and $L_2=\{c^j d^j p \mid j\geqslant 1\}$, where a, b, c, d, p are distinct symbols. L_1 and L_2 are in NFCM(0); in fact, they are in NCM(1). However, the insertion, L', of L_2 into L_1 is not in NFCM(0), since from the results in [6], $L'\cap a^*pc^*d^*pb^*=\{a^ipc^jd^jpb^i \mid i,j\geqslant 1\}$ is not in NFCM(0). It also follows that NCM(1) is not closed under synchronized insertion, since it is a special case of NFCM(0). □

The remaining proofs in Theorem 2 hold for the following:

Corollary 4. *NPCM and NFCM are closed under synchronized deletion and synchronized bi-deletion with NCM.*

Proposition 3. *NFCM*(0), *NPCM*(0), *NFCM, and NPCM are not closed under synchronized deletion and synchronized bi-deletion.*

Proof. Let

$$L_1=\#(\{a^ib^{2i} \mid i>0\}^* \sqcup\!\sqcup \{\#\})$$

$$L_2=a\{b^ia^i \mid i>0\}^*\#,$$

where $\sqcup\!\sqcup$ is the shuffle operation. Clearly, L_1 and L_2 are context-free languages, i.e., accepted by NPCM(0)’s (note that they are also accepted by NFCM(0)’s). (Similar languages were used in [6] to show that CFL is not closed under left quotient.) However, the language

$$(L_1\ominus L_2)\cap \#b^*=\{b^{2^n} \mid n>0\}$$

is not context-free and, therefore, not in NPCM(0) or in NFCM(0). This language is also not in NFCM or in NPCM, since the languages in these classes have semilinear property [9].

Similarly, let $L_1=\{a^ib^{2i} \mid i>0\}^*$ and $L_2=a\{b^ia^i \mid i>0\}^*$ be two context-free (NFCM(0)) languages. Define the following languages:

$$L_1'=\{a^ib^{2i} \mid i>0\}^*\{a^j\#b^{2j} \mid j>0\},$$

$$L_2'=a\{b^ia^i \mid i>0\}^*\{\#b^e\#a^e \mid e>0\}.$$

Clearly, L_1' and L_2' can also be accepted by NPCM(0)’s (NFCM(0)’s). Note now that

$$[L_2'\{a^*\}^{-1} \boxminus L_1'\{b^*\}^{-1}]\cap b^*\#=\{b^{2^n}\# \mid n\geqslant 0\}.$$

The proposition now follows since CFL (respectively, NFCM(0), NFCM, NPCM) is closed under right quotient with regular languages and intersection with regular languages but $\{b^{2^n}\# \mid n \geqslant 0\}$ is not a context-free language, and not semilinear. □

Proposition 4. *NCM*(1), *NFCM*(0), *and NPCM*(0) *are not closed under hairpin inversion.*

Proof. Let $\Sigma = \{a, b, c, d, p\}$. Define the language $L = \{a^i pb^i c^j pd^j \mid i, j \geqslant 1\}$. Clearly, L can be accepted by an NCM(0) (i.e., a finite automaton with one counter that makes 3 reversals). Hence, L is in NCM(1), NFCM(0), and NPCM(0). However, the hairpin inversion L' of L is not in NPCM(0). Otherwise, $L' \cap a^* pc^* b^* pd^* = \{a^i pc^j b^i pd^j \mid i, j \geqslant 1\}$ would be in NPCM(0) (i.e., context-free), which is not. □

In contrast to the above proposition, for NFCM and NCM, we have:

Theorem 3. *NFCM, NCM, and NCM*(0) (= *REG*) *are closed under hairpin inversion.*

Proof. Let M be an NFCM with k reversal-bounded counters (and, of course, one free counter). We will construct an NFCM M'' accepting the hairpin inversion of $L(M)$. We describe the construction of M'' when $k = 1$. The generalization for any k will be obvious.

Let $1, 2, 3, 4$ be three new symbols. For states q, p of M and nonnegative integers u_1, u_2, v_1, v_2, define the language $L_{qp} = \{1^{u_1} 2^{v_1} 3^{u_2} 4^{v_2} y \mid M$ when started in state q with the free and reversal-bounded counters having values u_1 and v_1, respectively, reaches state p with the free and reversal-bounded counter having values u_2 and v_2, respectively after processing $y\}$. Clearly, we can construct an NFCM M_{qp} (with one free counter and three reversal-bounded counters) accepting L_{qp}. From Theorem 1, mi(L_{pq}) (the reverse of language L_{qp}) can be accepted by an NFCM mi(M_{qp}).

First we describe the operation of an NFCM M' accepting a padded version of the hairpin inversion language. Given an input of the form $xa\text{mi}(y)4^{v_2}3^{u_2}2^{v_1}1^{u_1}az$, M' simulates M on xa. The pointer symbol a and the position where it occurs (there may be several places where it occurs) are nondeterministically guessed by M'. Suppose that after processing this substring, the state of M is q. M' stores the values of the free counter and the reversal-bounded counter into two auxiliary reversal-bounded counters. Then M' guesses a state p of M and simulates the computation of the NFCM mi(M_{qp}) on the substring mi$(y)4^{v_2}3^{u_2}2^{v_1}1^{u_1}$, following the substring xa. (Note that the free counter can be used in the simulation of mi(M_{qp}) since its old value has been stored in an auxiliary counter.) By using other reversal-bounded counters, at the end of the simulation of mi(M_{qp}), M' can check that u_1 and v_1 were the old values stored in the auxiliary counters. M' then sets the free counter and reversal-bounded counter to u_2 and v_2, respectively, and resumes the simulation of M starting in state p on the remaining input string az (note that a is the same pointer symbol). M' accepts if M accepts. Finally, we construct an NFCM M'' from M' by erasing the symbols $1, 2, 3, 4$ from the language accepted by M'. This can be done by Proposition 1. Clearly, M'' accepts the hairpin inversion.

Note that the proof above applies to the case when M is an NCM (i.e., it has no free counter), or when M is an NCM(0) (i.e., it has no counters). M'' will then be an NCM, or an NCM(0). □

We do not think that Theorem 3 can be generalized to hold for NPCM. For consider the language $L = \{xpc\text{mi}(x)dp \mid x \in \{a,b\}^*\}$ over the alphabet $\Sigma = \{a,b,c,d,p\}$. Clearly, L is a linear context-free language and can be accepted by an NPCM(0) which makes only 1-reversal on the stack. Let L' be the hairpin inversion of L. Then $L'' = L' \cap \{a,b\}^* pd\{A,b\}^* cp = \{xpdxcp \mid x \in \{a,b\}^*\}$ should be in NPCM. But we believe (although we do not have a formal proof at this time)that L'' is not in NPCM.

In the next section we will move further up the computational hierarchy and consider the closure properties of space- and time-bounded Turing machines under the three bio-operations.

4. Space-bounded and time-bounded turing machines

In this section, we investigate the closure properties of space-bounded and time-bounded Turing Machine (TM) complexity classes under ciliate bio-operations.

For a space bound $S(n)$, let *NSPACE*$(S(n))$ be the class of languages accepted by $S(n)$ space-bounded nondeterministic Turing machines, and *DSPACE*$(S(n))$ be the deterministic class. Thus, these machines have a two-way read-only input tape (with endmarkers) and multiple read-write worktapes which are $S(n)$ space-bounded. (It is known that any number of worktapes can be merged into one worktape with the same space bound.) Throughout, we assume that $S(n) \geqslant \log n$. Note that *NSPACE*(n) and *DSPACE*(n) are the classes of context-sensitive and deterministic context-sensitive languages, respectively.

Theorem 4. *NSPACE*$(S(n))$ *and* *DSPACE*$(S(n))$ *are closed under hairpin inversion.*

Proof. First consider the case *NSPACE*$(S(n))$. Let M be an NTM with a two-way read-only input (with endmarkers) and an $S(n)$ space-bounded read-write worktape. We construct an $S(n)$ space-bounded NTM M' accepting the hairpin inversion of $L(M)$. Without loss of generality, we may assume that M' has several $S(n)$ space-bounded read-write worktapes (since any number of worktapes can be easily merged into one). Given an input of the form $xa\text{mi}(y)az$, M' simulates the computation of M on input $xayaz$ of length n (not counting the endmarkers). To do this, M' guesses two positions i and j (with $1 \leqslant i < j \leqslant n$) and stores the positions on two worktapes in binary. It checks that the symbols in these positions are the same pointer symbol, say a. Then M' simulates the computation of M using other worktapes. As long as the computation of M is within the substring xa or az, the simulation is straightforward. Note that M' can tell if the computation of M is in these substrings. When M computes in y, M' first moves its head to the second pointer symbol a and then simulates M, where a right (left) move of M in substring y will be simulated as a left (right) move of M' on substring $\text{mi}(y)$. The details are straightforward. Clearly, M' is $S(n)$ space-bounded.

(Note that i and j take only $log\, n$ space, which is no more than $S(n)$ by assumption on $S(n)$.)

For the case $DSPACE(S(n))$, M is a DTM, and M' has to be deterministic also. The construction above still works, but now M' needs to systematically try (lexicographically) all possible values of i and j. We need to assume that M always halts so that a simulation on a current (i, j) that fails to accept can be abandoned by M' and proceed to the next lexicographic (i, j). This assumption can be made without loss of generality since any $S(n)$ space-bounded DTM can be made halting [8]. □

Clearly, the first part of the proof above applies to unrestricted nondeterministic TMs (which are equivalent to deterministic TMs), i.e., to recursively enumerable sets:

Corollary 5. *The class of recursively enumerable sets is closed under hairpin inversion.*

One can obtain similar results for time-bounded TMs. For example, let P (NP) denote the class of languages accepted by polynomial time-bounded deterministic (nondeterministic) TMs. Then the constructions in the proof of Theorem 4 also proves:

Corollary 6. *P and* NP *are closed under hairpin inversion.*

Using the ideas in the proofs of Theorems 2 and 4 (noting that the input is two-way and that positions in the input can be stored in binary on the worktapes), we have now:

Corollary 7. *$NSPACE(S(n))$, $DSPACE(S(n))$, P, and* NP *are closed under synchronized insertion.*

Turning now to the operations of synchronized deletion and synchronized bi-deletion, we have:

Proposition 5. *$DSPACE(S(n))$, $NSPACE(S(n))$, P, and* NP *are not closed under synchronized deletion and synchronized bi-deletion with regular sets.*

Proof. Let $L \subseteq \Sigma^*$ be a recursively enumerable language which is not in $DSPACE$ $(S(n))$ (respectively, $NSPACE(S(n))$, P, NP).

Let a, b be new symbols not in Σ. One can easily show that there exists a language L_1 in $DSPACE(S(n))$ (respectively, $NSPACE(S(n))$, P, NP) such that L_1 consists of words of the form $a^i b\alpha$ where $i \geqslant 0$ and $\alpha \in L$. Furthermore, for all $\alpha \in L$ there exists some $i \geqslant 0$ such that $a^i b\alpha \in L_1$ (see, e.g., [19]).

Suppose # is a symbol not in $\Sigma \cup \{a, b\}$. Consider the language

$$\#(L_1 \amalg \{\#\}) \ominus a^* b\#.$$

Clearly, $a^*b\#$ is regular and moreover, $\#(L_1 \amalg \{\#\}) \ominus a^*b\# = \#L$ which, by the definition of L, is not in $DSPACE(S(n))$ (respectively, $NSPACE(S(n))$, P, NP).

For synchronized bi-deletion, consider again L and L_1 as above. Let $L_2 = \{a^i b\#\alpha\# \mid a^i b\alpha \in L_1\}$. Clearly, L_2 is in $DSPACE(S(n))$ (respectively, $NSPACE(S(n))$, P, NP). However,

$$L_2 \boxminus a^*b\# = \{\alpha\# \mid \alpha \in L\}$$

is a not in $DSPACE(S(n))$ (respectively, $NSPACE(S(n))$, P, NP). □

However, for unrestricted TMs, we have

Proposition 6. *The class of recursively enumerable sets is closed under synchronized deletion and synchronized bi-deletion.*

Proof. Let M_1 and M_2 be two TMs. We construct a TM accepting the synchronized deletion of M_1 with M_2. Given input $z = u'aw$, M first scans and copies the input in an auxiliary worktape, where nondeterministically, after reading $u'a$ (for some pointer symbol a), it inserts a substring $v'a$ before copying the remaining string w. It also marks the two a's. Thus, at the end of this process, the auxiliary worktape contains a string of the form $u'av'aw$, where the two a's are marked. M then simulates M_1 on $u'av'a$ while also simulating M_2 on $v'a$. When M_2 accepts, M continues the simulation of M_1 on w and accepts if M_1 accepts.

We can use similar ideas to show closure under synchronized bi-deletion. □

This concludes our study of closure properties. In the next section, we apply the results obtained above to the study of the decidability of solution existence for language equations involving the ciliate bio-operations.

5. Language equations and decision questions

We begin this section by investigating equations of the type $hi(X) = R$, where R is a given language and X is the unknown. In addition to purely theoretical value, the results in this section give us insights into restrictions on the form of a ciliate genome. Clearly, for a ciliate to be able to descramble its (evolvable) genome using a given operation, it should be decidable if the associated language equation has a solution.

Proposition 7. *Let $R \subseteq \Sigma^*$ be regular language. If there exists a language $L \subseteq \Sigma^*$ such that $hi(L) = R$ then there exists a regular language $R', L \subseteq R' \subseteq \Sigma^*$, with the same property.*

Proof. Construct the language $R' = [hi(R^c)]^c$.

(i) We show that $hi(R') \subseteq R$ by way of contradiction. Assume there exists some $u \in hi(R')$ such that $u \notin R$. Since $u \notin R$, it must be the case that $u \in R^c$. As $u \in hi(R')$ it must be of the form $u = xa\mathrm{mi}(y)az$ where $xayaz \in R'$. However, $u = xa\mathrm{mi}(y)az$ implies $xayaz \in hi(u) \subseteq hi(R^c)$ a contradiction since $R' = [hi(R^c)]^c$.

(ii) We show now that every language $L \subseteq \Sigma^*$ such that $hi(L) \subseteq R$ is included in R'. Indeed, assume there exists $L \subseteq \Sigma^*$ with $hi(L) \subseteq R$ but $L \nsubseteq R'$. Then there must exist a word $u \in L \backslash R'$. As $u \notin R', u \in hi(R^c)$ implies that $u = xa\mathrm{mi}(y)az$ with $xayaz \in R^c$. However, as $u \in L$, by the definition of hairpin inversion, $xayaz \in hi(L) \subseteq R$ a contradiction.

Return now to the proof of the proposition. If there exists L with $hi(L) = R$ then, by (ii), $L \subseteq R'$. By (i) we have that $R = hi(L) \subseteq hi(R') \subseteq R$ which means $hi(R') = R$. By Proposition 1, and the closure of REG (= regular sets) under complement, R' is regular. □

Note that it follows from the proof that R' can be effectively constructed. The preceding proposition aids us in deciding whether an equation $hi(X) = R$ has a solution X in case R is a regular language.

Proposition 8. *If $R \subseteq \Sigma^*$ is a regular language, the problem of whether or not the equation $hi(X) = R$ has a solution $X \subseteq \Sigma^*$ is decidable.*

Proof. Construct $R' = [hi(R^c)]^c$. If $hi(X) = R$ has a solution then R' is also, by Proposition 7, a solution. An algorithm for deciding our problem will consist in effectively constructing R' and then checking whether or not $hi(R') = R$. The problem is thus decidable as the equality of regular languages is decidable. □

We now investigate equations of the form $X \diamond L = R$, $L \diamond Y = R$ where L and R are given languages, X and Y unknowns, and $\diamond$ signifies the synchronized insertion or deletion operation. To find their solutions, we proceed similarly to solving algebraic equations $x + a = b$. Namely, we must employ an operation "inverse" to addition (in this case subtraction) to determine the solution $x = b - a$. As, unlike addition, the operations of synchronized insertion and deletion are not commutative, we will need to define two separate notions: the notion of a left inverse for solving equations $X \diamond L = R$, and of right inverse for solving equations of the form $L \diamond Y = R$.

Definition 5 (Kari [10]). Let $\diamond$, $*$ be two binary word operations. The operation $*$ is said to be the left-inverse of the operation $\diamond$ if, for all words u, v, w over the alphabet Σ, the following relation holds:

$$w \in (u \diamond v) \text{ iff } u \in (w * v).$$

In other words, the operation $*$ is the left-inverse of the operation $\diamond$ if, given a word w in $u \diamond v$, the left operand u belongs to the set obtained from w and the other operand v, by using the operation $*$. The relation "is the left-inverse of" is symmetric.

Proposition 9. *The left-inverse of the operation $\oplus$ of synchronized insertion is the operation $\ominus$ of synchronized deletion.*

Proof. Let $u, v, w \in \Sigma^*$, and $w \in (u \oplus v)$. Then $w = xayaz$ where $a \in \Sigma$, $u = xaz$, $v = ya$, $x, y, z \in \Sigma^*$. By the definition of synchronized deletion, $u = xaz \in (w \ominus v)$. The converse is analogous. □

We can now use Proposition 9 and the following theorem, [10], to investigate solutions of language equations of the type $X \oplus L = R$ where L and R are given languages in Σ^* and X is the unknown.

Theorem 5. *Let L, R be languages over an alphabet Σ and $\diamond, *$ be two binary word (language) operations, left-inverses to each other. If the equation $X \diamond L = R$ has a solution $X \subseteq \Sigma^*$, then also the language $R' = (R^c * L)^c$ is a solution. Moreover, R' includes all the other solutions of the equation (set inclusion).*

Corollary 8. *If the equation $X \oplus L = R$ (respectively $X \ominus L = R$) has a solution, then $R' = (R^c \ominus L)^c$ (respectively $R' = (R^c \oplus L)^c$) is a maximal solution to the equation.*

We shall use the above results to investigate the decidability of the following problems: Given languages L and R over Σ, R regular, *Does there exist a solution X to the equation $X \oplus L = R$?*, and *Does there exist a singleton solution $X = \{w\}$ to the equation $X \oplus L = R$?*

Proposition 10. *The problem "Does there exist a solution X to the equation $X \oplus L = R$?", is decidable for regular languages L and R.*

Proof. For given regular languages $L, R \subseteq \Sigma^*$ define $R' = (R^c \ominus L)^c$. By Corollary 8, if there exists a solution $X \subseteq \Sigma^*$ to the given equation, then also $R' \oplus L = R$. Moreover, by the proof of Proposition 1, the language R' is regular and can be effectively constructed.

The algorithm which decides our problem will start with the construction of R'. Then, if $R' \oplus L$ (which, by 1, can be effectively constructed) equals R then the answer to our problem is YES, and NO otherwise. □

Proposition 11. *The problem "Does there exist a singleton solution $X = \{w\}$ to the equation $X \oplus L = R$?" is decidable for regular languages L and R.*

Proof. Let L and R be nonempty regular languages and let m be the length of the shortest word in R. If there exists a word w such that $\{w\} \oplus L = R$, then it must satisfy $|w| \leqslant m$.

The algorithm for deciding our problem will consist in checking whether or not $\{w\} \oplus L = R$ for each word w with $|w| \leqslant m$. The answer is YES if such a word w is found and NO otherwise. □

If L is a language over an alphabet Σ, the word $x \in \Sigma^+$ is called *left-useful* with respect to $\ominus$ and L (shortly, left-useful) if there exists a $y \in L$ such that $x \ominus y \neq \emptyset$. A language X is called left-useful with respect to $\ominus$ and L (shortly, left-useful), if it consists only of left-useful words. From the above definitions it follows that the problem "Does there exist a solution X to the equation $X \ominus L = R$?" and its singleton version are equivalent to the corresponding problems where the existence of a left-useful language or word are investigated. Therefore, in the sequel, we will mean a left-useful language when referring to a language or word whose existence is sought.

An argument similar to Proposition 10, and based on the effectiveness of the proofs of closure of REG under $\oplus$ and $\ominus$, shows that the problem "Does there exist a solution X to the equation $X \ominus L = R$?" is decidable for regular languages L and R.

The following decidability result is basically a consequence of the fact that the result of a synchronized deletion from a word is a finite set.

Proposition 12. *The problem "Does there exist a word w such that $w \ominus L = R$?" is decidable for regular languages L and R.*

Proof. Let L, R be regular languages over Σ. Note that, if R is an infinite language, the answer to our problem is NO. If R is finite, we can effectively construct the regular set

$$P = (R^c \oplus L)^c \setminus \bigcup_{S \subsetneq R} (S^c \oplus L)^c.$$

Claim. For all $w \in \Sigma^*$ we have: $w \in P$ iff $w \ominus L = R$.

Indeed, $(R^c \oplus L)^c = \{v \mid v \ominus L \subseteq R\}$. Therefore, if $w \ominus L = R$ then $w \in \{v \mid v \ominus L \subseteq R\}$, but $w \notin \{v \mid v \ominus L \subseteq S \subset R\}$ and consequently $w \in P$.

For the reverse implication, let $w \in P$. As $w \ominus L \subseteq R$ but $w \ominus L$ is not included in any proper subset of R, we have $w \ominus L = R$. The proof of the claim is thus complete.

The algorithm for deciding our problem will check first the finiteness of R. If R is infinite, the answer is NO. Otherwise, the set P is constructed and its emptiness is decided. If $P = \emptyset$, the answer is NO. Otherwise, the answer is YES and any word $w \in P$ satisfies the equation $w \ominus L = R$. $\square$

To investigate symmetric equations of the type $L \oplus Y = R$ and $L \ominus Y = R$ where L and R are given languages and Y is an unknown language, we shall make use of the following result from [10], keeping in mind that, in the case of synchronized deletion, we are actually investigating the existence of right-useful solutions (the notion is defined similarly to that of left-useful solutions).

Theorem 6. *Let L, R be languages over Σ and $\diamond, *$ be two binary word (language) operations right-inverses to each other. If the equation $L \diamond Y = R$ has a solution Y, the language $R' = (L * R^c)^c$ is a maximal solution.*

The notion of right-inverse in the preceding theorem, similar to the notion of left-inverse, is formally defined in [10] as follows.

Definition 6 (Kari [10]). Let $\diamond, *$ be two binary word operations. The operation $*$ is said to be right-inverse of the operation $\diamond$ if, for all words u, v, w in Σ^* the following relation holds: $w \in (u \diamond v)$ iff $v \in (u * w)$.

By using Theorem 6 we could find solutions to equations of the form $L \oplus Y = R$, $L \ominus Y = R$ if we found the right inverses of $\oplus$ and $\ominus$.

Proposition 13. *The right-inverse of synchronized deletion $\ominus$ is synchronized bi-deletion. The right-inverse of synchronized insertion is reversed synchronized bi-deletion.*

Corollary 9. *If the equation $L \oplus Y = R$ (respectively $(L \ominus Y = R)$) has a solution, then $R' = (R^c \boxminus L)^c$ (respectively $(L \boxminus R^c)^c$) is a maximal solution.*

Proposition 14. *The problem of whether or not there exists a solution Y to the equations $L \oplus Y = R$, $L \ominus Y = R$ is decidable for regular languages L and R.*

Proof. Similar to that of Proposition 10, using Theorem 6 and Proposition 1. $\square$

We now show some undecidable properties. We will use the fact that 1-reversal NCM(1) (i.e., nondeterministic finite automata augmented with one counter whose counter makes exactly 1 reversal) has undecidable universe (Is $L = \Sigma^*$?) and set difference (Is $L_1 - L_2 = \emptyset$?) problems [1,9]. In fact the undecidability holds even when the counter makes exactly 1 reversal.

Proposition 15. *The problem "Does there exist a solution X to the equation $X \oplus L = R$?" is undecidable for NCM(1) languages L and regular languages R.*

Proof. Let Σ be an alphabet, card$(\Sigma) \geqslant 2$, and let # be a letter which does not occur in Σ. There exists a regular language $R = \#\Sigma^*\#$ such that our problem is undecidable for this particular R and NCM(1) languages L. Indeed, note that the equation

$$X \oplus L\# = \#\Sigma^*\#$$

holds iff $X = \{\#\}$ and $L = \Sigma^*$. Hence, if we could decide our problem, we would be able to decide the problem " is $L = \Sigma^*$?" for NCM(1) languages L, which is impossible. $\square$

Note that in the preceding proof the language $X = \{\#\}$ is a singleton. Consequently, the singleton version of the problem in Proposition 15 is also undecidable for NCM(1) languages L and regular languages R.

Proposition 16. *The problem "Does there exist a language X such that $X \ominus L = R$" is undecidable for NCM(1) languages L and regular languages R.*

Proof. Let Σ be an alphabet, card$(\Sigma) \geqslant 2$ and #, \$ be symbols not in Σ. Let $R = \{\$\}$ and, f or arbitrary given languages $L_1, L_2 \in$ NCM(1), let

$$L =, \#L_1\#\$ \cup L_2\#$$

Claim. The equation $X \ominus [\#L_1\#\$ \cup L_2\#] = \{\$\}$ has a solution X iff $L_1 \backslash L_2 \neq \emptyset$.

"$\Leftarrow$" If $L_1 \backslash L_2 \neq \emptyset$, let $X = \{\$\#u\#\$\}$, where $u \in L_1 \backslash L_2$. It is easy to see that X is a solution.

"$\Rightarrow$" Assume X is a solution and let $w \in X$. Then w can be of the form $\$\#u\#\$$, $u \in L_1$, as $\$\#u\#\$ \ominus \#u\#\$ = \{\$\}$. However, if u belongs also to L_2 then $\$\#u\#\$ \ominus u\# = \$\#\$$, a contradiction with the form of words in R.

Consequently, $X = \$\#(L_1 \backslash L_2)\#\$$ and the claim is proved.

It follows then that, if we could solve the problem of the proposition we would also be able to decide for arbitrary NCM(1) languages L_1, L_2 whether $L_1 \backslash L_2 = \emptyset$, which is impossible. $\square$

The proof of Proposition 16 shows that also the singleton version of the problem is undecidable.

Proposition 17. *The existence of a solution Y to the equation $L \oplus Y = R$ is undecidable for regular languages R and NCM(1) languages L.*

Proof. Let Σ be an alphabet, card$(\Sigma) \geqslant 2$, and let $\#$ be a symbol not in Σ. Let L be an arbitrary NCM(1) language, and let $R = \Sigma^*\#\#$ be a regular language.

Note that the equation $L\# \oplus Y = \Sigma^*\#\#$ has a solution Y iff $Y = \{\#\}$ and $L = \Sigma^*$.

Consequently, if we could decide the problem in the Proposition we could decide for an arbitrary NCM(1) language whether it equals Σ^*, which is impossible. $\square$

Note that, in Proposition 17, the solution $Y = \{\#\}$ is a singleton, therefore also the singleton version of the problem of Proposition 17 is undecidable.

Proposition 18. *The problem of whether or not there exists a solution to the equation $L \ominus Y = R$ is undecidable for NCM(1) languages L and regular languages R.*

Proof. Similar to Proposition 17, by considering $R = \Sigma^*\#$ and the equation $L\#\# \ominus Y = \Sigma^*\#$ for arbitrary NCM(1) languages L. $\square$

Note that Proposition 18 solves also the singleton version of the problem.

This concludes our study of language equations involving the three bio-operations.

6. Conclusion

We have considered the properties of three operations used in the modeling of the ciliate gene descrambling process: synchronized insertion, synchronized deletion and hairpin inversion. We found that all the families of languages studied are closed under synchronized insertion with the exception of NFCM(0) and NCM(1), while only the families of regular, NCM and recursively enumerable languages are closed under synchronized deletion. Further, we found that all families considered except NCM(1), CFL, NFCM(0), NPCM(0) and NPCM were closed under hairpin inversion. In order to consider language equations involving each of the three operations we have also defined the operation of synchronized bi-deletion (the right-inverse of synchronized deletion) and showed only the families of regular, NCM and recursively enumerable languages to be closed under this operation.

We demonstrated that the existence of a solution X to the equation $hi(X) = R$, where R is a regular language is decidable. Additionally, the existence of a solution was shown to be decidable for equations of the form $L \diamond Y = R$ and $X \diamond L = R$ where $\diamond$ is one of synchronized insertion or synchronized deletion operations and L, R are regular languages. The same problems are undecidable in the case that L is a NCM(1) language.

By investigating the properties of these formal operations, we have provided some insight into the nature of the bio-operations that must be present in the ciliate gene descrambling mechanism. Continued theoretical study of the gene descrambling problem combined with improved biological results will hopefully lead to a better understanding of this fascinating process.

References

[1] B.S. Baker, R.V. Book, Reversal-bounded multipushdown machines, J. Comput. System Sci. 8 (1974) 315–332.

[2] M. Daley, L. Kari, Some properties of ciliate bio-operations, Preproc. 6th Internat. Conf. on Developments in Language Theory, 2002, pp. 122–139.

[3] J. Dassow, V. Mitrana, A. Salomaa, Operations and language generating devices suggested by the genome evolution, Theoret. Comput. Sci. 270 (2002) 701–738.

[4] A. Ehrenfeucht, D.M. Prescott, G. Rozenberg, Computational aspects of gene (un)scrambling in ciliates, in: L.F. Landweber, E. Winfree (Eds.), Evolution as Computation, Springer, Berlin, Heidelberg, 2001, pp. 45–86.

[5] R. Freund, C. Martin-Vide, V. Mitrana, On some operations on strings suggested by gene assembly in ciliates, New Gen. Comput. 20 (2002) 279–293.

[6] S. Ginsburg, Algebraic and Automata-Theoretic Properties of Formal Languages, North-Holland, Amsterdam, 1975.

[7] T. Harju, O.H. Ibarra, J. Karhumaki, A. Salomaa, Some decision problems concerning semilinearity and commutation, in: J. Comput. System Sci. extended abstract has appeared in Proc. Twenty Eighth Internat. Colloq. 2002, Automata, Languages and Programming, 2001, pp. 579–590, to appear.

[8] J.E. Hopcroft, R. Motwani, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Second Edition, Addison-Wesley, Reading, MA, 2001.

[9] O.H. Ibarra, Reversal-bounded multicounter machines and their decision problems, J. Assoc. Comput. Mach. 25 (1978) 116–133.

[10] L. Kari, L.F. Landweber, Computational power of gene rearrangement, in: E. Winfree, D. Gifford (Eds.), DNA5, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Vol. 54, 2000, pp. 207–216.

[11] L. Kari, On language equations with invertible operations, Theoret. Comput. Sci. 132 (1994) 129–150.

[12] L.F. Landweber, L. Kari, The evolution of cellular computing: nature's solutions to a computational problem, in: L. Kari, H. Rubin, D.H. Wood (Eds.), DNA4, BioSystems, Elsevier, Amsterdam, 1999, pp. 3–13.

[13] L.F. Landweber, T. Kuo, E. Curtis, Evolution and assembly of an extremely scrambled gene, Proc. Nat. Acad. Sci. 97 (7) (2000) 3298–3303.

[14] I. Petre, A. Ehrenfeucht, T. Harju, G. Rozenberg, Patterns of micronuclear genes in cilliates, in: N. Jonoska, N. Seeman (Eds.), DNA7, Lecture Notes in Computer Science, Vol. 2340, Springer, Berlin, 2002, pp. 279–289.

[15] D.M. Prescott, Cutting, splicing, reordering, and elimination of DNA sequences in hypotrichous ciliates, BioEssays 14 (5) (1992) 317–324.
[16] D.M. Prescott, The unusual organization and processing of genomic DNA in hypotrichous ciliates, Trends in Genet. 8 (1992) 439–445.
[17] D.M. Prescott, Genome gymnastics: Unique modes of DNA evolution and processing in ciliates, Nature Rev. Gen. 1 (2000) 191–198.
[18] D.M. Prescott, A. Ehrenfeucht, G. Rozenberg, Molecular operations for DNA processing in hypotrichous ciliates, Eur. J. Protistol. 37 (2001) 241–260.
[19] A. Salomaa, Formal Languages, Academic Press, New York, 1973.