Combinatorial Aspects of Minimal DNA Expressions

Rudy van Vliet, Hendrik Jan Hoogeboom, and Grzegorz Rozenberg

Leiden Institute of Advanced Computer Science (LIACS),
Leiden University,
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
{rvvliet, hoogeboo, rozenber}@liacs.nl

Abstract. We describe a formal language/notation for DNA molecules that may contain nicks and gaps. The elements of the language, DNA expressions, denote formal DNA molecules. Different DNA expressions may denote the same formal DNA molecule. We analyse the shortest DNA expressions denoting a given formal DNA molecule: what is their length, how are they constructed, how many of them are there, and how can they be characterized.

1 Introduction

Since the discovery of the structure and function of DNA molecules, DNA has become an 'intense' research topic among biologists and biochemists. Formal study of computational properties of DNA really began when Head [1987] defined formal languages consisting of strings that can be modified by operations based on the way that restriction enzymes process DNA molecules. Theoretical computer scientists explored the generative power and other properties of such languages, see, e.g., [Kari et al., 1996] and [Head et al., 1997]. The interest of the computer science community in the computational potential of DNA was boosted, when Adleman [1994] described a solution of an instance of the directed Hamiltonian path problem using DNA, enzymes and standard biomolecular operations. Since then, research on DNA computing is really flourishing, see, e.g., [Hagiya & Ohuchi, 2003], [Chen & Reif, 2004] and [Păun et al., 1998]. Recent developments include research on computations in living cells, see, e.g., [Landweber & Kari, 1999], [Daley et al., 2004] and [Ehrenfeucht et al., 2004].

Neither in the theoretical, nor in the applied publications, much attention is paid to the notation used to denote DNA molecules – exceptions are [Boneh et al., 1996] and [Li, 1999]. In most cases, one simply uses the standard double-string notation (like $\begin{smallmatrix} \text{ACATG} \\ \text{TGTAC} \end{smallmatrix}$) to describe a double-stranded DNA molecule.

In this paper, we describe a concise and precise notation for DNA molecules, based on the letters A, C, G and T and three operators $\uparrow$, $\downarrow$ and $\updownarrow$ (to be

C. Ferretti, G. Mauri and C. Zandron (Eds.): DNA10, LNCS 3384, pp. 375–388, 2005.

pronounced as *uparrow*, *downarrow* and *updownarrow*, respectively). The resulting DNA expressions denote formal DNA molecules – a formalization of DNA molecules. We do not only account for perfect double-stranded DNA molecules, but also for single-stranded DNA molecules and for double-stranded DNA molecules containing nicks (missing phosphodiester bonds between adjacent nucleotides in the same strand) and gaps (missing nucleotides in one of the strands). The notation is the first step towards a formal description of more complex DNA molecules. The ultimate goal is to also describe the secondary structure of DNA.

Our three operators bear some resemblance to the operators used in [Boneh et al., 1996] and [Li, 1999], but their functionality is quite different. The operator $\uparrow$ acts as a kind of ligase for the upper strands: it creates upper strands and connects the upper strands of its arguments. The operator $\downarrow$ is the analogue for lower strands. Finally, $\updownarrow$ fills up the gap(s) in its argument. The effects of the operators do not perfectly correspond to the effects of existing techniques in real-life DNA synthesis. Yet, the operators are useful to describe certain types of DNA molecules.

In our formal language, different DNA expressions may denote the same formal DNA molecule. We examine which DNA expressions are minimal, i.e., have the shortest length among DNA expressions denoting the same formal DNA molecule, and what their length is. Moreover, there may be different minimal DNA expressions denoting the same formal DNA molecule. We calculate the number of these minimal DNA expressions. Finally, we give a characterization of minimal DNA expressions, which makes it easy to check whether or not a given DNA expression is minimal.

Due to space limitations, we omit the formal proofs of the results we present in this paper. For some results, however, we will provide an intuitive argumentation. The proofs can be found in [Van Vliet, 2004].

2 $\mathcal{N}$-Words and Formal DNA Molecules

The letters A, C, G and T, denoting the four nucleotides that a DNA molecule consists of, are also important building blocks of our language. We use $\mathcal{N}$ to denote this alphabet: $\mathcal{N} = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$. The elements of $\mathcal{N}$ are called $\mathcal{N}$-*letters*. A non-empty string over $\mathcal{N}$ is called an $\mathcal{N}$-word.

For an $\mathcal{N}$-word α, $c(\alpha)$ is the element-wise (non-reversed) Watson-Crick *complement* of α. For example, $c(\mathrm{ACATG}) = \mathrm{TGTAC}$.

The semantical basis of our formal language are *formal DNA molecules*. Formal DNA molecules are strings over the set $\mathcal{A}_{\triangledown\triangle} = \mathcal{A}_+ \cup \mathcal{A}_- \cup \mathcal{A}_\pm \cup \{^{\triangledown}, {}_{\triangle}\}$, where $\mathcal{A}_+ = \left\{\binom{\mathrm{A}}{-}, \binom{\mathrm{C}}{-}, \binom{\mathrm{G}}{-}, \binom{\mathrm{T}}{-}\right\}$, $\mathcal{A}_- = \left\{\binom{-}{\mathrm{A}}, \binom{-}{\mathrm{C}}, \binom{-}{\mathrm{G}}, \binom{-}{\mathrm{T}}\right\}$ and $\mathcal{A}_\pm = \left\{\binom{\mathrm{A}}{\mathrm{T}}, \binom{\mathrm{C}}{\mathrm{G}}, \binom{\mathrm{G}}{\mathrm{C}}, \binom{\mathrm{T}}{\mathrm{A}}\right\}$. The elements of $\mathcal{A}_+ \cup \mathcal{A}_- \cup \mathcal{A}_\pm$ are called $\mathcal{A}$-letters. The elements of $\mathcal{A}_+$ and $\mathcal{A}_-$ correspond to gaps in the lower strand and the upper strand, respectively. The symbols $^{\triangledown}$ and ${}_{\triangle}$ are called *nick letters*. The *upper nick*

letter $^{\triangledown}$ represents a nick in the upper strand of the DNA molecule; the *lower nick letter* $_{\triangle}$ represents a nick in the lower strand.

Not all strings over $\mathcal{A}_{\triangledown\triangle}$ are formal DNA molecules. We impose three natural conditions on the strings, which, among others, prevent the DNA molecule represented from 'falling apart'.

Definition 1. *A formal DNA molecule is a string* $X = x_1x_2\dots x_r$ *with* $r \geq 1$ *and for* $i = 1,\dots,r$, $x_i \in \mathcal{A}_{\triangledown\triangle}$, *satisfying*

- *if* $x_i \in \mathcal{A}_+$, *then* $x_{i+1} \notin \mathcal{A}_-$ $\quad (i = 1,2,\dots,r-1)$,
  if $x_i \in \mathcal{A}_-$, *then* $x_{i+1} \notin \mathcal{A}_+$ $\quad (i = 1,2,\dots,r-1)$,
- $x_1, x_r \notin \{^{\triangledown}, _{\triangle}\}$,
- *if* $x_i \in \{^{\triangledown}, _{\triangle}\}$, *then* $x_{i-1}, x_{i+1} \in \mathcal{A}_\pm$ $\quad (i = 2,3,\dots,r-1)$.

A formal DNA molecule that does not contain nick letters, is called *nick free*.

Examples of formal DNA molecules are

$$X_1 = \binom{\mathrm{A}}{\mathrm{T}}\binom{\mathrm{C}}{\mathrm{G}}\binom{\mathrm{A}}{\mathrm{T}}\binom{\mathrm{T}}{\mathrm{A}}\binom{\mathrm{G}}{\mathrm{C}}, \tag{1}$$

$$X_2 = \binom{\mathrm{A}}{\mathrm{T}}^{\triangledown}\binom{\mathrm{C}}{\mathrm{G}}\binom{\mathrm{A}}{\mathrm{T}}_{\triangle}\binom{\mathrm{T}}{\mathrm{A}}\binom{\mathrm{G}}{-}, \text{ and} \tag{2}$$

$$X_3 = \binom{-}{\mathrm{T}}\binom{\mathrm{C}}{\mathrm{G}}\binom{\mathrm{A}}{-}\binom{\mathrm{T}}{-}\binom{\mathrm{G}}{\mathrm{C}}. \tag{3}$$

Both X_1 and X_3 are nick free. We assume that if two nucleotides in the same strand are separated by a gap (as is the case for the G and the C in the lower strand of X_3), then they are not connected by a (long) phosphodiester bond.

The following strings over $\mathcal{A}_{\triangledown\triangle}$ are no formal DNA molecules, because they violate one of the three conditions from Definition 1.

$$X_1' = \binom{-}{\mathrm{T}}\binom{-}{\mathrm{G}}\binom{\mathrm{A}}{-}\binom{\mathrm{T}}{\mathrm{A}}\binom{\mathrm{G}}{\mathrm{C}},$$

$$X_2' = {}_{\triangle}\binom{\mathrm{A}}{\mathrm{T}}\binom{\mathrm{C}}{\mathrm{G}}\binom{\mathrm{A}}{\mathrm{T}}\binom{\mathrm{T}}{\mathrm{A}}\binom{\mathrm{G}}{-}, \text{ and}$$

$$X_3' = \binom{-}{\mathrm{T}}^{\triangledown}\binom{\mathrm{C}}{\mathrm{G}}\binom{\mathrm{A}}{-}^{\triangledown}\binom{\mathrm{T}}{-}\binom{\mathrm{G}}{\mathrm{C}}.$$

Often, we simplify the notation of a formal DNA molecule. Let $X = x_1\dots x_r$ for some $r \geq 1$ be a formal DNA molecule. If two or more consecutive symbols of X are elements of $\mathcal{A}_+$, say $x_i\dots x_j = \binom{a_i}{-}\dots\binom{a_j}{-}$ with $1 \leq i < j \leq r$, then we may substitute $x_i\dots x_j$ by $\binom{a_i\dots a_j}{-}$. Analogously, we may substitute consecutive elements of $\mathcal{A}_-$ and consecutive elements of $\mathcal{A}_\pm$. When we simplify the *notation* of a formal DNA molecule, we do not modify the formal DNA molecule itself. In particular, it remains a string over $\mathcal{A}_{\triangledown\triangle}$.

A non-empty sequence of elements of $\mathcal{A}_+$ is called an *upper $\mathcal{A}$-word*. Analogously, we have a *lower $\mathcal{A}$-word* (with elements of $\mathcal{A}_-$) and a *double $\mathcal{A}$-word* (with elements of $\mathcal{A}_\pm$). These notions are needed to define the decomposition of a formal DNA molecule:

Definition 2. *Let X be a formal DNA molecule. The decomposition of X is the sequence $x'_1, \ldots, x'_k$ of $k \geq 1$ non-empty strings over $\mathcal{A}_{\triangledown\triangle}$ such that*

- $X = x'_1 \ldots x'_k$,
- *for $i = 1, \ldots, k$, x'_i is either an upper $\mathcal{A}$-word, or a lower $\mathcal{A}$-word, or a double $\mathcal{A}$-word, or a nick letter, and*
- *for $i = 1, \ldots, k-1$, if x'_i is an upper $\mathcal{A}$-word, then x'_{i+1} is not an upper $\mathcal{A}$-word, and analogously for lower $\mathcal{A}$-words and double $\mathcal{A}$-words.*

Hence, the decomposition of X cannot be simplified any further. For the ease of notation, we will in general write $x'_1 \ldots x'_k$ instead of $x'_1, \ldots, x'_k$.

For example, the decompositions of the formal DNA molecules X_1, X_2 and X_3 are

$$\begin{aligned} X_1 &= \binom{\text{ACATG}}{\text{TGTAC}} && \text{(with } k = 1\text{)}, \\ X_2 &= \binom{\text{A}}{\text{T}}^{\triangledown}\binom{\text{CA}}{\text{GT}}_{\triangle}\binom{\text{T}}{\text{A}}\binom{\text{G}}{-} && \text{(with } k = 6\text{), and} \\ X_3 &= \binom{-}{\text{T}}\binom{\text{C}}{\text{G}}\binom{\text{AT}}{-}\binom{\text{G}}{\text{C}} && \text{(with } k = 4\text{).} \end{aligned}$$

If $x'_1 \ldots x'_k$ for some $k \geq 1$ is the decomposition of a formal DNA molecule X, then the substrings x'_i are called the *components* of X. For $i = 1, \ldots, k$, if x'_i is an upper $\mathcal{A}$-word (lower $\mathcal{A}$-word or double $\mathcal{A}$-word), then it is called an *upper component* (*lower component* or *double component*, respectively) of X. If x'_i is either an upper component or a lower component, then we may also call it a *single-stranded component* of X.

Because, by definition, an upper component of a formal DNA molecule X cannot be followed by a lower component and vice versa, and nick letters occurring in X must be preceded and followed by a double component, we have

Lemma 3. *For each formal DNA molecule X, the decomposition of X is an alternating sequence of double components on the one hand and other types of components on the other hand.*

For example, the decomposition of X_2 consists of a double component, an upper nick letter, a double component, a lower nick letter, a double component and an upper component, respectively.

We define three functions on formal DNA molecules. Let $X = x_1 \ldots x_r$ for some $r \geq 1$ be a formal DNA molecule. Then $L(X) = x_1$ and $R(X) = x_r$. Hence, the functions L and R give the leftmost symbol and the rightmost symbol of a formal DNA molecule. Further, $|X|_{\mathcal{A}}$ counts the $\mathcal{A}$-letters occurring in X. For example, $L(X_2) = \binom{\text{A}}{\text{T}}$, $R(X_2) = \binom{\text{G}}{-}$ and $|X_2|_{\mathcal{A}} = 5$.

3 DNA Expressions

The elements of our language are called *DNA expressions*. The semantics of a DNA expression E is a formal DNA molecule, denoted by $\mathcal{S}(E)$. In this paper,

$$\mathcal{S}\left(\left\langle\uparrow \begin{matrix}\mathrm{C}\\ \mathrm{G}\end{matrix}\ \mathrm{AT}\ \begin{matrix}\overset{\triangledown}{\mathrm{G}}\mathrm{C}\\ \mathrm{CG}\end{matrix}\right\rangle\right) = \begin{matrix}\mathrm{CATGC}\\ \mathrm{G}\quad\mathrm{CG}\end{matrix} \qquad \mathcal{S}\left(\left\langle\uparrow \begin{matrix}\mathrm{A}\\ \mathrm{T}\end{matrix}\quad \begin{matrix}\mathrm{T}\\ \mathrm{A}\end{matrix}\right\rangle\right) = \begin{matrix}\mathrm{AT}\\ \mathrm{T}\overset{\triangle}{\mathrm{A}}\end{matrix} \quad \text{(a)}$$

$$\mathcal{S}\left(\left\langle\downarrow \mathrm{T}\ \begin{matrix}\mathrm{CATGC}\\ \mathrm{G}\quad\mathrm{CG}\end{matrix}\quad \begin{matrix}\mathrm{AT}\\ \mathrm{T}\overset{\triangle}{\mathrm{A}}\end{matrix}\right\rangle\right) = \begin{matrix}\mathrm{CATGC}\overset{\triangledown}{\mathrm{A}}\mathrm{T}\\ \mathrm{TG}\quad\mathrm{CGTA}\end{matrix} \quad \text{(b)}$$

$$\mathcal{S}\left(\left\langle\updownarrow \begin{matrix}\mathrm{CATGC}\overset{\triangledown}{\mathrm{A}}\mathrm{T}\\ \mathrm{TG}\quad\mathrm{CGTA}\end{matrix}\right\rangle\right) = \begin{matrix}\mathrm{ACATGC}\overset{\triangledown}{\mathrm{A}}\mathrm{T}\\ \mathrm{TGTACGTA}\end{matrix} \quad \text{(c)}$$

Fig. 1. Examples of (a) the effect of the operator $\uparrow$; (b) the effect of the operator $\downarrow$; (c) the effect of the operator $\updownarrow$

we will describe the syntax and semantics of a DNA expression in words and by means of examples. For a formal definition, we refer to [Van Vliet, 2004]. One can also define a context-free grammar that generates the DNA expressions.

DNA expressions are the result of applying the three operators $\uparrow$, $\downarrow$ and $\updownarrow$ to basic $\mathcal{N}$-words. In general, the operator $\uparrow$ can have any number $n \geq 1$ arguments $\varepsilon_1, \ldots, \varepsilon_n$, which may be $\mathcal{N}$-words or DNA expressions. The result of applying $\uparrow$ to these arguments is the DNA expression $\langle\uparrow \varepsilon_1 \ldots \varepsilon_n\rangle$. It is called an $\uparrow$-*expression*. Analogously, we may have a $\downarrow$-*expression* $\langle\downarrow \varepsilon_1 \ldots \varepsilon_n\rangle$. The operator $\updownarrow$ can have only one argument ε_1, which may again be an $\mathcal{N}$-word or a DNA expression, yielding an $\updownarrow$-*expression* $\langle\updownarrow \varepsilon_1\rangle$.

Hence, the set of all DNA expressions is a language over the alphabet $\mathcal{N} \cup \{\uparrow, \downarrow, \updownarrow, \langle\ , \rangle\}$. The length of a specific DNA expression E is defined as the number of its symbols and is denoted by $|E|$. The *outermost operator* of a DNA expression is (the occurrence of) the operator which has been performed last. For example, the outermost operator of an $\uparrow$-expression is $\uparrow$. All other occurrences of operators in a DNA expression, i.e., the occurrences in the argument(s) of the outermost operator, are called *inner occurrences*.

The effect of $\uparrow$ is the following: (1) for each argument that is an $\mathcal{N}$-word α, it produces an upper $\mathcal{A}$-word $\binom{\alpha}{-}$, (2) it removes all upper nick letters occurring in its arguments, and (3) it connects the upper strands of consecutive arguments.

Step (3) requires that for $i = 1, \ldots, n-1$, the upper strand of the formal DNA molecule X_i corresponding to argument ε_i extends at least as far to the right as the lower strand: $R(X_i)$ must not be an element of $\mathcal{A}_-$. Analogously, if X_{i+1} is the formal DNA molecule corresponding to argument ε_{i+1}, then $L(X_{i+1})$ must not be an element of $\mathcal{A}_-$. Otherwise, there would be a gap in the upper strand 'between' X_i and X_{i+1}, and we would not be able to connect the upper strands. Such natural requirements are tedious to formalize, which is why we omit a full formal definition of DNA expressions.

Lower nick letters that occur in the arguments of $\uparrow$ are not removed. On the contrary, if both $R(X_i)$ and $L(X_{i+1})$ are elements of $\mathcal{A}_\pm$, then $\uparrow$ produces a lower nick letter between X_i and X_{i+1}. Thus, the lower strands of consecutive arguments are not connected.

The simplest $\uparrow$-expression is of the form $\langle\uparrow \alpha_1\rangle$ for an $\mathcal{N}$-word α_1. Its semantics is the formal DNA molecule $\binom{\alpha_1}{-}$. Figure 1(a) shows the effect of $\uparrow$ for two less trivial examples. For the ease of understanding, we replaced the arguments

of $\uparrow$ that are DNA expressions by pictorial representations of the corresponding DNA molecules. The result of the operator is depicted in the same way. For example, the first $\uparrow$-expression has three arguments: a DNA expression, an $\mathcal{N}$-word and another DNA expression, respectively.

On the other hand, although the DNA molecules corresponding to $\begin{smallmatrix}\mathrm{ACAT}\\ \mathrm{TGT}\end{smallmatrix}$ and $\begin{smallmatrix}\mathrm{G}\\ \mathrm{AC}\end{smallmatrix}$ have matching sticky ends, $\left\langle \uparrow \begin{smallmatrix}\mathrm{ACAT}\\ \mathrm{TGT}\end{smallmatrix} \begin{smallmatrix}\mathrm{G}\\ \mathrm{AC}\end{smallmatrix} \right\rangle$ is not a DNA expression, because $L(\binom{-}{\mathrm{A}}\binom{\mathrm{G}}{\mathrm{C}}) \in \mathcal{A}_{-}$. Hence, the operator $\uparrow$ does not account for annealing. Analogously, $\left\langle \uparrow \begin{smallmatrix}\mathrm{AC}\\ \mathrm{TGT}\end{smallmatrix} \begin{smallmatrix}\mathrm{G}\\ \mathrm{AC}\end{smallmatrix} \right\rangle$ is not a DNA expression.

The effect of the operator $\downarrow$ is analogous to that of $\uparrow$. However, instead of upper $\mathcal{A}$-words, upper nick letters and upper strands, we must read lower $\mathcal{A}$-words, lower nick letters and lower strands, respectively. For step (3), we also have analogous requirements. The effect of $\downarrow$ is illustrated in Fig. 1(b).

Finally, the operator $\updownarrow$ complements its argument: it provides a complementary nucleotide for every nucleotide that is not yet complemented. Each nucleotide added is connected to its direct neighbours. The operator does not introduce or remove nick letters. The argument of $\updownarrow$ may be any $\mathcal{N}$-word or any DNA expression. If the argument is an $\mathcal{N}$-word α_1, then it is interpreted as $\langle\uparrow \alpha_1\rangle$. Hence, $\mathcal{S}(\langle\updownarrow \alpha_1\rangle) = \binom{\alpha_1}{c(\alpha_1)}$.

Figure 1(c) illustrates the effect of $\updownarrow$. A complete DNA expression denoting the formal DNA molecule from this example is

$$E = \langle\updownarrow \langle\downarrow \mathrm{T} \langle\uparrow \langle\updownarrow \mathrm{C}\rangle \mathrm{AT} \langle\downarrow \langle\updownarrow \mathrm{G}\rangle \langle\updownarrow \mathrm{C}\rangle\rangle \langle\uparrow \langle\updownarrow \mathrm{A}\rangle \langle\updownarrow \mathrm{T}\rangle\rangle\rangle\rangle\,. \tag{4}$$

It is the result of the step-by-step construction from Fig. 1.

We say that a formal DNA molecule X is *expressible*, if there exists a DNA expression E with $\mathcal{S}(E) = X$. Unfortunately, there exist formal DNA molecules that are not expressible. In fact, we have:

Theorem 4. *A formal DNA molecule is expressible, if and only if it does not both contain upper nick letters and lower nick letters.*

Hence, the formal DNA molecules X_1 and X_3 from (1) and (3) are expressible, for example by DNA expressions $\langle\updownarrow \mathrm{ACATG}\rangle$ and $\langle\downarrow \mathrm{T} \langle\uparrow \langle\updownarrow \mathrm{C}\rangle \mathrm{AT} \langle\updownarrow \mathrm{G}\rangle\rangle\rangle$, respectively. X_2, however, is not expressible.

4 The Length of a DNA Expression

Different DNA expressions may denote the same formal DNA molecule. Such DNA expressions are called *equivalent*. In fact, for each expressible formal DNA molecule X, there exist infinitely many DNA expressions denoting X. For example, it is easily verified that if E is an $\uparrow$-expression denoting X, then so is $\langle\uparrow E\rangle$. By repeating the construction, adding three symbols (two brackets and an operator) at a time, we can find arbitrarily long, equivalent DNA expressions. Hence, there is no maximal length for DNA expressions denoting a given formal

DNA molecule. There does, however, exist a minimal length. We will examine this length for all types of expressible formal DNA molecules and we will also describe the DNA expressions that achieve this length. Before we do so, we make an elementary observation:

Lemma 5. *Let E be a DNA expression denoting a formal DNA molecule X, and let p be the number of operators occurring in E. Then $|E| = 3 \cdot p + |X|_{\mathcal{A}}$.*

Because each occurrence of an operator is accompanied by an opening bracket and a closing bracket, the term $3 \cdot p$ accounts for the operators and the brackets in E. Consequently, $|X|_{\mathcal{A}}$ counts the $\mathcal{N}$-letters occurring in E. Note that this number only depends on X, and not on the specific DNA expression E.

Indeed, for the DNA expression E from (4), the number p of operators is 10, the number of $\mathcal{A}$-letters in X is 8 and $|E| = 3 \cdot 10 + 8 = 38$.

5 Lower Bounds for the Length of a DNA Expression

We first examine lower bounds for the length of a DNA expression E denoting a formal DNA molecule X. These lower bounds will be expressed in terms of some simple counting functions of X. We now introduce these counting functions.

It follows from the definition of a DNA expression that both *upper* components and *lower* nick letters are the result of an occurrence of the operator $\uparrow$. Therefore, these type of components are called $\uparrow$-*components*. Analogously, lower components and upper nick letters are called $\downarrow$-*components*.

Recall that the decomposition of a formal DNA molecule is an alternating sequence of double components on the one hand and other types of components on the other hand. If we disregard the double components, then we only have a sequence of other types of components, which are $\uparrow$-components and $\downarrow$-components. Consecutive $\uparrow$-components form a *(maximal) series* of $\uparrow$-components. Analogously, we have maximal series of $\downarrow$-components.

Definition 6. *Let X be a formal DNA molecule.*

- $T_{\uparrow}(X)$ *is the number of maximal series of $\uparrow$-components of X.*
- $T_{\downarrow}(X)$ *is the number of maximal series of $\downarrow$-components of X.*
- $n_{\updownarrow}(X)$ *is the number of double components of X.*

We illustrate this definition by the formal DNA molecule X depicted in Fig. 2. The α_i's occurring in this picture denote the $\mathcal{N}$-words determining the upper, lower and double components of X. The $\uparrow$-components of X are $\binom{\alpha_4}{-}$ (series 1), $\binom{\alpha_8}{-}$ and $\binom{\alpha_{10}}{-}$ (series 2), and $\binom{\alpha_{13}}{-}$ (series 3). The $\downarrow$-components of X are the first and the second upper nick letter (series 1), $\binom{-}{\alpha_6}$ (series 2), the third upper nick letter (series 3) and the fourth upper nick letter and $\binom{-}{\alpha_{16}}$ (series 4). Hence, $T_{\uparrow}(X) = 3$ and $T_{\downarrow}(X) = 4$. Further, $n_{\updownarrow}(X) = 10$.

Intuitively, $T_{\uparrow}(X)$ counts the *transitions* (from $\downarrow$-components) to $\uparrow$-components. It requires an occurrence of the operator $\uparrow$ to achieve this transition.

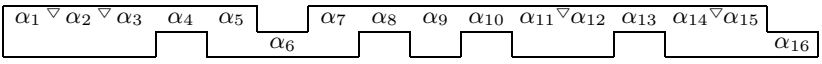

Fig. 2. Pictorial representation of a formal DNA molecule containing upper nick letters

There is, of course, an analogous interpretation of $T_{\downarrow}(X)$. Note that, unless a formal DNA molecule X only consists of a double component, hence, unless $X = \binom{\alpha_1}{c(\alpha_1)}$ for an $\mathcal{N}$-word α_1, either $T_{\uparrow}(X) > 0$, or $T_{\downarrow}(X) > 0$ (or both).

Because maximal series of $\uparrow$-components and maximal series of $\downarrow$-components alternate in a formal DNA molecule, we have

Lemma 7. *For each formal DNA molecule X, $T_{\uparrow}(X) - 1 \leq T_{\downarrow}(X) \leq T_{\uparrow}(X) + 1$.*

We can now formulate lower bounds on the lengths of DNA expressions:

Theorem 8. *Let E be a DNA expression, and let $X = \mathcal{S}(E)$.*

1. *If E is an $\uparrow$-expression, then $|E| \geq 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\updownarrow}(X) + |X|_{\mathcal{A}}$.*
2. *If E is a $\downarrow$-expression, then $|E| \geq 3 + 3 \cdot T_{\uparrow}(X) + 3 \cdot n_{\updownarrow}(X) + |X|_{\mathcal{A}}$.*
3. *If $E = \langle \updownarrow \alpha_1 \rangle$ for an $\mathcal{N}$-word α_1, then $|E| = 3 \cdot n_{\updownarrow}(X) + |X|_{\mathcal{A}}$.*
4. *If $E = \langle \updownarrow E_1 \rangle$ for a DNA expression E_1, then $|E| \geq 3 + 3 \cdot n_{\updownarrow}(X) + |X|_{\mathcal{A}}$.*

The terms $3+3 \cdot T_{\downarrow}(X)$ and $3+3 \cdot T_{\uparrow}(X)$ occurring in the first two lower bounds correspond to occurrences of the two operators $\uparrow$ and $\downarrow$. The term $3 \cdot n_{\updownarrow}(X)$ occurring in all lower bounds corresponds to occurrences of the operator $\updownarrow$, which are needed to obtain the double components of X. For example, an $\uparrow$-expression denoting a formal DNA molecule X contains at least $(1 + T_{\downarrow}(X))$ occurrences of $\uparrow$ and $\downarrow$ together, and at least $n_{\updownarrow}(X)$ occurrences of $\updownarrow$.

The symmetry between Claims 1 and 2 is due to the symmetrical effects of the operators $\uparrow$ and $\downarrow$. In later results, we will refer to this symmetry rather than fully stating a symmetrical claim.

6 Minimal DNA Expressions for a Nick Free Molecule

We are not just interested in lower bounds on the lengths of DNA expressions; we also want to be able to *construct* the shortest DNA expressions denoting a given formal DNA molecule. A DNA expression E is called *minimal*, if for every equivalent DNA expression E', $|E'| \geq |E|$.

We first consider nick free formal DNA molecules. These consist only of upper components, lower components and double components. By Theorem 4, each nick free formal DNA molecule is expressible.

Theorem 9. *Let X be a nick free formal DNA molecule.*

1. *If $X = \binom{\alpha_1}{c(\alpha_1)}$ for an $\mathcal{N}$-word α_1, then the only minimal DNA expression denoting X is $E = \langle \updownarrow \alpha_1 \rangle$, with length $|E| = 3 + |X|_{\mathcal{A}}$.*
2. *If $T_{\uparrow}(X) = T_{\downarrow}(X) \geq 1$, then each minimal DNA expression E denoting X is either an $\uparrow$-expression or a $\downarrow$-expression and has length*

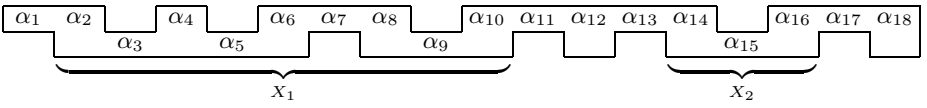

Fig. 3. Pictorial representation of a nick free formal DNA molecule X with single-stranded components. The lower components have been partitioned in submolecules X_1 and X_2 (see the construction below Theorem 9)

$$\begin{aligned} |E| &= 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\updownarrow}(X) + |X|_{\mathcal{A}} \\ &= 3 + 3 \cdot T_{\uparrow}(X) + 3 \cdot n_{\updownarrow}(X) + |X|_{\mathcal{A}}. \end{aligned}$$

3. *If $T_{\uparrow}(X) > T_{\downarrow}(X)$, then each minimal DNA expression E denoting X is an $\uparrow$-expression and has length*

$$|E| = 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\updownarrow}(X) + |X|_{\mathcal{A}}.$$

4. *If $T_{\downarrow}(X) > T_{\uparrow}(X)$, then ... (symmetric to Claim 3).*

For nick free formal DNA molecules with at least one single-stranded component, we did not mention *how to construct* the minimal DNA expressions. We will describe this construction now, in an intuitive way, by means of an example.

Consider the nick free formal DNA molecule X depicted in Fig. 3, for which $T_{\uparrow}(X) = 4$, $T_{\downarrow}(X) = 3$ and $n_{\updownarrow}(X) = 9$. Because $T_{\uparrow}(X) > T_{\downarrow}(X)$, a minimal DNA expression denoting X must be an $\uparrow$-expression. Upper components $\binom{\alpha_i}{-}$ result when $\uparrow$ has arguments that are $\mathcal{N}$-words α_i, and double components $\binom{\alpha_i}{c(\alpha_i)}$ of X can be produced efficiently by arguments of the form $\langle\updownarrow \alpha_i\rangle$. The lower components of X, however, require a special treatment. We partition the lower components of X in submolecules $X_1, X_2, \ldots, X_r$ for some $r \geq 1$, which, if possible, start with a double component preceding a maximal series of $\downarrow$-components and end with a double component succeeding a maximal series of $\downarrow$-components. If the first component of X is a $\downarrow$-component, then X_1 starts with this component. Analogously, X_r may end with a $\downarrow$-component.

For our nick free formal DNA molecule X, we may take $r = 2$ and

$$\begin{aligned} X_1 &= \tbinom{\alpha_2}{c(\alpha_2)}\tbinom{-}{\alpha_3}\tbinom{\alpha_4}{c(\alpha_4)}\tbinom{-}{\alpha_5}\tbinom{\alpha_6}{c(\alpha_6)}\tbinom{\alpha_7}{-}\tbinom{\alpha_8}{c(\alpha_8)}\tbinom{-}{\alpha_9}\tbinom{\alpha_{10}}{c(\alpha_{10})}, \\ X_2 &= \tbinom{\alpha_{14}}{c(\alpha_{14})}\tbinom{-}{\alpha_{15}}\tbinom{\alpha_{16}}{c(\alpha_{16})} \end{aligned}$$

(see also Fig. 3). We now recursively determine minimal DNA expressions E_1 and E_2 denoting X_1 and X_2, respectively. These minimal DNA expressions become arguments of the $\uparrow$-expression E we are constructing, together with $\mathcal{N}$-words α_i and $\updownarrow$-expressions $\langle\updownarrow \alpha_i\rangle$ for the upper components and double components of X which are neither in X_1, nor in X_2:

$$E = \langle\uparrow \alpha_1 E_1 \alpha_{11} \langle\updownarrow \alpha_{12}\rangle \alpha_{13} E_2 \alpha_{17} \langle\updownarrow \alpha_{18}\rangle\rangle\,.$$

Because $T_{\uparrow}(X_1) = 1 < 2 = T_{\downarrow}(X_1)$, E_1 must be a $\downarrow$-expression, which is constructed in an analogous way: the only upper component of X_1 is 'partitioned' in the submolecule $X_{1,1} = \binom{\alpha_6}{c(\alpha_6)}\binom{\alpha_7}{-}\binom{\alpha_8}{c(\alpha_8)}$. We determine a minimal DNA expression $E_{1,1}$ for $X_{1,1}$, which is, in turn, an $\uparrow$-expression:

$$E_{1,1} = \langle\uparrow \langle\updownarrow \alpha_6\rangle \alpha_7 \langle\updownarrow \alpha_8\rangle\rangle\,.$$

We then get

$$E_1 = \langle\downarrow \langle\updownarrow \alpha_2\rangle \alpha_3 \langle\updownarrow \alpha_4\rangle \alpha_5 \langle\uparrow \langle\updownarrow \alpha_6\rangle \alpha_7 \langle\updownarrow \alpha_8\rangle\rangle \alpha_9 \langle\updownarrow \alpha_{10}\rangle\rangle\,.$$

The minimal DNA expression E_2 is relatively easy to construct:

$$E_2 = \langle\downarrow \langle\updownarrow \alpha_{14}\rangle \alpha_{15} \langle\updownarrow \alpha_{16}\rangle\rangle\,.$$

Consequently,

$$\begin{aligned} E = \langle\uparrow\ \ &\alpha_1\ \ \langle\downarrow \langle\updownarrow \alpha_2\rangle \alpha_3 \langle\updownarrow \alpha_4\rangle \alpha_5 \langle\uparrow \langle\updownarrow \alpha_6\rangle \alpha_7 \langle\updownarrow \alpha_8\rangle\rangle \alpha_9 \langle\updownarrow \alpha_{10}\rangle\rangle \\ &\alpha_{11} \langle\updownarrow \alpha_{12}\rangle \alpha_{13}\ \ \langle\downarrow \langle\updownarrow \alpha_{14}\rangle \alpha_{15} \langle\updownarrow \alpha_{16}\rangle\rangle\ \ \alpha_{17} \langle\updownarrow \alpha_{18}\rangle\ \ \rangle\,. \end{aligned} \tag{5}$$

Indeed,

$$|E| = 39 + |X|_{\mathcal{A}} = 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\updownarrow}(X) + |X|_{\mathcal{A}}.$$

All minimal DNA expressions denoting X are constructed in this way. The result only depends on the way that lower components (for a minimal $\uparrow$-expression) or upper components (for a minimal $\downarrow$-expression) are partitioned in submolecules $X_1, \ldots, X_r$. The construction (of one minimal DNA expression) takes a time linear in the length of X.

The construction of a minimal $\uparrow$-expression for a formal DNA molecule X with $T_{\uparrow}(X) = T_{\downarrow}(X) \geq 1$ (see Theorem 9(2)) proceeds along the same lines.

7 Minimal DNA Expressions for a Molecule with Nicks

To construct minimal DNA expressions for an expressible formal DNA molecule X containing nick letters, we first decompose X into nick free pieces and nick letters. We call the result the *nick free decomposition* of X.

Consider, for example, the formal DNA molecule X depicted in Fig. 4. This molecule contains three lower nick letters and no upper nick letters. The nick free decomposition of X is $Z_{1\,\triangle} Z_{2\,\triangle} Z_{3\,\triangle} Z_4$, where

$$\begin{aligned} Z_1 &= \tbinom{\alpha_1}{-}\tbinom{\alpha_2}{c(\alpha_2)}\tbinom{-}{\alpha_3}\tbinom{\alpha_4}{c(\alpha_4)}, \\ Z_2 &= \tbinom{\alpha_5}{c(\alpha_5)}\tbinom{-}{\alpha_6}\tbinom{\alpha_7}{c(\alpha_7)}\tbinom{\alpha_8}{-}\tbinom{\alpha_9}{c(\alpha_9)}\tbinom{-}{\alpha_{10}}\tbinom{\alpha_{11}}{c(\alpha_{11})}, \\ Z_3 &= \tbinom{\alpha_{12}}{c(\alpha_{12})}\tbinom{\alpha_{13}}{-}\tbinom{\alpha_{14}}{c(\alpha_{14})}\tbinom{\alpha_{15}}{-}\tbinom{\alpha_{16}}{c(\alpha_{16})}, \\ Z_4 &= \tbinom{\alpha_{17}}{c(\alpha_{17})}. \end{aligned}$$

A DNA expression E is called *operator-minimal*, if for every equivalent DNA expression E' with the same outermost operator, $|E'| \geq |E|$. For example, consider the formal DNA molecule Z_2, for which $T_{\uparrow}(Z_2) = 1$, $T_{\downarrow}(Z_2) = 2$ and $n_{\updownarrow}(Z_2) = 4$. The $\uparrow$-expression

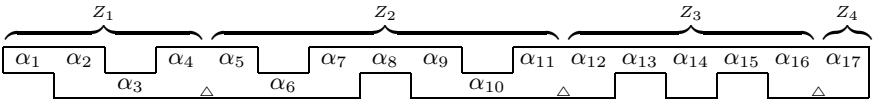

Fig. 4. Pictorial representation of a formal DNA molecule X containing lower nick letters. The nick free decomposition of X is $Z_1 {}_\triangle Z_2 {}_\triangle Z_3 {}_\triangle Z_4$

$$E_2 = \langle\uparrow \langle\downarrow \langle\updownarrow \alpha_5\rangle \alpha_6 \langle\updownarrow \alpha_7\rangle\rangle \alpha_8 \langle\downarrow \langle\updownarrow \alpha_9\rangle \alpha_{10} \langle\updownarrow \alpha_{11}\rangle\rangle\rangle\,,$$

which denotes Z_2 and has length

$$|E_2| = 21 + |Z_2|_{\mathcal{A}} = 3 + 3 \cdot T_{\downarrow}(Z_2) + 3 \cdot n_{\updownarrow}(Z_2) + |Z_2|_{\mathcal{A}},$$

is operator-minimal, because by Theorem 8(1), there can be no shorter $\uparrow$-expression denoting Z_2.

However, because $T_{\downarrow}(Z_2) > T_{\uparrow}(Z_2)$, E_2 is not minimal. By Theorem 9(4), each minimal DNA expression E_2' denoting Z_2 is a $\downarrow$-expression and has length

$$|E_2'| = 3 + 3 \cdot T_{\uparrow}(Z_2) + 3 \cdot n_{\updownarrow}(Z_2) + |Z_2|_{\mathcal{A}} = 18 + |Z_2|_{\mathcal{A}}.$$

We are in particular interested in operator-minimal $\uparrow$-expressions and $\downarrow$-expressions denoting nick free formal DNA molecules. These operator-minimal DNA expressions appear to be constructed in exactly the same way as the minimal $\uparrow$-expressions and $\downarrow$-expressions for nick free formal DNA molecules, which we have seen in the previous section. The only difference is that operator-minimal $\uparrow$-expressions and $\downarrow$-expressions can be constructed for *every* nick free formal DNA molecule, and not just for formal DNA molecules X satisfying certain conditions on $T_{\uparrow}(X)$ and $T_{\downarrow}(X)$.

We can now describe the minimal DNA expressions denoting expressible formal DNA molecules containing nick letters. We only give the formulation for molecules with lower nick letters, as the formulation for the case with upper nick letters is completely analogous. Note that by definition, there do not exist $\downarrow$-expressions that denote a formal DNA molecule containing lower nick letters.

Theorem 10. *Let X be an expressible formal DNA molecule which contains at least one lower nick letter $\triangle$, and let $Z_1 {}_\triangle Z_2 {}_\triangle \cdots {}_\triangle Z_m$ for some $m \geq 2$ be the nick free decomposition of X.*

For $h = 1, \ldots, m$, let E_h be an operator-minimal $\uparrow$-expression denoting Z_h and let the string $\widehat{E}_h$ be the sequence of the arguments of E_h. Then $E = \left\langle\uparrow \widehat{E}_1 \ldots \widehat{E}_m\right\rangle$ is a minimal DNA expression denoting X and

$$|E| = 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\updownarrow}(X) + |X|_{\mathcal{A}}.$$

Each minimal DNA expression denoting X is constructed in this way.

We return to the formal DNA molecule X from Fig. 4, for which $T_{\downarrow}(X) = 3$ and $n_{\updownarrow}(X) = 10$. We already established the nick free decomposition $Z_1 {}_\triangle Z_2 {}_\triangle Z_3 {}_\triangle Z_4$ of X and considered an operator-minimal $\uparrow$-expression E_2 denoting Z_2. It is not difficult to also construct operator-minimal $\uparrow$-expressions for Z_1, Z_3 and Z_4:

$$E_1 = \langle\uparrow\, \alpha_1\, \langle\downarrow\, \langle\updownarrow\, \alpha_2\rangle\, \alpha_3\, \langle\updownarrow\, \alpha_4\rangle\rangle\rangle\,,$$
$$E_3 = \langle\uparrow\, \langle\updownarrow\, \alpha_{12}\rangle\, \alpha_{13}\, \langle\updownarrow\, \alpha_{14}\rangle\, \alpha_{15}\, \langle\updownarrow\, \alpha_{16}\rangle\rangle\,,$$
$$E_4 = \langle\uparrow\, \langle\updownarrow\, \alpha_{17}\rangle\rangle\,.$$

The corresponding minimal DNA expression denoting the entire formal DNA molecule X is

$$E = \langle\uparrow\ \ \alpha_1\, \langle\downarrow\, \langle\updownarrow\, \alpha_2\rangle\, \alpha_3\, \langle\updownarrow\, \alpha_4\rangle\rangle\ \ \langle\downarrow\, \langle\updownarrow\, \alpha_5\rangle\, \alpha_6\, \langle\updownarrow\, \alpha_7\rangle\rangle\, \alpha_8\, \langle\downarrow\, \langle\updownarrow\, \alpha_9\rangle\, \alpha_{10}\, \langle\updownarrow\, \alpha_{11}\rangle\rangle$$
$$\langle\updownarrow\, \alpha_{12}\rangle\, \alpha_{13}\, \langle\updownarrow\, \alpha_{14}\rangle\, \alpha_{15}\, \langle\updownarrow\, \alpha_{16}\rangle\ \ \langle\updownarrow\, \alpha_{17}\rangle\ \ \rangle\,.$$

Indeed,

$$|E| = 42 + |X|_{\mathcal{A}} = 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\updownarrow}(X) + |X|_{\mathcal{A}}.$$

Also this construction requires linear time.

8 The Number of Minimal DNA Expressions

In principle, there may be many different minimal DNA expressions which denote the same formal DNA molecule. This is due to the different partitionings of lower or upper components that we can choose for the construction of an (operator-)-minimal $\uparrow$-expression or $\downarrow$-expression denoting a nick free formal DNA molecule.

Let X be a nick free formal DNA molecule. There appears to be an elegant bijection between (operator-)minimal $\uparrow$-expressions E denoting X and sequences of $T_{\downarrow}(X)$ well-nested pairs of brackets. This sequence is obtained from E by removing all symbols from E except the brackets corresponding to inner occurrences of the operators $\uparrow$ and $\downarrow$. The result for the minimal $\uparrow$-expression from (5) is $\langle\langle\rangle\rangle\,\langle\rangle$, which is indeed a sequence of $T_{\downarrow}(X) = 3$ well-nested pairs of brackets.

The number of such sequences is one of the many combinatorial interpretations of the well-known Catalan numbers (see [Stanley, 1999]). For $p \geq 0$, there exist $C_p = \frac{1}{p+1}\binom{2p}{p}$ sequences of p well-nested pairs of brackets.

Now, for an expressible formal DNA molecule X, let $n_{\min}(X)$ be the number of different minimal DNA expressions denoting X. We have:

Theorem 11. *Let X be an expressible formal DNA molecule.*

1. *If X is $\langle\updownarrow\, \alpha_1\rangle$ for an $\mathcal{N}$-word α_1, then $n_{min}(X) = 1$.*
2. *If X is nick free and $T_{\uparrow}(X) = T_{\downarrow}(X) = p$ with $p \geq 1$, then $n_{min}(X) = 2 \cdot C_p$.*
3. *If X is nick free and $T_{\uparrow}(X) > T_{\downarrow}(X) = p$ with $p \geq 0$, then $n_{min}(X) = C_p$.*
4. *If X is nick free and $T_{\downarrow}(X) > T_{\uparrow}(X) = p$ with $p \geq 0$, then ... (symmetric to Claim 3).*
5. *If X contains at least one lower nick letter, then let $Z_1 {}_{\triangle} Z_2 {}_{\triangle} \ldots {}_{\triangle} Z_m$ for some $m \geq 2$ be the nick free decomposition of X, and let for $h = 1, \ldots, m$, $p_h = T_{\downarrow}(Z_h)$. Then $n_{min}(X) = C_{p_1} \times \cdots \times C_{p_m}$.*
6. *If X contains at least one upper nick letter, then ... (symmetric to Claim 5).*

9 Characterization of Minimal DNA Expressions

When we want to decide whether or not a given DNA expression E is minimal, we can determine its semantics $X = \mathcal{S}(E)$, look up the length of a minimal DNA expression denoting X and compare this to the length $|E|$ of E. There is, however, also a direct way, based on the following characterization:

Theorem 12. *A DNA expression E is minimal, if and only if*

- *each occurrence of the operator $\updownarrow$ in E has as its argument an $\mathcal{N}$-word α (i.e., not a DNA expression),*
- *and no occurrence of the operator $\uparrow$ in E has an argument that is an $\uparrow$-expression, and no occurrence of the operator $\downarrow$ in E has an argument that is a $\downarrow$-expression,*
- *and unless $E = \langle\uparrow \alpha\rangle$ or $E = \langle\downarrow \alpha\rangle$ for an $\mathcal{N}$-word α, each occurrence of an operator $\uparrow$ or $\downarrow$ in E has at least two arguments,*
- *and for each inner occurrence of an operator $\uparrow$ or $\downarrow$ in E, the arguments are $\mathcal{N}$-words and DNA expressions, alternately,*
- *and for each inner occurrence of an operator $\uparrow$ or $\downarrow$ in E,*
  - *the first argument is either an $\mathcal{N}$-word or an $\updownarrow$-expression,*
  - *and the last argument is either an $\mathcal{N}$-word or an $\updownarrow$-expression,*
- *and if the outermost operator of E is $\uparrow$ or $\downarrow$, then*
  - *either it has two consecutive arguments which are DNA expressions,*
  - *or its first argument is an $\mathcal{N}$-word or an $\updownarrow$-expression,*
  - *or its last argument is an $\mathcal{N}$-word or an $\updownarrow$-expression.*

For an arbitrary DNA expression E, we can verify these six properties in a time linear in the length of E.

10 Conclusions and Directions for Future Research

We have introduced DNA expressions as a formal notation for DNA molecules that may contain nicks and gaps. There exist, however, (formal) DNA molecules with nicks that cannot be represented. For each expressible formal DNA molecule, we have described the minimal DNA expression(s) denoting it and we have determined the number of such minimal DNA expressions. For almost all types of expressible formal DNA molecules, the number of minimal DNA expressions can be expressed in terms of the Catalan numbers. Finally, we have characterized minimal DNA expressions by six properties which can easily be verified.

Because each expressible formal DNA molecule can be denoted by infinitely many DNA expressions, one may ask for a normal form: a well-defined set of properties such that for each expressible formal DNA molecule X, there is a unique DNA expression denoting X and satisfying those properties. And given a normal form, one may ask for an algorithm that, for each DNA expression, determines the equivalent DNA expression in normal form. We already have a normal form and a corresponding algorithm for nick free formal DNA molecules.

We also have ideas for another normal form and a corresponding algorithm, which applies to *all* expressible formal DNA molecules. A nice feature of this new normal form is that each DNA expression satisfying it is minimal.

One may also consider a new set of operators to construct DNA expressions. The result may be that each formal DNA molecule becomes expressible, or that two formal DNA molecules with complementary sticky ends can anneal. It would be desirable/interesting to find extensions such that the new DNA expressions could be used to denote DNA molecules with a variety of other 'imperfections', such as, e.g., hairpin loops and circular strands.

References

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

D. Boneh, C. Dunworth, R.J. Lipton: Breaking DES using a molecular computer, *DNA based computers – Proceedings of a DIMACS workshop* (R.J. Lipton, E.B. Baum, eds.), American Mathematical Society, Providence, RI (1996), 37-66.

J. Chen, J. Reif (eds.): *DNA computing – 9th International workshop on DNA based computers*, LNCS **2943**, Springer-Verlag, Berlin (2004).

M. Daley, L. Kari, I. McQuillan: Families of languages defined by ciliate bio-operations, *Theoretical Computer Science* **320**(1) (2004), 51-69.

A. Ehrenfeucht, T. Harju, I. Petre, D.M. Prescott, G. Rozenberg: *Computation in living cells – Gene assembly in ciliates*, Springer-Verlag, Berlin (2004).

M. Hagiya, A. Ohuchi (eds.): *DNA computing – 8th International workshop on DNA-based computers*, LNCS **2568**, Springer-Verlag, Berlin (2003).

T. Head: Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors, *Bulletin of Mathematical Biology* **49**(6) (1987), 737-759.

T. Head, Gh. Păun, D. Pixton: Language theory and molecular genetics: generative mechanisms suggested by DNA recombination, *Handbook of formal languages* (G. Rozenberg, A. Salomaa, eds.), Vol. 2, Springer-Verlag, Berlin (1997), 295-360.

L. Kari, Gh. Păun, A. Salomaa: The power of restricted splicing with rules from a regular language, *Journal of Universal Computer Science* **2**(4) (1996), 224-240.

L.F. Landweber, L. Kari: The evolution of cellular computing: nature's solution to a computational problem, *Proceedings of the fourth international meeting on DNA based computers*, *BioSystems* **52** (1999), 3-13.

Z. Li: Algebraic properties of DNA operations, *Proceedings of the fourth international meeting on DNA based computers*, *BioSystems* **52** (1999), 55-61.

Gh. Păun, G. Rozenberg, A. Salomaa: *DNA computing – New computing paradigms*, Springer-Verlag, Berlin (1998).

R. P. Stanley: *Enumerative combinatorics*, Vol. 2, Cambridge University Press, Cambridge (1999).

R. van Vliet: Combinatorial aspects of minimal DNA expressions (ext.), Technical Report 2004-03, Leiden Institute of Advanced Computer Science, Leiden University (2004), see `www.liacs.nl/home/rvvliet/mindnaexpr.html`.