Language-Theoretic Aspects of DNA Complementarity

Valeria Mihalache
Arto Salomaa

Turku Centre for Computer Science
TUCS Technical Report No 202
September 1998
ISBN 952-12-0278-5
ISSN 1239-1891

Abstract

The optimism about the possibilities of DNA computing is based on two central issues: the Watson-Crick complementarity and the massive parallelism of DNA strands. While the latter issue renders exhaustive searches possible and thus may settle problems previously considered intractable, the former issue is the cause behind the universality of many models of DNA computing. Moreover, complementarity can be viewed as a purely language-theoretic operation: undesirable circumstances in a string trigger a transition to the complementary string. This aspect of complementarity is investigated in the present paper, mainly from the point of view of L systems. New types of word sequences will be discovered. Sometimes the resulting decision problems are equivalent to well-known open problems from other areas.

TUCS Research Group
Mathematical Structures of Computer Science

1 Introduction

Adleman's celebrated experiment, [1], gave rise and motivation to a large number of studies dealing with DNA computing. The reader is referred to [6] as an overall survey. Apart from creating new computing paradigms, such studies have also enhanced the classical theory of formal languages. Indeed, the relevance of DNA computing to the theory of formal languages can be concluded from many chapters of the recent Handbook [9]. We begin with a brief description of the basics about DNA and complementarity.

DNA (deoxyribonucleic acid) is found in all living organisms as the storage medium for genetic information. It consists of polymer chains, customarily referred to as *DNA strands*. A chain is composed of *nucleotides*, also referred to as *bases*. The chains are also referred to as *oligonucleotides*, briefly *oligos*. The four DNA nucleotides or bases are customarily denoted by A (adenine), C (cytosine), G (guanine), and T (thymine). The *DNA alphabet* $\Sigma_{DNA} = \{A, C, G, T\}$ will be important in our subsequent considerations.

Thus, DNA strands may be viewed as words over the DNA alphabet. According to a chemical convention, each strand has a " 5' end" and a " 3' end", for instance,

$$5'\ ATTAGCAT\ 3' \quad \text{or} \quad 3'\ TAATCGTA\ 5',$$

making the words oriented. However, this orientation is irrelevant for our purposes and will be ignored in the sequel.

The familiar double helix of DNA arises by the bondage of two separate strands. In the formation of such double strands a phenomenon known as *Watson-Crick complementarity* comes into the picture. Bonding happens by the pairwise attraction of bases: A bonds with T, and C bonds with G. This is the reason why the unordered pairs (A, T) and (C, G) are referred to as *complementary* pairs of bases. Bonding occurs only if the bases in the corresponding positions in the two strands are complementary. (Moreover, they have to have opposite orientation but, as already pointed out, we will ignore here the orientation.) Thus, the two strands mentioned above will form the double strand

$$\begin{array}{c} ATTAGCAT \\ TAATCGTA \end{array}$$

Such double strands form a data structure of a new type, a data structure characteristic for the theory of DNA computing. The very nature of this data structure is essential for DNA computing. It is the source of the strength of DNA computing because, in some sense, it makes the powerful *twin-shuffle language* "freely" available.

The interconnection between the Watson-Crick complementarity and the twin-shuffle language was pointed out in [8] and further investigated in [6] and [10]. It is important to note that the information content of a double strand is the same as that of a single strand; because of the complementarity the two strands only repeat each other. However, in DNA computing we are supposed to know the history of the double strands, how they came into being. This knowledge makes a crucial difference. It enables Adleman to conclude that the double strands represent proper paths in his graph, instead of only arbitrary sequences of nodes. This knowledge also forms the basis for using the universality properties of the twin-shuffle language in models of DNA computing, [6].

Apart from these universality considerations, complementarity gives rise to an interesting language-theoretic operation. A word is changed to a complementary word, either randomly or by same control mechanism. The study of this operational aspect of complementarity was begun in [5] and will be continued in the present paper.

Our exposition is largely self-contained. In particular, no previous knowledge about DNA computing is assumed on the part of the reader. Many of the results require some knowledge about $D0L$ sequences and L systems in general. The reference [7] may be consulted in this respect.

2 Operational Complementarity. Triggers

Consider again the DNA alphabet $\Sigma_{DNA} = \{A, C, G, T\}$. The letter-to-letter endomorphism h_W of Σ^*_{DNA} mapping each letter to the complementary one is called the *Watson-Crick morphism.* Thus,

$$h_W(A) = T, \ h_W(T) = A, \ h_W(C) = G, \ h_W(G) = C.$$

The two strands in a double strand are always of the form $(x, h_W(x))$ with $x \in \Sigma^+_{DNA}$.

The following generalization of these notions is straightforward.

A *DNA-like alphabet* Σ is an alphabet with even cardinality $2n$, $n \geq 1$, where the letters are enumerated as follows:

$$\Sigma = \{a_1, \ldots, a_n, \overline{a_1}, \ldots, \overline{a_n}\}.$$

Thus, each of the non-barred letters a_i, $1 \leq i \leq n$, has its barred version $\overline{a_i}$. We say that a_i and $\overline{a_i}$ are *complementary.* The letter-to-letter endomorphism of Σ^* mapping each letter to the complementary letter is referred to as the *Watson-Crick morphism.*

When the DNA alphabet Σ_{DNA} is viewed as DNA-like, we consider the *purines* A and G as non-barred letters: $a_1 = A$ and $a_2 = G$. Hence the *pyrimidines* T and C are their barred versions: $\overline{a_1} = T$ and $\overline{a_2} = C$.

It is both natural and sufficient for our purposes to consider only the "mild" generalization of the alphabet Σ_{DNA} introduced above. Of course, a more general notion of a DNA-like alphabet would be the pair (Σ, ρ), where ρ is a binary relation on Σ, satisfying some suitably chosen restrictive conditions.

We now consider complementarity in an *operational* sense. This means that complementarity is viewed as an operation: together with or instead of a word w we consider the complementary word $h_W(w)$. In other words, a string induces the complementary string either randomly or guided by a control device. For instance, when considering a generative process for words, we might want to exclude certain "bad" words. Whenever the process is about to generate such a "bad" word w, we take instead of w the complementary word $h_W(w)$. This induces the following condition, referred to in the sequel as the "soundness of triggers". Whenever w is "bad", then $h_W(w)$ is "good". This condition guarantees that we generate only "good" words, provided the transition to the complementary words is affected in the way described above.

Clearly, the above discussion is in very general terms: the generative process and the "good" and "bad" words can be almost anything. In this paper, our generative processes will be *L systems*. In fact, the attention is focused on *D0L systems*.

We now define the fundamental notion of a *trigger for complementarity transition* or, briefly, a *trigger*. Consider a DNA-like alphabet

$$\Sigma = \{a_1, \ldots, a_n, \overline{a_1}, \ldots, \overline{a_n}\}$$

and the associated Watson-Crick morphism h_W:

$$h_W(a_i) = \overline{a_i}, \quad h_W(\overline{a_i}) = a_i, \quad 1 \leq i \leq n.$$

(Sometimes we denote h_W simply by a bar. Then it is understood that the double bar is the identity: $\overline{\overline{a_i}} = a_i$, $1 \leq i \leq n$.) By definition, a *trigger* TR is a recursive subset of Σ^*, $TR \subseteq \Sigma^*$. A trigger TR is *sound* if, whenever w is in TR, then $h_W(w)$ is not in TR.

The intuitive meaning of the notion of a trigger should now be clear. The set TR consists of words referred to above as "bad". Whenever a generative process leads to such a word, it is replaced by its complementary word. The soundness condition guarantees that the new word belongs to the complement of TR, that is, to the set of "good" words. The soundness condition does not say anything about the words w in the complement of TR. Thus, for such a w, the word $h_W(w)$ may belong to either one of the two sets TR and $\Sigma^* - TR$.

The assumption of TR being recursive is made because we want both of the sets of "bad" and "good" words to be recursive. If we are dealing with a class of languages closed under intersection with regular sets, most of the considerations become rather simple under the additional assumption of TR being regular. Somewhat more general is the case, where intersections of languages in the original class and regular sets belong to a class with strong decidability properties. Then some of the difficulties met below can be avoided. For instance, the intersection of a $D0L$ or $0L$ language with a regular set is always in the class of $ET0L$ languages.

Our very natural choice of a trigger, presented below in the definition of a Watson-Crick $D0L$ system, is a context-free language. Thus, arguments and results based on intersections with regular sets, are not applicable. For instance, this holds with respect to the results of [2] and [3], where some of the notions are similar to our notion of a Watson-Crick $D0L$ system.

After the present section, we will focus the attention in this paper to a specific generative device augmented with a specific trigger. However, we want to emphasize that the notion is still quite general and may lead to surprising observations.

One can think of various ways of constructing sound triggers. In [5] we consider an $0L$ system augmented with the trigger $\Sigma^* w \Sigma^*$, where w is a specific word. Thus, all words containing w as a subword are "bad". Clearly, triggers of this type are regular. As such they are not sound because, for instance, both the word $wh_W(w)$ and its complementary word $h_W(wh_W(w)) = h_W(w)w$ are "bad". However, *relative soundness* (that is, soundness with respect to the $0L$ system considered) was achieved in [5]; among the words generated, the complementary word of a "bad" word is always "good".

We conclude this section with examples, showing different possibilities of defining a trigger.

Example 1. Consider an ordinary $D0L$ system, say the system G with the alphabet $\Sigma = \{a, t, c, g\}$, the axiom aga and the productions

$$a \to a, \ g \to ta, \ c \to a, \ t \to g.$$

Supplement the system G with the trigger

$$TR = \{w \in \Sigma^* \mid w \notin \{a, g\}^*\}.$$

By this trigger, whenever a string not in $\{a, g\}^*$ would be produced in the normal course of a derivation in the $D0L$ system G, the complementary word must be taken. Denoting by $\Rightarrow$ a derivation step in a $D0L$ system and by $\Rightarrow_t$ a derivation step in the system supplemented with the trigger, we have that a rewriting

$$aga \Rightarrow ataa$$

must be replaced by the derivation step

$$aga \Rightarrow_t tatt.$$

Formally, the yield relation $\Rightarrow_t$ is defined from the yield relation $\Rightarrow$ of the $D0L$ system G as follows. Let h_W be the Watson-Crick morphism, with the convention $a = a_1, g = a_2, t = \overline{a_1}, c = \overline{a_2}$. Assume that α is a word over the set $\{a, g\}$. Then $\alpha \Rightarrow_t \beta$ is either $\alpha \Rightarrow \beta$ and $\beta \in \{a, g\}^*$, or $\alpha \Rightarrow \beta'$, $\beta' \notin \{a, g\}^*$ and $\beta = h_W(\beta')$.

A derivation in the system G proceeds as

$$aga \Rightarrow_t \mathbf{tatt} \Rightarrow_t gagg \Rightarrow_t \mathbf{attatat} \Rightarrow_t aggagag \Rightarrow_t \ldots$$

We have denoted with bold characters the strings that are complementary of the strings resulted in a normal $D0L$ derivation.

One can observe an alternation of the usual derivation steps with steps where the complementary transition is taken. Observe that our trigger is not sound.

Example 2. Let $a_1 = a, \overline{a_1} = b$ and consider a $D0L$ system G' defined by the alphabet $\Sigma' = \{a, b\}$, the axiom a and the production rules

$$a \rightarrow a,\ b \rightarrow ba.$$

Consider further the trigger

$$TR' = \{w \in \Sigma^* \mid |w|_a \geq |w|_b\}.$$

With the same convention of putting the strings resulted from a complementary transition in bold characters, a derivation in the system G' enhanced with the trigger TR' starts as

$$a \Rightarrow_t \mathbf{b} \Rightarrow_t \mathbf{ab} \Rightarrow_t \mathbf{bab} \Rightarrow_t \mathbf{abbab} \Rightarrow_t \mathbf{bababbab} \Rightarrow_t \ldots$$

Observe that at each derivation step the complementary transition is taken. Also, the sequence of strings thus produced is the well known $D0L$ sequence whose growth function is the famous Fibonacci sequence!

Such a property holds for any system as above. Consider a Watson-Crick system constructed from a $D0L$ system $G = (\Sigma, g, w_0)$, such that in the resulted sequence a complementary transition is taken at any derivation step. Then one can prove that there exists a $D0L$ system $G' = (\Sigma, g', w_0)$ producing the same word sequence. The morphism g' is defined as $g'(x) = h_W(g(x))$ for any $x \in \Sigma$, where h_W is the Watson-Crick morphism.

3 Watson-Crick $D0L$ systems

We will now introduce the basic model investigated in this paper. In fact, it was discussed already in [5], with a slightly different terminology.

Consider again the DNA-like alphabet $\Sigma = \{a_1, \ldots, a_n, \overline{a_1}, \ldots, \overline{a_n}\}$. In analogy with the DNA alphabet (recall our convention $A = a_1, G = a_2$), we call also now the non-barred letters *purines* and the barred letters *pyrimidines*. The subset of Σ^* consisting of all words, where the number of occurrences of pyrimidiness is strictly greater than that of purines is denoted by PYR. The complement of PYR is denoted by PUR. Thus, words with equally many occurrences of purines and pyrimidiness belong to PUR. Clearly, both PUR and PYR are context-free nonregular languages.

Definition 1 *A* Watson-Crick $D0L$ system *is a construct* $G = (\Sigma, g, w_0)$, *where* $\Sigma = \{a_1, \ldots, a_n, \overline{a_1}, \ldots, \overline{a_n}\}$, $g : \Sigma^* \to \Sigma^*$ *is a morphism, and* w_0 *(the* axiom*) is a nonempty word in the set* PUR*. The* sequence $S(G)$ *defined by* G *consists of the words* $w_0, w_1, w_2, \ldots,$ *where for* $i \geq 0$,

$$w_{i+1} = \begin{cases} g(w_i), & \text{if } g(w_i) \in PUR \\ h_W(g(w_i)), & \text{if } g(w_i) \in PYR. \end{cases}$$

The language $L(G)$ *generated by* G *consists of all words in the sequence* $S(G)$*. The sequence of word lengths* $|w_0|, |w_1|, |w_2|, \ldots$ *is referred to as the* length sequence *of* G *and the function defined by* $f(i) = |w_i|$, *for* $i \geq 0$, *as the* growth function *of* G.

As a construct, a Watson-Crick $D0L$ system is the same as a $D0L$ system. The restrictions concerning the alphabet and the form of the axiom are unessential because any $D0L$ system has an isomorphic variant satisfying the restrictions. However, as will become clear later on, the items defined by G (sequence, language, length sequence, growth function) are very different from the corresponding items defined by ordinary $D0L$ systems.

Within the framework of our discussion in Section 2, we can say that the sequence $S(G)$ was defined using a $D0L$ system augmented with the trigger PYR. Indeed, the following lemma is immediate from the definitions.

Lemma 1 *The trigger PYR is sound. If G is a Watson-Crick $D0L$ system then every word in the sequence $S(G)$ belongs to the set PUR.*

The set PUR is not sound as a trigger. The situation is otherwise symmetric but the words with an equal number of occurrences of purines and pyrimidines should not be included in a trigger.

Much of the standard terminology about $D0L$ systems can be readily extended to concern Watson-Crick $D0L$ systems as well. For instance, the *sequence equivalence problem* for Watson-Crick $D0L$ systems is the problem of deciding for given Watson-Crick $D0L$ systems G_1 and G_2 whether or not the sequences $S(G_1)$ and $S(G_2)$ coincide. The sequence equivalence problem (as well as the language equivalence problem) is decidable for $D0L$ systems, whereas its decidability is open for Watson-Crick $D0L$ systems.

Although Watson-Crick $D0L$ systems bear some resemblance to the piecewise deterministic systems discussed in [2] and [3], the whole setup is very different because the trigger set is not regular. A Watson-Crick $D0L$ system can be viewed also as a $DT0L$ system with two morphisms g and $h_W g$, as well as with a special control device guiding the selection of the morphism. Contrary to $DT0L$ systems in general, the control device gives rise to determinism and, thus, the system generates a unique sequence of words.

We may also ask the question: Is a sequence of words, defined by some other means, actually a Watson-Crick $D0L$ sequence? If the sequence is defined by an ordinary $D0L$ system, the answer is always positive, in a trivial way. We only consider the letters of the $D0L$ system as purines, and introduce their barred versions for which the morphism can be extended in an arbitrary way. In this fashion the pyrimidines will be "ghost" letters which never actually occur in the sequence. The sequence is a Watson-Crick $D0L$ sequence in which a complementarity transition never takes place. Below in Section 7, we will consider less trivial solutions of this problem.

We will now discuss a simple example of a Watson-Crick $D0L$ system. The example, introduced in [5], is theoretically very important because it demonstrates clearly the amazing capabilities of Watson-Crick $D0L$ systems, reaching far beyond those of ordinary $D0L$ systems. The example will be used in some of our technical results.

Define

$$G_{mix} = (\Sigma, g, a_1 a_2 \overline{a_3}),$$

where $\Sigma = \{a_1, a_2, a_3, \overline{a_1}, \overline{a_2}, \overline{a_3}\}$ and the morphism g is given by the rules

$$a_1 \to a_1,\ a_2 \to a_2,\ a_3 \to a_3,\ \overline{a_1} \to \overline{a_1 a_2},\ \overline{a_2} \to \overline{a_2},\ \overline{a_3} \to \overline{a_3 a_3 a_3}.$$

The beginning of the sequence $S(G_{mix})$ looks as follows, where bold characters indicate that complementarity transition has taken place:

$$a_1 a_2 \overline{a_3},\ \overline{\mathbf{a_1}}\ \overline{\mathbf{a_2}}\mathbf{a_3^3},\ \overline{a_1}\ \overline{a_2}^2 a_3^3,\ \mathbf{a_1 a_2^3}\overline{\mathbf{a_3}}^{\mathbf{3}},\ \overline{\mathbf{a_1}}\ \overline{\mathbf{a_2}}^{\mathbf{3}}\mathbf{a_3^9},\ \overline{a_1}\ \overline{a_2}^4 a_3^9,\ \overline{a_1}\ \overline{a_2}^5 a_3^9,$$

$$\overline{a_1}\ \overline{a_2}^6 a_3^9,\ \overline{a_1}\ \overline{a_2}^7 a_3^9,\ \overline{a_1}\ \overline{a_2}^8 a_3^9,\ \mathbf{a_1 a_2^9}\overline{\mathbf{a_3}}^{\mathbf{9}},\ \overline{\mathbf{a_1}}\ \overline{\mathbf{a_2}}^{\mathbf{9}}\mathbf{a_3^{27}},\ \overline{a_1}\ \overline{a_2}^{10} a_3^{27},$$

after which there are 16 words before the next complementarity transition.

Observe that only the barred letters induce growth. The length sequence is strictly growing:

$$3, \mathbf{5}, 6, \mathbf{7}, \mathbf{13}, 14, 15, 16, 17, 18, \mathbf{19}, \mathbf{37}, 38, \ldots$$

We see that the growth fluctuates between linear growth (from i to $i+1$) and exponential growth (from $1 + 2 \times 3^i$ to $1 + 3^i + 3^{i+1}$). Such a mixture of growth types is very strange from the point of view of $D0L$ growth functions. (Our notation G_{mix} reflects this mixture.) The growth function f of G_{mix} was analyzed further in [5], and also the following explicit characterization of it was obtained:

$$f(i) = \begin{cases} 3, & \text{if } i = 0, \\ 1 + 3^n + 3^{n+1} + k, & \text{if } i = 3^n + n + k, \\ & \quad \text{for any } n \geq 0,\ 0 \leq k \leq 2 \times 3^n - 1, \\ 1 + 2 \times 3^{n+1}, & \text{if } i = 3^n + n + 2 \times 3^n, \\ & \quad \text{for any } n \geq 0. \end{cases}$$

It can be shown that the growth function and the length sequence of G_{mix} are indeed much more general than anything obtainable by ordinary $D0L$ systems. In what follows we use the terminology of length sequences; of course everything can be expressed in terms of growth functions as well. We say that a sequence $z(i)$, $i = 0, 1, 2, \ldots,$ of integers is $\mathbb{Z}$-*rational* if it can be expressed as the difference of two $D0L$ length sequences:

$$z(i) = d_1(i) - d_2(i), \quad \text{for all } i \geq 0,$$

where $d_j(i)$, $j = 1, 2$ is the length sequence of a $D0L$ system. (The reader is referred to [4] and [7] for other, perhaps more natural ways of defining a $\mathbb{Z}$-rational sequence. Our definition is equivalent with the others and especially suitable for our purposes. We mention here the following alternative very intuitive definition. A sequence $z(i)$ is $\mathbb{Z}$-rational if there is a square matrix M with integer entries such that, for $i \geq 1$, $z(i)$ equals the number in the upper right-hand corner of M^i.)

It is shown in [5] that the growth function of G_{mix} is not $\mathbb{Z}$-rational. Consequently, we obtain the following result which shows the rather amazing possibilities inherent in Watson-Crick $D0L$ systems.

Theorem 1 *The class of growth functions of Watson-Crick $D0L$ systems contains functions that are not $\mathbb{Z}$-rational.*

We will now define a decision problem that will turn out to be very significant later in the paper.

Problem $\mathbb{Z}_{pos}$. *Given a $\mathbb{Z}$-rational sequence $z(i)$ (by some effective means such as a matrix or two $D0L$ systems), decide whether or not $z(i) \geq 0$ holds for all $i \geq 0$.*

The decidability status of $\mathbb{Z}_{pos}$ is open. The reader is referred to [4] and [7] for a further discussion and equivalent versions of the problem.

4 Nonperiodicity. Watson-Crick walks

Let u_i, $i \geq 0$, be the sequence of an ordinary $D0L$ system (Σ, g, u_0). It is well-known that the sequence $alph(u_i)$, $i \geq 0$, of the minimal alphabets of the words is almost periodic. Indeed, this follows immediately from the fact that $alph(u_{i+1})$ is always determined by $alph(u_i)$ and, consequently, if $alph(u_{i+k}) = alph(u_i)$ holds for some i and k, then $alph(u_{i+k+j}) = alph(u_{i+j})$ holds for all j. The sequence of alphabets can be depicted by the following directed graph, where i and k are assumed to be minimal:

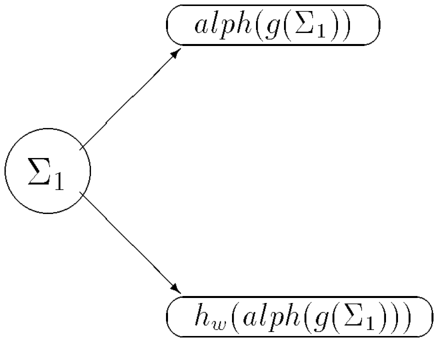
$$\begin{tikzcd}
{} \arrow[r] & \boxed{alph(u_0)} \arrow[r] & \boxed{alph(u_1)} \arrow[r] & \cdots \arrow[r] & \boxed{alph(u_i)} \arrow[r] & \cdots \arrow[r] & \boxed{alph(u_{i+k-1})} \arrow[ll, bend left]
\end{tikzcd}$$

An l-step walk in this graph leads to $alph(u_l)$, for any l.

The situation is quite different for Watson-Crick $D0L$ systems $G = (\Sigma, g, w_0)$. For a subalphabet Σ_1 of Σ, we denote by $alph(g(\Sigma_1))$ the union of the alphabets $alph(g(a))$, where a ranges over Σ_1. The notation $h_W(\Sigma_1)$, as well as the commuting relation

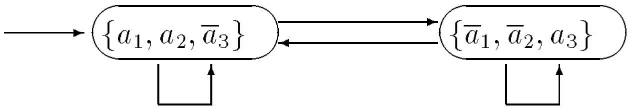
$$alph(h_W(g(\Sigma_1))) = h_W(alph(g(\Sigma_1)))$$

are obvious. We now associate a directed graph H_G to G as follows. The nodes of H_G are labeled by subsets of Σ. The initial node of H_G is labeled by $alph(w_0)$. Whenever a node labeled by $\Sigma_1 \subseteq \Sigma$ has been constructed, two new nodes labeled by $alph(g(\Sigma_1))$ and $h_W(alph(g(\Sigma_1)))$ are created (provided they do not already exist) and an arrow from the node labeled by Σ_1 to both of them is added:



The procedure is continued as long as new nodes can be created in this fashion. For a DNA-like alphabet Σ with $2n$ letters, the graph H_G contains at most 2^{2n} nodes. In most cases the cardinality of the set of nodes is much smaller.

For the system G_{mix} discussed in Section 3, we obtain the simple graph



Contrary to the case of ordinary $D0L$ systems, the graph H_G does not yet determine the sequence of alphabets $alph(w_i)$, $i = 0, 1, \ldots$, where the words w_i, $i = 0, 1, \ldots$, constitute $S(G)$. The graph gives many possibilities for walks but only one of them yields the proper sequence of alphabets.

Definition 2 *The* Watson-Crick walk $W(H_G)$ *in the graph* H_G *begins from the node labeled by* $alph(w_0)$. *If the i-th step,* $i \geq 0$, *in* $W(H_G)$ *has led to the node labeled by* $alph(w_i)$, *then the* $(i+1)$*st step leads to the node labeled by* $alph(g(w_i))$ *or* $h_W(alph(g(w_i)))$, *depending on whether* $g(w_i)$ *is in* PUR *or* PYR.

The following result is immediate from the definitions.

Lemma 2 *The Watson-Crick walk* $W(H_G)$ *is uniquely determined and follows a proper path in* H_G. *For any* $i \geq 0$, *the node reached in* $W(H_G)$ *after the i-th step is labeled by* $alph(w_i)$, *where the words* $w_0, w_1, \ldots$ *constitute the sequence* $S(G)$.

It is understood in the discussion above that the 0-th step means that no steps have been taken and the position is at the initial node.

The Watson-Crick walk $W(H_{G_{mix}})$ can be characterized as follows. The arrow leading from the initial node to itself is never used. The initial node is reached after $0, 3, 10, 29$ and, in general, after $3^{i+1} + i$ steps, for $i \geq 0$. Hence the sequence of alphabets of the words in $S(G)$ is not ultimately periodic. Moreover, since the two alphabets involved are disjoint, we get the following results. The results are opposite to those concerning ordinary $D0L$ systems.

Theorem 2 *Let G be a Watson-Crick $D0L$ system. The alphabets of the words in $S(G)$ do not necessarily form am ultimately periodic sequence, and neither do the prefixes or suffixes of any chosen length.*

Both of the nodes of $H_{G_{mix}}$ have been visited already after one step in the walk $W(H_{G_{mix}})$. In general the situation is much more complicated. The graph H_G is constructed on the basis of the alphabets alone, starting from $alph(w_0)$. A specific edge might never be used in the walk $W(H_G)$, or might be used at a late stage. It is also possible that a specific node of H_G is never reached in $W(H_G)$. We will return below in Section 6 to the following problem.

Reachability problem for graphs H_G. *Given a Watson-Crick $D0L$ system G and a node x in the graph H_G, decide whether or not the Watson-Crick walk $W(H_G)$ passes through x.*

5 Stability

We say that a Watson-Crick $D0L$ system $G = (\Sigma, g, w_0)$ is *stable* if the complementarity transition never takes place in the sequence $S(G)$, that is, the sequence consists of the words $g^i(w_0)$, $i \geq 0$. Clearly, stability is a desirable property because, in the presence of stability, the results known for $D0L$ sequences hold. It is easy to construct examples of stable and nonstable systems. However, in general stability is not an easily decidable property. Indeed, we will show in this section that, from an algorithmic point of view, the problem of stability is equivalent to the problem $\mathbb{Z}_{pos}$.

Theorem 3 *Any algorithm for solving the stability of a given Watson-Crick $D0L$ system can be converted to an algorithm for solving the problem $\mathbb{Z}_{pos}$, and conversely.*

Proof: Assume first that we know an algorithm $\mathcal{A}$ for solving the stability problem of Watson-Crick $D0L$ systems. Let $z(i)$, $i = 0, 1, \ldots$, be a given $\mathbb{Z}$-rational sequence. We show how to decide whether or not $z(i) \geq 0$ holds for all $i \geq 0$. We may assume that $z(0) \geq 0$ because, otherwise, the answer to our question is immediately negative. Since the sequence $z(i)$ is given by some effective means, we may construct two $D0L$ systems $G_j = (\Sigma_j, g_j, u_0^{(j)})$ with growth sequences $d_j(i)$, $j = 1, 2$, such that $z(i) = d_1(i) - d_2(i)$ holds for all $i \geq 0$. Without affecting the length sequences $d_j(i)$, we may rename the letters of G_j in such a way that

$$\Sigma_1 = \{a_1, \ldots, a_n\}, \quad \Sigma_2 = \{\overline{a_1}, \ldots, \overline{a_m}\}.$$

Moreover, we may assume that $m = n$ because if this is not the case originally, we add new letters to the smaller alphabet and extend the definition of the appropriate morphism for the new letters. (It is irrelevant how the extension is defined.)

Denote now $\Sigma = \Sigma_1 \cup \Sigma_2$ and let $g : \Sigma^* \to \Sigma^*$ be the morphism whose restriction to Σ_j^* equals g_j, for $j = 1, 2$. Then $G = (\Sigma, g, u_0^{(1)} u_0^{(2)})$ is a Watson-Crick $D0L$ system. In particular, because $z(0) \geq 0$, we have $|u_0^{(1)}| \geq |u_0^{(2)}|$, and hence the axiom is in PUR. We now apply the algorithm $\mathcal{A}$ to settle whether or not G is stable. (Observe that before the first complementarity transition every purine appears before every pyrimidine in words of $S(G)$. Because, for all i, $z(i) = d_1(i) - d_2(i)$ and $d_1(i)$ (resp. $d_2(i)$) equals the number of purines (resp. pyrimidines) in the i-th word of $S(G)$, $i \geq 0$, we conclude that $z(i) \geq 0$ holds for all i iff G is stable.

Conversely, assume that we have an algorithm for solving $\mathbb{Z}_{pos}$. Let $G = (\Sigma, g, w_0)$ be a given Watson-Crick $D0L$ system. We show how to decide whether or not G is stable. Observe that the right sides of the rules resulting from g may contain purines and pyrimidines in an arbitrary fashion and, consequently, no difference $d_1(i) - d_2(i)$ resembling the one in the first part of the proof is directly available. However, we may proceed as follows. Let M be the growth matrix of G, viewed as an ordinary $D0L$ system. Assume that $\Sigma = \{a_1, \ldots, a_n, \overline{a_1}, \ldots, \overline{a_n}\}$ and that the rows and columns of M correspond to the letters of Σ in the order indicated. Let π be the $2n$-dimensional row vector resulting from w_0, and let η be the $2n$-dimensional column vector whose first n components equal 1, and the remaining components equal -1. Then it is easy to see that, for any $i \geq 0$, the value $z(i) = \pi M^i \eta$ equals the difference between the numbers of (occurrences of) purines and pyrimidines in the word $g^i(w_0)$. But clearly the function $z(i)$, having the matrix representation indicated, is $\mathbb{Z}$-rational. We apply our algorithm to decide whether or not $z(i) \geq 0$ holds for all i. The system G is stable iff the answer is positive. $\square$

6 Reachability and Appearance of Letters

The preceding section already indicated the difficulties inherent in characterizing the Watson-Crick $D0L$ sequences. Of course the first problem we encounter in such a characterization is the problem of stability, and this problem we showed to be equivalent to a celebrated open problem. We will now consider some other simply formulated decision problems. From an algorithmic point of view, they turn out to be at least as hard as $\mathbb{Z}_{pos}$. In fact, in their case, we are not able to perform the reduction in the other direction. Consider first the following

Letter-appearance problem. Given a Watson-Crick $D0L$ system $G = (\Sigma, g, w_0)$ and a letter $a \in \Sigma$, decide whether or not a appears in some word in the sequence $S(G)$. □

The analogous problem has been widely considered for various language families $\mathcal{L}$. Clearly, the letter-appearance problem is decidable for a family $\mathcal{L}$ if the emptiness problem is decidable for $\mathcal{L}$ and $\mathcal{L}$ is closed under intersection with regular sets. In our case we obtain first the following immediate result.

Theorem 4 *An algorithm for solving the reachability problem for graphs H_G can be converted to an algorithm for solving the letter-appearance problem.*

Proof: Given a Watson-Crick $D0L$ system $G = (\Sigma, g, w_0)$ and a letter $a \in \Sigma$, we just check whether or not some node labeled by a subalphabet containing a is reachable in H_G. □

We have not been able to perform the reduction of Theorem 4 in the other direction. If we can settle the letter-appearance problem, we can of course decide whether or not each of the letters belonging to the subalphabet labeling a node in H_G actually appears in $S(G)$. However, all of these letters may individually appear early in the Watson-Crick walk $W(H_G)$, whereas the subalphabet we are interested in appears very late.

Thus the letter-appearance problem is "easy" compared with the reachability problem. We now establish another reduction result, showing that the letter-appearance problem is "hard".

Theorem 5 *An algorithm for solving the letter-appearance problem can be converted to an algorithm for solving $\mathbb{Z}_{pos}$.*

Proof: Assume that $\mathcal{A}$ is an algorithm for solving the letter-appearance problem. Let $z(i)$, $i \geq 0$, be a given $\mathbb{Z}$-rational sequence. We first represent $z(i)$ as the difference $z(i) = d_1(i) - d_2(i)$, where $d_j(i)$, $i \geq 0$, is the length sequence generated by the $D0L$ system $G_j = (\Sigma_j, g_j, u_0^{(j)})$, $j = 1, 2$. We may again assume that $z(0) \geq 0$ and, consequently, $|u_0^{(1)}| \geq |u_0^{(2)}|$. We may also assume that the alphabets Σ_1 and Σ_2 are disjoint (but now we do not need the assumption that they are of the same cardinality). We denote, further, by Σ_j' the alphabet consisting of the primed versions a' of the letters a of Σ_j, $j = 1, 2$.

Consider now the Watson-Crick $D0L$ system

$$G = ((\Sigma_1 \cup \Sigma_2') \cup (\Sigma_2 \cup \Sigma_1'), g, u_0^{(1)} u_0^{(2)}).$$

The letters of the alphabet $\Sigma_1 \cup \Sigma_2'$ (resp. $\Sigma_2 \cup \Sigma_1'$) are purines (resp. pyrimidines). The correspondence between purines and pyrimidines is defined in the obvious fashion. Thus, if b_1 is the first letter of Σ_2, then b_1' is the complementary purine in Σ_2'. The morphism g is defined in such a way that its restriction to Σ_j^* equals g_j, $j = 1, 2$. (It is irrelevant how g is defined on $\Sigma_1' \cup \Sigma_2'$.)

We now apply the algorithm $\mathcal{A}$ to decide whether or not one of the letters of Σ_2' appears in the sequence $S(G)$. We claim that $z(i) \geq 0$ holds for all i exactly in case none of the letters of Σ_2' appears in the sequence $S(G)$. Indeed, such a letter appears exactly in case a complementarity transition has taken place which, in turn, happens iff the number of letters of Σ_2 (pyrimidines) exceeds that of letters of Σ_1 (purines) in the iterations of the morphism g on the axiom $u_0^{(1)} u_0^{(2)}$. But the last condition means that $z(i)$ assumes a negative value. (Observe that we have to apply $\mathcal{A}$ to all letters of Σ_2' rather than, for instance, only to b_1'. This is due to the fact that we cannot be sure that the letter b_1 occurs in the i-th word in the sequence $S(G_2)$ whose length exceeds that of the i-th word in the sequence $S(G_1)$.) $\square$

We have not been able to prove the converse of Theorem 5. The following remarks illustrate the difficulties involved and, more generally, the complicated structure of Watson-Crick walks.

Suppose we know an algorithm for solving $\mathbb{Z}_{pos}$ and want to solve the letter-appearance problem. By Theorem 3, we have an algorithm for solving the stability problem. We are given a Watson-Crick $D0L$ system $G = (\Sigma, g, w_0)$ and a letter $b \in \Sigma$. We have to decide whether or not b appears in the sequence $S(G)$. We first apply our algorithm to decide whether or not G is stable. If it is, our question concerning b can be immediately settled because we are dealing with a $D0L$ sequence. If we get the answer that G is not stable, we compute the sequence $S(G)$ until a complementarity transition occurs. The first word after the transition is taken as the new axiom, and the question of stability is asked again. The procedure is continued. If a "yes" answer to the stability question is returned at some stage, we are through. If we continue getting "no" answers and we do not encounter b in $S(G)$, we do not know when to stop. Even if we pass through same node in H_G several times in our Watson-Crick walk (as we necessarily do when we have sufficiently many complementarity transitions), the walk might take different route at later stages, and b might still come up!

Similar difficulties are inherent in the **problem of ultimate stability:** given a Watson-Crick $D0L$ system $G = (\Sigma, g, w_0)$, decide whether or not there is some word w_k in the sequence $S(G)$ such that the system $G' = (\Sigma, g, w_k)$ is stable. Ultimate stability is equivalent to the fact that only finitely many complementarity transitions occur in the sequence $S(G)$.

7 Ghost letters and dual sequences

We will now consider some further properties of Watson-Crick $D0L$ sequences. Since $D0L$ sequences have been very widely studied in the past, the exact characterization of the difference between $D0L$ and Watson-Crick $D0L$ sequences is a very central issue here. The following two decision problems constitute a first step in such a characterization.

$D0L$-ness. Decide whether or not a given Watson-Crick $D0L$ sequence is a $D0L$ sequence.

Watson-Crick-ness. Decide whether or not a given $D0L$ sequence is a Watson-Crick $D0L$ sequence.

The key issue in the $D0L$-ness problem is the problem of stability and, thus, we are back in Theorem 3. If a Watson-Crick $D0L$ sequence is stable, it is a $D0L$ sequence. Also some nonstable sequences can be $D0L$ sequences, for instance, the sequence generated by the Watson-Crick $D0L$ system

$$(\{a_1, \overline{a_1}\}, \{a_1 \to \overline{a_1}, \overline{a_1} \to a_1\overline{a_1}^2\}, a_1).$$

A complementarity transition takes place at every step but the resulting sequence, consisting of a_1's only, is clearly also a $D0L$ sequence.

It was pointed out already in Section 3 that the Watson-Crick-ness problem always has a trivial solution. We only extend the given $D0L$ system $G = (\Sigma, g, u_0)$ to a Watson-Crick $D0L$ system $G' = (\Sigma \cup \overline{\Sigma}, g', u_0)$, where g' coincides with g in Σ and is defined arbitrarily for the barred letters in $\overline{\Sigma}$. Then G' is stable and $S(G') = S(G)$. However, this solution is not satisfactory because all pyrimidines in G' are only "ghosts": they never appear in the sequence $S(G')$.

Consider, for instance, the $D0L$ system $G = (\{a, b\}, \{a \to b, b \to a\}, a)$. The sequence

$$S(G): \quad a, b, a, b, a \ldots$$

is not a Watson-Crick $D0L$ sequence if we do not allow all pyrimidines being "ghosts". In fact, a word in a Watson-Crick $D0L$ sequence can never consist of pyrimidines alone and, thus, neither a nor b can be a pyrimidine. Thus, if we exclude the trivial solution, some $D0L$ sequences are not Watson-Crick. Observe, however, that $S(G)$ is generated by the Watson-Crick $D0L$ system $(\{a, b, \overline{a}, \overline{b}\}, \{a \to \overline{b}, b \to \overline{a}, \overline{a} \to \overline{a}, \overline{b} \to \overline{b}\}, a)$, where pyrimidines are only "semi-ghosts": they appear before the complementarity transition.

Thus, we say that a letter a in the alphabet Σ of a Watson-Crick $D0L$ system G is a *ghost* if a never appears in the sequence $S(G)$. (In view of Theorem 5, it is a rather tricky task to tell whether or not a given letter is a ghost.) A Watson-Crick $D0L$ sequence is *nontrivial* if there is at least one pyrimidine which is not a ghost. Nontrivial solutions of the Watson-Crickness problem are characterized in the following theorem. We denote by $|w|_\Delta$ the number of occurrences of letters of Δ in w.

Theorem 6 *The sequence $S(G)$ of a $D0L$ system $G = (\Sigma, g, u_0)$ is a nontrivial Watson-Crick $D0L$ sequence iff the following condition is satisfied:*

(i) There is a letter a appearing in $S(G)$ such that $|u_i|_{\{a\}} \leq |u_i|_{\Sigma - \{a\}}$ holds for all u_i in $S(G)$.

An algorithm for $\mathbb{Z}_{pos}$ can be converted to an algorithm to settle (i).

Proof: Assume first that the condition (i) holds. We convert G to a Watson-Crick $D0L$ system G' as follows. The letter a is viewed as a pyrimidine. All letters of $\Sigma - \{a\}$ are purines. Finally, $card(\Sigma)$ new ghost letters are added (to complete Σ to a DNA-like alphabet) and the definition of g is extended in an arbitrary fashion to cover them. We have $S(G') = S(G)$, and $S(G')$ is a stable nontrivial Watson-Crick $D0L$ sequence.

Assume next that (i) does not hold. Then if $S(G)$ is viewed as a Watson-Crick $D0L$ sequence, no letter of Σ can be a pyrimidine and, consequently, all pyrimidines are ghosts.

Since the condition (i) concerns the difference of two $\mathbb{Z}$-rational sequences, the last sentence of the theorem follows. $\square$

The family of $D0L$ languages, as well as families of L languages in general, has very weak closure properties. The same holds true for the family of Watson-Crick $D0L$ languages. There are also not many operations defined on sequences that preserve the property of Watson-Crick-ness.

The *complementary* or *dual* sequence consists of the complementary words of the words of the given sequence. If $S(G)$ is defined by the Watson-Crick $D0L$ system $G = (\Sigma, g, w_0)$, then its complementary sequence is defined by the *dual* system $\overline{G} = (\Sigma, \overline{g}, \overline{w_0})$, where $\overline{g}(\overline{a}) = \overline{g(a)}$ for all a in Σ. (Recall our convention concerning double bars!) In the dual system, a majority of purines (rather than pyrimidines) acts as a trigger. We leave to the reader the inductive verification of the fact that the sequence $S(\overline{G})$ consists of words complementary to those in $S(G)$.

8 Sequence Equivalence. Conclusion

We have investigated various language-theoretic phenomena associated to the Watson-Crick complementarity, mainly within the framework of $D0L$ systems. The significance to DNA computing of such studies of operational complementarity is not yet clear. However, we hope that it has become evident from our considerations that such studies may enhance traditional formal language theory by new ideas and techniques and, consequently, are of interest on their own right. A simply defined notion, such as that of a Watson-Crick walk, may lead to really challenging problems.

Many challenging problems remain outside the scope of this paper. One of them is the sequence equivalence problem for Watson-Crick $D0L$ systems, whose decidability status is open. As constructs Watson-Crick $D0L$ systems do not differ from ordinary $D0L$ systems. It is possible to give examples of systems G_1 and G_2 that are sequence equivalent as $D0L$ systems but not as Watson-Crick $D0L$ systems, and, vice versa, examples of systems G_1' and G_2' that are sequence equivalent as Watson-Crick $D0L$ systems but not as $D0L$ systems.

References

[1] L. M. Adleman, Molecular computation of solutions to combinatorial problems. *Science*, 266 (1994), 1021-1024.

[2] J. Harrison, Morphic Congruences and $D0L$ Languages. *Theoret. Comput. Sci.* 134 (1994), 537-544.

[3] J. Harrison, Dynamical Properties of $PWD0L$ Systems. *Theoret. Comput. Sci.* 14 (1995), 269-284.

[4] W. Kuich, A. Salomaa, *Semirings, Automata, Languages.* Springer-Verlag, Berlin, Heidelberg, 1986.

[5] V. Mihalache, A. Salomaa, Lindenmayer and DNA: Watson-Crick $D0L$ systems, *EATCS Bulletin* 62 (1997), 160-175.

[6] Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, Springer-Verlag, Berlin, Heidelberg, New York, 1998.

[7] G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems.* Academic Press, New York, London, 1980.

[8] G. Rozenberg, A. Salomaa, Watson-Crick complementarity, universal computations and genetic engineering. *Leiden University, Computer Science Technical Report* (1996), 96-28.

[9] G. Rozenberg, A. Salomaa (ed.), *Handbook of Formal Languages, Vol. 1-3.* Springer-Verlag, Berlin, Heidelberg, New York, 1997.

[10] A. Salomaa, Turing, Watson-Crick and Lindenmayer. Aspects of DNA Complementarity. In C.S. Calude, J. Casti, M.J. Dinneen, eds., *Unconventional Models of Computation*, Springer-Verlag, Singapore, Berlin, Heidelberg (1998), 94-107.

Turku Centre for Computer Science
Lemminkäisenkatu 14
FIN-20520 Turku
Finland

http://www.tucs.abo.fi

University of Turku
- **Department of Mathematical Sciences**

Åbo Akademi University
- **Department of Computer Science**
- **Institute for Advanced Management Systems Research**

Turku School of Economics and Business Administration
- **Institute of Information Systems Science**