



Universiteit Leiden

Technical Report 2004-03

March 2004

Universiteit Leiden

Leiden Institute of Advanced Computer Science

Combinatorial Aspects of Minimal DNA Expressions (ext.)

Rudy van Vliet

`rvvliet@liacs.nl`

Leiden Institute of Advanced Computer Science
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands



Universiteit Leiden

Technical Report 2004-03

March 2004

Universiteit Leiden

Leiden Institute of Advanced Computer Science

Combinatorial Aspects of Minimal DNA Expressions (ext.)

Rudy van Vliet

`rvvliet@liacs.nl`

Leiden Institute of Advanced Computer Science
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Preface

On March 8, 2004, Rudy van Vliet, Hendrik Jan Hoogeboom and Grzegorz Rozenberg submitted a paper entitled *Combinatorial Aspects of Minimal DNA expressions* as full paper to DNA 10, the Tenth International Meeting on DNA Based Computers, which will be held at the University of Milano-Bicocca from June 7 till June 10, 2004. Due to space limitations, we could not include the formal proofs of the results in this paper. These proofs can be found in the present technical report.

The following table specifies the definitions and results in the technical report which correspond to the definitions and results in the paper.

In paper	In technical report
Definition 1	Definition 2.1
Definition 2	Definition 2.3
Lemma 3	Lemma 2.4
DNA expression	Definition 2.8, Definition 2.9
Theorem 4	Theorem 3.4, Theorem 3.5
Lemma 5	Lemma 4.1
Definition 6	Definition 4.3, Definition 4.8
Lemma 7	Lemma 4.11(1),(2)
Theorem 8	Theorem 4.18, Corollary 4.19
Theorem 9	Theorem 4.23, Theorem 4.53, Theorem 4.77
construction of operator- minimal DNA expression	Theorem 4.65, Theorem 4.78
Theorem 10	Theorem 4.67, Theorem 4.79
Theorem 11	Lemma 4.83, Corollary 4.95, Corollary 4.96
Theorem 12	Definition 4.97, Theorem 4.100

In the paper, we use the notation $|X|_{\mathcal{A}}$ to count the \mathcal{A} -letters occurring in a formal DNA molecule X . This is equal to the quantity $|\nu(X)|$ we use in this technical report. The function ν is introduced in § 2.5.

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.

Contents

1	Introduction	1
2	Terminology and notation	3
2.1	Strings and \mathcal{N} -words	3
2.2	Formal DNA molecules	5
2.3	A simplified notation for formal DNA molecules	7
2.4	Properties of and relations between formal DNA molecules	10
2.5	Functions on formal DNA molecules	11
2.6	Operators and DNA expressions	12
2.7	Brackets, arguments and DNA subexpressions	20
2.8	The functions \mathbf{L} and \mathbf{R} for DNA expressions	22
2.9	Recognition of DNA expressions	23
2.10	Concatenation of DNA expressions	24
2.11	DNA expressions and trees	24
2.12	Equivalent DNA expressions	26
3	Basic results on DNA expressions	29
3.1	Expressible formal DNA molecules	29
3.2	Some equivalences	31
4	The length of a DNA expression	39
4.1	Lower bounds for the length of a DNA expression	41
4.2	Minimal DNA expressions	58
4.2.1	Minimal DNA expressions for a nick free formal DNA molecule	59
4.2.2	Minimal DNA expressions for a formal DNA molecule with nick letters	89
4.2.3	All minimal DNA expressions for a formal DNA molecule	98
4.2.4	The number of minimal DNA expressions	111
4.2.5	Recurrence relation for the number of operator-minimal \uparrow -expressions and \downarrow -expressions	132
4.2.6	Recognition of minimal DNA expressions	133
4.2.7	Trees of minimal DNA expressions	142
5	Conclusions and directions for future research	146
A	Recognition of DNA expressions (an implementation)	147
	List of symbols	152
	Index	153

Chapter 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., 1998]). 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 direct Hamiltonian path problem using DNA, enzymes and standard biomolecular operations. His approach exploited the massive parallelism of the huge number of molecules in a test tube of DNA. 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., 2003] 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 $\begin{matrix} \text{ACATG} \\ \text{TGTAC} \end{matrix}$) to describe a double-stranded DNA molecule.

In this report, 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 . 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).

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.

In Chapter 2, we define the formal DNA molecules and DNA expressions. We also introduce most of the terminology related to them and the notation that we use. In Chapter 3, we present some basic results on DNA expressions. We consider the length of our DNA expressions in Chapter 4. This chapter contains the most important results of the present report. Finally, in Chapter 5, we draw the conclusions from our work and indicate directions for future research.

Chapter 2

Terminology and notation

In this chapter, we introduce the terminology and the notation that we use in this report. We start with some basic notions related to strings and we define \mathcal{N} -words. Next, we describe a formalization of DNA molecules called formal DNA molecules. We then define DNA expressions – character strings which denote (formal) DNA molecules. This is the central concept of this report. Among other things, we consider a natural tree representation of DNA expressions and introduce the notion of equivalence, for DNA expressions that denote (almost) the same formal DNA molecule.

2.1 Strings and \mathcal{N} -words

An *alphabet* is a finite set, the elements of which are called *symbols* or *letters*. A finite sequence of symbols from an alphabet Σ will be called a *string* over Σ . For a string $X = x_1x_2\dots x_r$ over an alphabet Σ , with $x_i \in \Sigma$ for $i = 1, 2, \dots, r$, the *length* of X is r and it is denoted by $|X|$. The length of the empty string λ equals 0.

For a non-empty string $X = x_1x_2\dots x_r$, we define $L(X) = x_1$ and $R(X) = x_r$. The concatenation of two strings X_1 and X_2 over an alphabet Σ is usually denoted by X_1X_2 ; sometimes, however, we will write $X_1 \cdot X_2$.

The set of all strings over an alphabet Σ is denoted by Σ^* , and $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$ (the set of non-empty strings). A *language* over Σ is a subset \mathcal{L} of Σ^* .

Let $\mathcal{N} = \{A, C, G, T\}$ be the alphabet of nucleotides. The elements of \mathcal{N} are called *\mathcal{N} -letters*. We will reserve the symbol a (possibly with a subscript) to denote \mathcal{N} -letters. A *non-empty* string over \mathcal{N} is called an *\mathcal{N} -word*. Clearly, the set \mathcal{N}^+ of \mathcal{N} -words is closed under concatenation. We will reserve the symbol α (possibly with a subscript) to denote \mathcal{N} -words.

Substrings

A *substring* of a string X is a (possibly empty) string X^s such that there are (possibly empty) strings X_1 and X_2 with $X = X_1X^sX_2$. If $X^s \neq X$, then X^s is a *proper substring* of X . We call the pair (X_1, X_2) an *occurrence* of X^s in X . If there exists a (possibly empty) string X_2 such that $X = X^sX_2$, then X^s is a *prefix* of X ; if there exists a (possibly empty) string X_1 such that $X = X_1X^s$, then X^s is a *suffix* of X .

If (X_1, X_2) and (Y_1, Y_2) are different occurrences of X^s in X , then (X_1, X_2) *precedes* (Y_1, Y_2) if $|X_1| < |Y_1|$. Hence, all occurrences in X of a given string X^s are linearly ordered, and we can talk about the first, second, \dots occurrence of X^s in X . Although, formally, an occurrence of a substring X^s in a string X is the pair (X_1, X_2) surrounding

X^s in X , the term will also be used to refer to the substring itself, at the position in X determined by (X_1, X_2) .

Note that for a string $X = x_1x_2\dots x_r$ of length r , the empty string λ has $r + 1$ occurrences: $(\lambda, X), (x_1, x_2\dots x_r), \dots, (x_1\dots x_{r-1}, x_r), (X, \lambda)$.

If $X^s = a$ for a letter a from the alphabet Σ , then the number of occurrences of X^s in X is denoted by $\#_a(X)$. Obviously, when $X = x_1x_2\dots x_r$ with $x_1, x_2, \dots, x_r \in \Sigma$, $\#_a(X)$ is the number of x_i 's that are equal to a . Sometimes, we are not so much interested in the number of occurrences of *one* letter in a string X , but rather in the total number of occurrences of two different letters a and b in X . This total number is denoted by $\#_{a,b}(X)$.

If a string X is the catenation of k times the same substring X^s , hence $X = \underbrace{X^s \dots X^s}_{k \text{ times}}$, then we may write X in the form $(X^s)^k$.

Let (Y_1, Y_2) and (Z_1, Z_2) be occurrences in a string X of substrings Y^s and Z^s , respectively. We say that (Y_1, Y_2) and (Z_1, Z_2) are *disjoint*, if either $|Y_1| + |Y^s| \leq |Z_1|$ or $|Z_1| + |Z^s| \leq |Y_1|$. Intuitively, one of the substrings occurs (in its entirety) before the other one.

If the two occurrences are not disjoint, hence if $|Z_1| < |Y_1| + |Y^s|$ and $|Y_1| < |Z_1| + |Z^s|$, then they are said to *intersect*. Note that, according to this formalization of intersection, an occurrence of the empty string λ may intersect with an occurrence of a non-empty string. For example, in the string $X = \text{ACATGAT}$ over the alphabet \mathcal{N} , the third occurrence of λ (the occurrence $(\text{AC}, \text{ATGAT})$) intersects with the (only) occurrence of CAT . In the remainder of this thesis, however, we will not come across intersections of λ with other strings. Occurrences of two non-empty substrings intersect, if and only if the substrings have at least one (occurrence of a) letter in common.

We say that (Y_1, Y_2) *overlaps* with (Z_1, Z_2) , if either $|Y_1| < |Z_1| < |Y_1| + |Y^s| < |Z_1| + |Z^s|$ or $|Z_1| < |Y_1| < |Z_1| + |Z^s| < |Y_1| + |Y^s|$. Hence, one of the substrings starts *before* and ends *inside* the other one.

Finally, the occurrence (Y_1, Y_2) of Y^s *contains* the occurrence (Z_1, Z_2) of Z^s , if $|Y_1| \leq |Z_1|$ and $|Z_1| + |Z^s| \leq |Y_1| + |Y^s|$.

If it is clear from the context which occurrences of Y^s and Z^s in X are considered, e.g., if these strings occur in X exactly once, then we may also say that the substrings Y^s and Z^s *themselves* are disjoint, intersect or overlap, or that one contains the other.

Note the difference between intersection and overlap. If (occurrences of) two substrings intersect, then either they overlap, or one contains the other, and these two possibilities are mutually exclusive. For example, in the string $X = \text{ACATGAT}$ over \mathcal{N} , the (only occurrence of the) substring $Y^s = \text{ATGA}$ intersects with both occurrences of the substring $Z^s = \text{AT}$. It contains the first occurrence of Z^s and it overlaps with the second occurrence of Z^s .

In Figure 2.1, we have schematically depicted the notions of disjointness, intersection, overlap and containment.

Functions on strings

Let Σ be an alphabet. A function h on Σ^* is called a *homomorphism* if $h(X_1X_2) = h(X_1)h(X_2)$ for all $X_1, X_2 \in \Sigma^*$. Hence, to specify h it suffices to give its values for the letters from Σ . Note that $h(\lambda) = \lambda$.

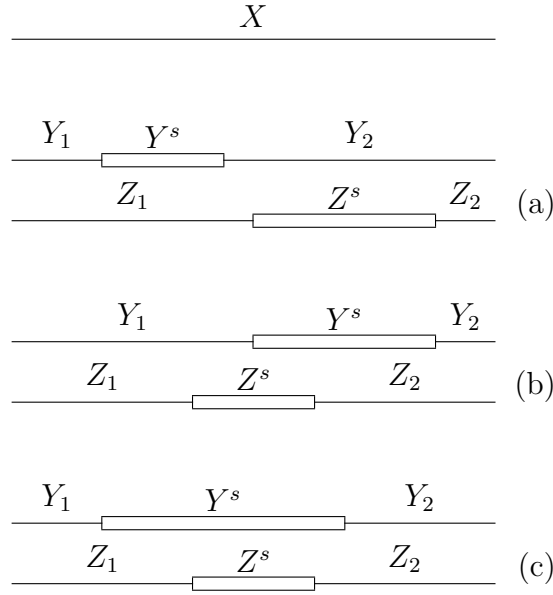


Figure 2.1: Examples of disjoint and intersecting occurrences (Y_1, Y_2) of Y^s and (Z_1, Z_2) of Z^s in a string X : (a) the occurrences are disjoint: $|Y_1| + |Y^s| \leq |Z_1|$; (b) the occurrences overlap: $|Z_1| < |Y_1| < |Z_1| + |Z^s| < |Y_1| + |Y^s|$; (c) the occurrence of Y^s contains the occurrence of Z^s : $|Y_1| \leq |Z_1|$ and $|Z_1| + |Z^s| \leq |Y_1| + |Y^s|$.

If, additionally, h maps the elements of Σ^* into Σ^* , then h is called an *endomorphism*.

The symbol c will denote the complement function. It is an endomorphism on \mathcal{N}^* , specified by

$$c(A) = T, \quad c(C) = G, \quad c(G) = C, \quad c(T) = A.$$

Thus, for an \mathcal{N} -word α , $c(\alpha)$ results by replacing each letter of α by its Watson-Crick complement. For example, $c(\text{ACATG}) = \text{TGTAC}$.

2.2 Formal DNA molecules

Before we define the expressions in our DNA language, we want to be more precise about their meaning – the semantics of the DNA expressions. For this purpose, we formalize the double-word notation for DNA molecules that may contain nicks and gaps.

Every symbol in the upper word of a double word corresponds to a symbol in the lower word. If there are no gaps, then two such corresponding symbols denote a base pair – two complementary nucleotides that are connected through a hydrogen bond. In the formal semantics of our DNA expressions, a pair of corresponding elements in the upper and the lower word is denoted by a composite symbol $x = \begin{pmatrix} x^+ \\ x^- \end{pmatrix}$. Here x^+ stands for the nucleotide in the upper word and x^- stands for the nucleotide in the lower word. If we happen to have a gap in either of the strands, the missing nucleotide is denoted by $-$. Hence, $x^+, x^- \in \mathcal{N} \cup \{-\}$. For convenience, we will speak of a base pair also if one of two complementary nucleotides is missing. If both nucleotides are present, we may call the base pair *complete*.

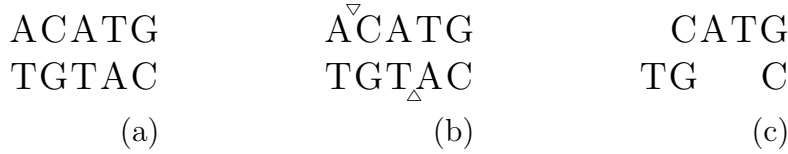


Figure 2.2: Examples of possible DNA molecules: (a) a perfect double-stranded DNA molecule; (b) a DNA molecule with nicks; (c) a DNA molecule with gaps.

Of course, the value of x^+ restricts the value of x^- , and vice versa. Because of the Watson-Crick complementarity and the fact that a missing nucleotide cannot face another missing nucleotide, only 12 out of the 25 possible composite symbols $\binom{x^+}{x^-}$ are really allowed: $\binom{A}{T}$, $\binom{C}{G}$, $\binom{G}{C}$, $\binom{T}{A}$, $\binom{A}{-}$, $\binom{C}{-}$, $\binom{G}{-}$, $\binom{T}{-}$, $\binom{-}{A}$, $\binom{-}{C}$, $\binom{-}{G}$, $\binom{-}{T}$. The set of these 12 composite symbols is denoted by \mathcal{A} .

For the future use, we partition \mathcal{A} into three subsets: $\mathcal{A}_\pm = \left\{ \binom{A}{T}, \binom{C}{G}, \binom{G}{C}, \binom{T}{A} \right\}$, $\mathcal{A}_+ = \left\{ \binom{A}{-}, \binom{C}{-}, \binom{G}{-}, \binom{T}{-} \right\}$ and $\mathcal{A}_- = \left\{ \binom{-}{A}, \binom{-}{C}, \binom{-}{G}, \binom{-}{T} \right\}$. The elements of \mathcal{A} are called *\mathcal{A} -letters*, the elements of \mathcal{A}_\pm are called *double \mathcal{A} -letters*, the elements of \mathcal{A}_+ are called *upper \mathcal{A} -letters*, and the elements of \mathcal{A}_- are called *lower \mathcal{A} -letters*. Consequently, a non-empty string over \mathcal{A} is called an *\mathcal{A} -word*, a non-empty string over \mathcal{A}_\pm is called a *double \mathcal{A} -word*, a non-empty string over \mathcal{A}_+ is called an *upper \mathcal{A} -word*, and a non-empty string over \mathcal{A}_- is called a *lower \mathcal{A} -word*.

We also need symbols to denote nicks in a double word. There are three possibilities for the connection structure of two adjacent base pairs in a double word: there can be a nick in the upper word, there can be a nick in the lower word, or there can be no nick at all between the base pairs. Note that there cannot be both a nick in the upper word and a nick in the lower word between two adjacent base pairs. In such a situation, there would be no connection whatsoever between the base pairs, so they would be parts of different DNA molecules.

The case that there is no nick at all is the default; it is not denoted explicitly. A nick in the upper word is denoted by ∇ and a nick in the lower word by \triangle . We call ∇ and \triangle the *nick letters* – ∇ is the *upper* nick letter, and \triangle the *lower* nick letter.

Now, a complete description of a linear DNA molecule possibly containing nicks and gaps can be given by a non-empty string X over $\mathcal{A}_{\nabla\triangle} = \mathcal{A} \cup \{\nabla, \triangle\}$.

For example, the DNA molecules depicted in Figure 2.2 are denoted by

$$X_1 = \binom{A}{T} \binom{C}{G} \binom{A}{T} \binom{T}{A} \binom{G}{C}, \quad (2.1)$$

$$X_2 = \binom{A}{T}^{\nabla} \binom{C}{G} \binom{A}{T}^{\triangle} \binom{T}{A} \binom{G}{C}, \text{ and} \quad (2.2)$$

$$X_3 = \binom{-}{T} \binom{C}{G} \binom{A}{-} \binom{T}{-} \binom{G}{C}, \quad (2.3)$$

respectively. X_1 and X_3 have length 5, and X_2 has length 7.

Not every string over $\mathcal{A}_{\nabla\triangle}$ represents a DNA molecule. The requirements that strings over $\mathcal{A}_{\nabla\triangle}$ need to satisfy follow from three observations on DNA molecules:

1. to enable at least one phosphodiester bond between adjacent base pairs, a gap in one strand cannot be adjacent to a gap in the other strand (see Figure 2.3(a));

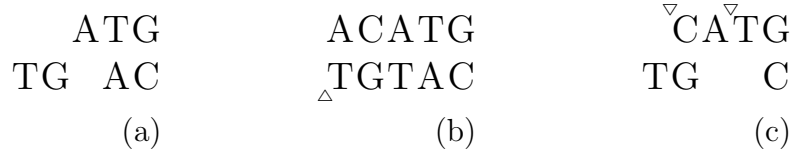


Figure 2.3: Examples of impossible DNA molecules: (a) adjacent gaps in different strands of the molecule; (b) a nick at an extremity of the molecule; (c) nicks between base pairs that are not (both) complete.

2. a nick may occur only *between* two base pairs; in particular, it cannot occur at an extremity of a DNA molecule (see Figure 2.3(b));
3. since a nick is a missing phosphodiester bond between two adjacent nucleotides in the same strand, we really need to have nucleotides on both sides of the nick; moreover, the complementary nucleotides in the other strand must be present and they must be connected by a phosphodiester bond; hence, a nick may occur only between two complete base pairs (see Figure 2.3(c)).

Now, we are ready to define our notation for DNA molecules that may contain nicks and gaps:

Definition 2.1 *A formal DNA molecule is a non-empty string $X = x_1x_2\dots x_r$ with $x_i \in \mathcal{A}_{\nabla\triangle}$ for $i = 1, \dots, r$, satisfying*

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

The language of all formal DNA molecules is denoted by \mathcal{F} . Since $X \in \mathcal{F}$ is called a molecule (albeit ‘formal’), we will refer to the sequence of (possibly missing) nucleotides x_i^+ and upper nick letters in X as the upper strand of X . The lower strand of X is defined analogously.

Note, however, that it does not make sense to talk about the upper and lower strands of a (physical) DNA molecule, because a rotation of the molecule over an angle π would change the upper strand into a lower strand and vice versa.

If a formal DNA molecule does not contain upper nick letters, then we say that its *upper strand is nick free*. Similarly, if a formal DNA molecule does not contain lower nick letters, then its *lower strand is nick free*. If a formal DNA molecule does not contain nick letters at all, then the molecule is called *nick free*.

2.3 A simplified notation for formal DNA molecules

Let $X = x_1\dots x_r$ be a formal DNA molecule, with $x_i \in \mathcal{A}_{\nabla\triangle}$ for $i = 1, \dots, r$. A *formal DNA submolecule* of X is a substring X^s of X such that X^s is a formal DNA molecule. It is easy to see that

Lemma 2.2 *A substring X^s of a formal DNA molecule X is a formal DNA molecule if and only if $|X^s| \geq 1$ and $L(X^s), R(X^s) \in \mathcal{A}$.*

Hence, X^s should not be empty and neither its first symbol nor its last symbol should be a nick letter.

If a formal DNA submolecule X^s of X is an upper \mathcal{A} -word, a lower \mathcal{A} -word or a double \mathcal{A} -word, and $|X^s| \geq 2$, it is possible to simplify the notation for X^s and X . Let $\alpha = a_1 \dots a_l$ be an \mathcal{N} -word with $a_i \in \mathcal{N}$ ($i = 1, \dots, l$), and let X^s be a formal DNA submolecule of X with $X^s = x_{i_0} \dots x_{i_0+l-1}$ for some i_0 with $1 \leq i_0 \leq r - l + 1$ (so $|X^s| = l$).

If $X^s = \binom{a_1}{-} \dots \binom{a_l}{-}$, we may write

$$X^s = \binom{\alpha}{-} \text{ and } X = x_1 \dots x_{i_0-1} \binom{\alpha}{-} x_{i_0+l} \dots x_r.$$

Similarly, if $X^s = \binom{-}{a_1} \binom{-}{a_2} \dots \binom{-}{a_l}$, we may write

$$X^s = \binom{-}{\alpha} \text{ and } X = x_1 \dots x_{i_0-1} \binom{-}{\alpha} x_{i_0+l} \dots x_r.$$

Finally, if $X^s = \binom{a_1}{c(a_1)} \binom{a_2}{c(a_2)} \dots \binom{a_l}{c(a_l)}$, we may write

$$X^s = \binom{\alpha}{c(\alpha)} \text{ and } X = x_1 \dots x_{i_0-1} \binom{\alpha}{c(\alpha)} x_{i_0+l} \dots x_r.$$

For an \mathcal{N} -word α , we call the set $\left\{ \binom{\alpha}{-}, \binom{-}{\alpha}, \binom{\alpha}{c(\alpha)} \right\}$ the set of \mathcal{A} -words determined by α , and denote it by $\mathcal{W}_{\mathcal{A}}(\alpha)$.

By simplifying the notation we seem to extend the alphabet of \mathcal{F} with infinitely many symbols $\binom{\alpha}{-}$, $\binom{-}{\alpha}$ and $\binom{\alpha}{c(\alpha)}$. We want to remark, however, that we actually only simplify the *presentation* of the formal DNA molecules. The formal DNA molecules themselves do not change; they are still strings over the finite alphabet $\mathcal{A}_{\nabla\Delta}$. In particular, the length of a formal DNA molecule $X = x_1 \dots x_r$ with $x_i \in \mathcal{A}_{\nabla\Delta}$ for $i = 1, \dots, r$ remains r , even if X is written in a simplified notation.

Definition 2.3 *Let X be a formal DNA molecule. Then the decomposition of X is the sequence x'_1, \dots, x'_k of $k \geq 1$ non-empty strings over $\mathcal{A}_{\nabla\Delta}$ such that*

- $X = x'_1 \dots x'_k$,
- for $i = 1, \dots, k$, 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 $i = 1, \dots, k - 1$, if x'_i is an upper \mathcal{A} -word, then x'_{i+1} is not an upper \mathcal{A} -word, and similarly 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 $x'_1 \dots x'_k$ instead of x_1, \dots, x_k .

For example, the decompositions of the formal DNA molecules from (2.1), (2.2) and (2.3) (denoting the molecules shown in Figure 2.2) are

$$\begin{aligned} X_1 &= \binom{\text{ACATG}}{\text{TGTAC}}, \\ X_2 &= \binom{\text{A}}{\text{T}} \nabla \binom{\text{CA}}{\text{GT}} \Delta \binom{\text{TG}}{\text{AC}} \text{ and} \end{aligned} \tag{2.4}$$

$$X_3 = \begin{pmatrix} - \\ \text{T} \end{pmatrix} \begin{pmatrix} \text{C} \\ \text{G} \end{pmatrix} \begin{pmatrix} \text{AT} \\ - \end{pmatrix} \begin{pmatrix} \text{G} \\ \text{C} \end{pmatrix}, \quad (2.5)$$

respectively.

If $x'_1 \dots x'_k$ 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 $i = 1, \dots, k$, if x'_i is an upper \mathcal{A} -word (lower \mathcal{A} -word or double \mathcal{A} -word), then x'_i 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 .

We now have the following result:

Lemma 2.4 *Let X be a formal DNA molecule and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X . Then*

- *for $i = 1, \dots, k - 1$, if x'_i is an upper component, a lower component or a nick letter then x'_{i+1} is a double component;*
- *for $i = 1, \dots, k - 1$, if x'_i is a double component, then x'_{i+1} is an upper component, a lower component or a nick letter.*

Hence, the components of a formal DNA molecule are double components and other types of components, alternately.

Proof: If for some i with $1 \leq i \leq k - 1$, x'_i is a double component, then by the definition of the decomposition, the next component x'_{i+1} is an upper component, a lower component or a nick letter. Because nick letters can only occur between two double components and because an upper component cannot occur next to a lower component (see Definition 2.1), the reverse is also true: if for some i with $1 \leq i \leq k - 1$, x'_i is an upper component, a lower component or a nick letter, then the next component x'_{i+1} is a double component. \square

Two special cases of this result will also appear to be useful:

Corollary 2.5 *Let X be a nick free formal DNA molecule and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X . Then*

- *for $i = 1, \dots, k$, x'_i is either an upper component, or a lower component, or a double component;*
- *for $i = 1, \dots, k - 1$, if x'_i is an upper component or a lower component, then x'_{i+1} is a double component;*
- *for $i = 1, \dots, k - 1$, if x'_i is a double component then x'_{i+1} is an upper component or a lower component.*

When we observe that by definition the first and the last component of a formal DNA molecule cannot be nick letters, we also find

Corollary 2.6 *Let X be a formal DNA molecule and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .*

If X does not contain any single-stranded component, then

- *for $i = 1, \dots, k$, x'_i is either a double component, or an upper nick letter, or a lower nick letter;*

- k is odd and

$$X = \left(\begin{array}{c} \alpha_1 \\ c(\alpha_1) \end{array} \right) y_1 \left(\begin{array}{c} \alpha_2 \\ c(\alpha_2) \end{array} \right) y_2 \cdots y_{\frac{k-1}{2}} \left(\begin{array}{c} \alpha_{\frac{k+1}{2}} \\ c(\alpha_{\frac{k+1}{2}}) \end{array} \right)$$

for \mathcal{N} -words $\alpha_1, \dots, \alpha_{\frac{k+1}{2}}$ and nick letters $y_1, \dots, y_{\frac{k-1}{2}}$.

2.4 Properties of and relations between formal DNA molecules

Let $X = x_1 \dots x_r$ be a formal DNA molecule, with $x_i \in \mathcal{A}_{\nabla\Delta}$ for $i = 1, \dots, r$. Then the upper strand of X is said to *cover* the lower strand *to the right* if $R(X) = x_r \notin \mathcal{A}_-$, hence, if $x_r^+ \neq -$; note that, since x_r is not allowed to be a nick letter (condition 2 of Definition 2.1), x_r^+ is well defined. Intuitively, the upper strand extends at least as far to the right as the lower strand then.

If also $R(X) = x_r \in \mathcal{A}_+$, hence $x_r^- = -$ (the upper strand extends even *beyond* the lower strand to the right), then the upper strand *strictly* covers the lower strand to the right. In an analogous way we can define ‘(strict) covering *to the left*’.

Of course, the definition of ‘(strict) covering’ can also be formulated for the lower strand. For example, in the formal DNA molecule from (2.3) the lower strand strictly covers the upper strand to the left. Here, the strands extend equally far to the right, and so we say that the upper strand covers the lower strand to the right and vice versa.

We say that a formal DNA molecule X_1 *prefits* a formal DNA molecule X_2 *by upper strands*, denoted by $X_1 \overline{\sqsubset} X_2$, if the upper strand of X_1 covers the lower strand to the right and the upper strand of X_2 covers the lower strand to the left, hence, if $R(X_1) \notin \mathcal{A}_-$ and $L(X_2) \notin \mathcal{A}_-$; we also say that X_1 is an *upper prefit* for X_2 then. Intuitively, when we write X_1 and X_2 after each other in such a case, the respective upper strands ‘make contact’.

Analogously, we define X_1 to *prefit* X_2 *by lower strands* (to be a *lower prefit* for X_2) if $R(X_1) \notin \mathcal{A}_+$ and $L(X_2) \notin \mathcal{A}_+$, and write then $X_1 \underline{\sqsubset} X_2$. If either $X_1 \overline{\sqsubset} X_2$ or $X_1 \underline{\sqsubset} X_2$, we say that X_1 *prefits* X_2 or that X_1 is a *prefit* for X_2 , and write then $X_1 \sqsubset X_2$.

If X_1 *prefits* X_2 (by upper/lower strands), then, from the perspective of X_2 , we say that X_2 *postfits* X_1 (by upper/lower strands), or that X_2 is an (upper/lower) *postfit* for X_1 .

If the order of the formal DNA molecules is clear, then we may also say that X_1 and X_2 *fit together* (by upper/lower strands).

In fact, we used the notion of prefitting (or postfitting) already in the definition of a (single) formal DNA molecule X . When we demanded that an element of \mathcal{A}_+ cannot be followed or preceded by an element of \mathcal{A}_- (condition 1 in Definition 2.1), we actually demanded that the formal DNA molecule $\begin{pmatrix} x_i^+ \\ x_i^- \end{pmatrix}$ (of length 1) should prefit the formal DNA molecule $\begin{pmatrix} x_{i+1}^+ \\ x_{i+1}^- \end{pmatrix}$ (for each i such that neither x_i , nor x_{i+1} is a nick letter).

Unlike the set of all \mathcal{N} -words \mathcal{N}^+ , the set of formal DNA molecules \mathcal{F} is not closed under concatenation. Let, for instance, X_1 and X_2 be formal DNA molecules such that the upper strand of X_1 strictly covers the lower strand to the right and



Figure 2.4: Schematic representation of two formal DNA molecules X_1 and X_2 such that the concatenation X_1X_2 is not a formal DNA molecule.

that the lower strand of X_2 strictly covers the upper strand to the left. Then the concatenation X_1X_2 is not a formal DNA molecule, because condition 1 of Definition 2.1 is violated for $i = |X_1|$ – this is illustrated in Figure 2.4. Thus in particular, even if $X_1 = \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \begin{pmatrix} \text{C} \\ \text{G} \end{pmatrix} \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \begin{pmatrix} \text{T} \\ - \end{pmatrix} \begin{pmatrix} \text{G} \\ - \end{pmatrix}$ and $X_2 = \begin{pmatrix} - \\ \text{A} \end{pmatrix} \begin{pmatrix} - \\ \text{C} \end{pmatrix} \begin{pmatrix} \text{C} \\ \text{G} \end{pmatrix} \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \begin{pmatrix} \text{T} \\ \text{A} \end{pmatrix}$ (so that the respective sticky ends of the DNA molecules form a perfect match), then X_1X_2 is not a formal DNA molecule.

As a matter of fact, the following property holds:

The concatenation of two formal DNA molecules X_1 and X_2 is again a formal DNA molecule if and only if $X_1 \sqsubset X_2$.

2.5 Functions on formal DNA molecules

We define four endomorphisms on the set $\mathcal{A}_{\nabla\Delta}^*$: ν^+ , ν^- , ν and κ . Let $x \in \mathcal{A}_{\nabla\Delta}$. Then

$$\nu^+(x) = \begin{cases} x & \text{if } x \in \mathcal{A} \cup \{\Delta\} \\ \lambda & \text{if } x = \nabla \end{cases} \quad (2.6)$$

$$\nu^-(x) = \begin{cases} x & \text{if } x \in \mathcal{A} \cup \{\nabla\} \\ \lambda & \text{if } x = \Delta \end{cases} \quad (2.7)$$

$$\nu(x) = \begin{cases} x & \text{if } x \in \mathcal{A} \\ \lambda & \text{if } x \in \{\nabla, \Delta\} \end{cases} \quad (2.8)$$

$$\kappa(x) = \begin{cases} x & \text{if } x \in \mathcal{A}_{\pm} \cup \{\nabla, \Delta\} \\ \begin{pmatrix} a \\ c(a) \end{pmatrix} & \text{if } x = \begin{pmatrix} a \\ - \end{pmatrix} \text{ for } a \in \mathcal{N} \\ \begin{pmatrix} c(a) \\ a \end{pmatrix} & \text{if } x = \begin{pmatrix} - \\ a \end{pmatrix} \text{ for } a \in \mathcal{N} \end{cases} \quad (2.9)$$

Thus, ν^+ removes all upper nick letters from its argument, ν^- removes all lower nick letters from its argument, ν removes both the upper nick letters and the lower nick letters from its argument, and κ replaces every symbol from \mathcal{A}_+ and \mathcal{A}_- in its argument by the ‘corresponding’ symbol from \mathcal{A}_{\pm} .

From the point of view of the molecules represented, ν^+ replaces all nicks in the upper strand of its argument by phosphodiester bonds, and ν^- does the same for nicks in the lower strand of its argument. The function ν replaces all nicks in both the upper and the lower strand by phosphodiester bonds. Finally, κ provides a complementary nucleotide for every nucleotide in its argument which is not complemented yet. The nicks are not affected by κ .

It is easy to see (by inspecting the action of the functions on the symbols from $\mathcal{A}_{\nabla\Delta}$), that the composition of functions from the set $\{\nu^+, \nu^-, \nu, \kappa\}$ is *commutative*, i.e.,

$$h_2(h_1(X)) = h_1(h_2(X)) \text{ for all } h_1, h_2 \in \{\nu^+, \nu^-, \nu, \kappa\} \text{ and } X \in \mathcal{A}_{\nabla\Delta}^*. \quad (2.10)$$

For example, $\kappa(\nu^+(X)) = \nu^+(\kappa(X))$ for each $X \in \mathcal{A}_{\nabla\Delta}^*$.

Further, applying the same function more than one time, does not change the result:

$$h(h(X)) = h(X) \text{ for each } h \in \{\nu^+, \nu^-, \nu, \kappa\} \text{ and } X \in \mathcal{A}_{\nabla\Delta}^*. \quad (2.11)$$

For example, $\nu(\nu(X)) = \nu(X)$ for each $X \in \mathcal{A}_{\nabla\Delta}^*$.

Finally, one can verify that

$$\nu^-(\nu^+(X)) = \nu(X) \text{ for each } X \in \mathcal{A}_{\nabla\Delta}^*. \quad (2.12)$$

Hence, ν is equal to the composition of ν^+ and ν^- (and, by commutativity, ν is equal to the composition of ν^- and ν^+).

Because \mathcal{F} , the set of formal DNA molecules, is a subset of $\mathcal{A}_{\nabla\Delta}^*$, ν^+ , ν^- , ν and κ can be applied to \mathcal{F} . It is easy to verify that for each $X \in \mathcal{F}$ and $h \in \{\nu^+, \nu^-, \nu, \kappa\}$, also $h(X) \in \mathcal{F}$.

For example, because of condition 3 of Definition 2.1, every nick letter in X is both preceded and followed by an element of \mathcal{A}_{\pm} . If such a nick letter is removed from X , by either ν^+ , ν^- or ν , these elements of \mathcal{A}_{\pm} become adjacent and this does not violate any condition of Definition 2.1.

Since we are really interested in \mathcal{F} , we will consider the restriction of the functions ν^+ , ν^- , ν and κ to this subdomain. In order not to burden our notation too much, we will still use the notation ν^+ , ν^- , ν and κ , respectively for these restricted functions, instead of $\nu^+|_{\mathcal{F}}$, etc. – this should, however, not lead to confusion.

For the composition of functions from $\{\nu^+, \nu^-, \nu, \kappa\}$ with the functions L and R we have the following results (they follow directly from the definitions of L , R , ν^+ , ν^- , ν and κ and the definition of a formal DNA molecule):

Lemma 2.7 *For each $X \in \mathcal{F}$,*

$$L(\nu^+(X)) = L(\nu^-(X)) = L(\nu(X)) = L(X),$$

$$R(\nu^+(X)) = R(\nu^-(X)) = R(\nu(X)) = R(X),$$

$$L(\kappa(X)), R(\kappa(X)) \in \mathcal{A}_{\pm}.$$

2.6 Operators and DNA expressions

The formal DNA molecules constitute the foundation of our DNA language. They allow us to define the elements of the DNA language: the DNA expressions.

The basic building blocks of DNA expressions are \mathcal{N} -words. DNA expressions result by applying operators to \mathcal{N} -words. The set of operators we consider in this thesis is $\mathcal{O} = \{\uparrow, \downarrow, \updownarrow\}$. DNA expressions also contain opening and closing brackets: \langle and \rangle , which delimit the scope of the operators – each (occurrence of an) operator acts only on the part of the expression that is contained between its opening and closing brackets. Hence, the set of all DNA expressions, denoted by \mathcal{D} , is a language over the alphabet $\Sigma_{\mathcal{D}}$, where $\Sigma_{\mathcal{D}} = \mathcal{N} \cup \mathcal{O} \cup \{\langle, \rangle\} = \{A, C, G, T, \uparrow, \downarrow, \updownarrow, \langle, \rangle\}$.

We will use the symbol E (possibly with a subscript) to denote a DNA expression. If a string can be either an \mathcal{N} -word or a DNA expression, then we use ε (possibly with a subscript) to denote it.

Informally, a DNA expression is a string of the form $\langle \uparrow \varepsilon_1 \varepsilon_2 \dots \varepsilon_n \rangle$, $\langle \downarrow \varepsilon_1 \varepsilon_2 \dots \varepsilon_n \rangle$ or $\langle \updownarrow \varepsilon_1 \rangle$, where $n \geq 1$ and the ε_i 's are either \mathcal{N} -words or DNA expressions themselves. The ε_i 's are called the *arguments* of the operator involved. We say that an operator is *applied* to its arguments. The arguments of the operators \uparrow and \downarrow must satisfy certain conditions, which will be explained shortly.

Clearly, not every string over $\Sigma_{\mathcal{D}}$ is a DNA expression. In particular, every DNA expression contains brackets and at least one operator, which implies that \mathcal{N} -words are not DNA expressions.

If E is a DNA expression, then the *semantics* of E , denoted by $\mathcal{S}(E)$, is the formal DNA molecule represented by E . For every DNA expression, there will be exactly one such formal DNA molecule, so \mathcal{S} is a mapping from the DNA language into \mathcal{F} .

Properties of formal DNA molecules carry over in a natural way to DNA expressions by the following convention:

$$\begin{array}{l} \text{property } P \text{ holds for a DNA expression } E_1 \text{ (DNA expressions } E_1 \text{ and } E_2) \\ \iff \\ \text{property } P \text{ holds for } \mathcal{S}(E_1) \text{ (} \mathcal{S}(E_1) \text{ and } \mathcal{S}(E_2), \text{ respectively).} \end{array}$$

Thus, e.g., we may say that the upper strand of DNA expression E_1 strictly covers the lower strand to the right, or that DNA expression E_1 prefits DNA expression E_2 by upper strands. We can also extend the definition of the functions L and R to DNA expressions. If E is a DNA expression, then $L(E) = L(\mathcal{S}(E))$ and $R(E) = R(\mathcal{S}(E))$. The reader must take good notice of this. Often, in a statement or in a proof, we will implicitly make the step from a DNA expression to its semantics, e.g., from $L(E)$ to $L(\mathcal{S}(E))$, or vice versa.

Before we present the formal definition of a DNA expression, we want to provide some intuition for the action of the three operators and for the restrictions that are imposed onto their arguments.

The most elementary expressions in our DNA language are the applications of the operators to a (single) \mathcal{N} -word α : $\langle \uparrow \alpha \rangle$, $\langle \downarrow \alpha \rangle$ and $\langle \updownarrow \alpha \rangle$. The expression $\langle \uparrow \alpha \rangle$ denotes the upper \mathcal{A} -word $\binom{\alpha}{-}$ (which, in turn, denotes the strand 5'- α -3'), $\langle \downarrow \alpha \rangle$ denotes the lower \mathcal{A} -word $\binom{-}{\alpha}$ (the strand 3'- α -5'), and $\langle \updownarrow \alpha \rangle$ denotes the double \mathcal{A} -word $\binom{\alpha}{c(\alpha)}$ with upper strand α (the double-stranded DNA molecule $\begin{array}{c} 5' - \alpha - 3' \\ 3' - c(\alpha) - 5' \end{array}$ without nicks).

For example, if $\alpha = \text{ACATG}$, then $\langle \uparrow \alpha \rangle$ denotes $\binom{\text{ACATG}}{-}$, $\langle \downarrow \alpha \rangle$ denotes $\binom{-}{\text{ACATG}}$ and $\langle \updownarrow \alpha \rangle$ denotes $\binom{\text{ACATG}}{\text{TGTAC}}$.

In the basic DNA expressions, the three operators have one argument, an \mathcal{N} -word α . In general, however, the operators \uparrow and \downarrow may have more than one argument. Moreover, the arguments of an operator do not have to be \mathcal{N} -words; they may also be DNA expressions. Then, starting from the simple, basic DNA expressions, one can build more and more complex DNA expressions. There are, however, some restrictions on the arguments, which we will describe now for each of the operators.

The operator \uparrow can have an arbitrary number $n \geq 1$ of arguments. Each argument ε_i ($i = 1, 2, \dots, n$) must be either an \mathcal{N} -word α , or a DNA expression E . We further demand that for $i = 1, 2, \dots, n - 1$ the argument ε_i prefits ε_{i+1} by upper strands. Since we have defined 'prefitting each other by upper strands' only for formal DNA

$$\left\langle \uparrow \begin{array}{c} \text{C} \\ \text{G} \end{array} \quad \text{AT} \quad \begin{array}{c} \overline{\text{G}}\text{C} \\ \text{CG} \end{array} \right\rangle = \begin{array}{c} \text{CATGC} \\ \text{G} \quad \text{CG} \end{array} \quad \left\langle \uparrow \begin{array}{c} \text{A} \\ \text{T} \end{array} \quad \begin{array}{c} \text{T} \\ \text{A} \end{array} \right\rangle = \begin{array}{c} \text{AT} \\ \text{T}\overline{\text{A}} \end{array} \quad (\text{a})$$

$$\left\langle \downarrow \begin{array}{c} \text{T} \\ \text{G} \end{array} \quad \begin{array}{c} \text{CATGC} \\ \text{G} \quad \text{CG} \end{array} \quad \begin{array}{c} \text{AT} \\ \text{T}\overline{\text{A}} \end{array} \right\rangle = \begin{array}{c} \text{CATGC}\overline{\text{A}}\text{T} \\ \text{TG} \quad \text{CGTA} \end{array} \quad (\text{b})$$

$$\left\langle \updownarrow \begin{array}{c} \text{CATGC}\overline{\text{A}}\text{T} \\ \text{TG} \quad \text{CGTA} \end{array} \right\rangle = \begin{array}{c} \text{ACATGC}\overline{\text{A}}\text{T} \\ \text{TGTACGTA} \end{array} \quad (\text{c})$$

Figure 2.5: Examples of (a) the action of the operator \uparrow ; (b) the action of the operator \downarrow ; (c) the action of the operator \updownarrow .

molecules and for DNA expressions, we consider an \mathcal{N} -word α here as the DNA expression $\langle \uparrow \alpha \rangle$, which represents the upper \mathcal{A} -word $\binom{\alpha}{-}$. The resulting DNA expression is $\langle \uparrow \varepsilon_1 \varepsilon_2 \dots \varepsilon_n \rangle$.

From the molecular point of view, the operator \uparrow connects all pairs of adjacent nucleotides in the upper strands of its arguments. So it repairs all nicks inside these strands by establishing the missing phosphodiester bonds and it fixes such connections between the upper strands of the successive arguments. This is illustrated by the first example of Figure 2.5(a). However, nicks that are present in the lower strands of its arguments are not repaired. As a matter of fact, the operator \uparrow introduces nicks between the lower strands of successive arguments if these successive arguments happen to prefit each other by lower strands, i.e., if they have a blunt edge at each other's side. The second example of Figure 2.5(a) shows such a situation.

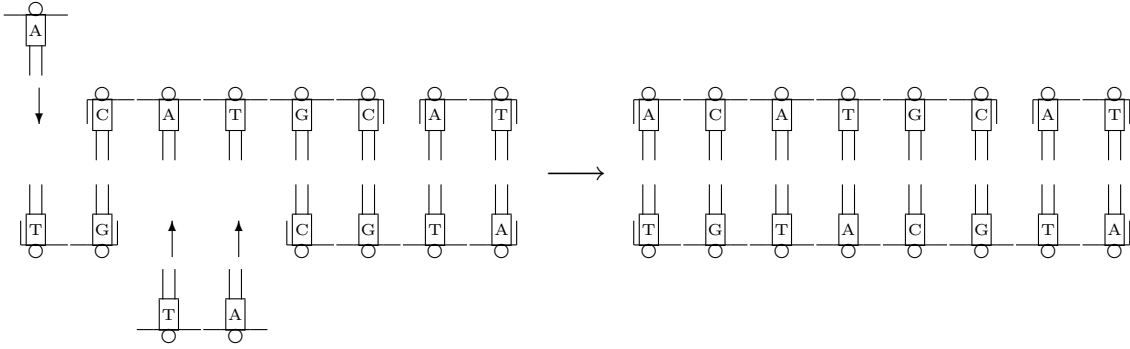
We have to say that establishing phosphodiester bonds within one strand only is not realistic from the molecular point of view.

The operator \downarrow is the dual of \uparrow . It can have an arbitrary number $n \geq 1$ of arguments, with each argument ε_i ($i = 1, \dots, n$) being either an \mathcal{N} -word or a DNA expression. Here we require ε_i to prefit ε_{i+1} by lower strands for $i = 1, 2, \dots, n-1$. Further, when an argument ε_i is an \mathcal{N} -word α , it is interpreted as the DNA expression $\langle \downarrow \alpha \rangle$, which denotes the lower \mathcal{A} -word $\binom{-}{\alpha}$. The resulting DNA expression is $\langle \downarrow \varepsilon_1 \varepsilon_2 \dots \varepsilon_n \rangle$. The effect of this operator is similar to that of \uparrow ; the only difference is that the roles of the upper strands and the lower strands of the arguments are changed. This is illustrated by Figure 2.5(b).

Unlike the other two operators, \updownarrow can have only one argument ε_1 . It is either an \mathcal{N} -word or an (arbitrary) DNA expression. The resulting DNA expression is $\langle \updownarrow \varepsilon_1 \rangle$.

If ε_1 is a DNA expression E , then, intuitively, in the DNA molecule denoted by E , the operator \updownarrow provides a complementary nucleotide for every nucleotide which is not yet complemented. So it fills up every gap in the DNA molecule. Further, the operator establishes phosphodiester bonds between the nucleotides added and their respective neighbours in the strand. Hence, it does not introduce new nicks. On the other hand, if the DNA molecule denoted by E has nicks already, these nicks are not repaired by \updownarrow . The action of this operator is illustrated in Figure 2.5(c).

The basic DNA expression $\langle \updownarrow \alpha \rangle$ was the result of applying \updownarrow to an \mathcal{N} -word α . This result can also be explained in terms of complements, as follows: if the argument of \updownarrow is an \mathcal{N} -word α , the operator conceives it as the DNA expression $\langle \uparrow \alpha \rangle$ and then performs the same action as for 'ordinary' DNA expressions.

Figure 2.6: Pictorial representation of the action of the operator \updownarrow .

The notation \updownarrow may be a bit misleading. It may suggest to be a combination of the operators \uparrow and \downarrow . It would, e.g., repair nicks in both upper and lower strands then, like the function ν does with formal DNA molecules. In fact, an operator with such effect might be more realistic than the separate operators \uparrow and \downarrow that we have, as this effect comes closer to the action of the enzyme ligase than the separate effects of \uparrow and \downarrow . In this thesis, however, we will build a theory with the operators \uparrow , \downarrow and \updownarrow as we have introduced them.

There is a nice pictorial interpretation of the operators' actions. We can consider a nucleotide as a puppet, the phosphate group at the 5'-site and the hydroxyl group at the 3'-site being its arms. When there is a horizontal connection between two adjacent nucleotides, we can view that as if both puppets raised one arm and joined hands. A phosphate group or a hydroxyl group that is not used for a phosphodiester bond corresponds to an arm hanging down. So in case of a nick, the two nucleotides involved keep the arm on the other one's side down.

Now when the \uparrow -operator is applied, the puppets in the upper strand raise their arms and, if there is an adjacent puppet, they connect. The action of the \downarrow -operator can be viewed similarly. Finally, when the \updownarrow -operator complements a nucleotide, it inserts a puppet with both arms raised. Either of these arms seizes the arm of a neighbour and makes a connection. This case is depicted in Figure 2.6.

We are ready now to give a formal definition of DNA expressions and their semantics.

Definition 2.8 A DNA expression is a string in any of the following forms:

- $\langle \uparrow \varepsilon_1 \varepsilon_2 \dots \varepsilon_n \rangle$,
where $n \geq 1$, ε_i is an \mathcal{N} -letter or a DNA expression for $i = 1, 2, \dots, n$, and $\mathcal{S}^+(\varepsilon_i) \sqsubseteq \mathcal{S}^+(\varepsilon_{i+1})$ for $i = 1, 2, \dots, n-1$, where the function \mathcal{S}^+ is defined by

$$\mathcal{S}^+(\varepsilon) = \begin{cases} \begin{pmatrix} \alpha \\ - \end{pmatrix} & \text{if } \varepsilon \text{ is an } \mathcal{N}\text{-word } \alpha \\ \mathcal{S}(\varepsilon) & \text{if } \varepsilon \text{ is a DNA expression} \end{cases} \quad (2.13)$$

Further,

$$\mathcal{S}(\langle \uparrow \varepsilon_1 \varepsilon_2 \dots \varepsilon_n \rangle) = \nu^+(\mathcal{S}^+(\varepsilon_1))y_1\nu^+(\mathcal{S}^+(\varepsilon_2))y_2 \dots y_{n-1}\nu^+(\mathcal{S}^+(\varepsilon_n)) \quad (2.14)$$

with

$$y_i = \begin{cases} \triangle & \text{if } \mathcal{S}^+(\varepsilon_i) \sqsubseteq \mathcal{S}^+(\varepsilon_{i+1}), \text{ i.e., if both } R(\mathcal{S}^+(\varepsilon_i)) \in \mathcal{A}_\pm \\ & \text{and } L(\mathcal{S}^+(\varepsilon_{i+1})) \in \mathcal{A}_\pm \\ \lambda & \text{otherwise, i.e., if either } R(\mathcal{S}^+(\varepsilon_i)) \in \mathcal{A}_+ \\ & \text{or } L(\mathcal{S}^+(\varepsilon_{i+1})) \in \mathcal{A}_+ \text{ (or both)} \end{cases} \quad (2.15)$$

$$(i = 1, 2, \dots, n-1).$$

- $\langle \downarrow \varepsilon_1 \varepsilon_2 \dots \varepsilon_n \rangle$,
where $n \geq 1$, ε_i is an \mathcal{N} -letter or a DNA expression for $i = 1, 2, \dots, n$, and $\mathcal{S}^-(\varepsilon_i) \sqsubseteq \mathcal{S}^-(\varepsilon_{i+1})$ for $i = 1, 2, \dots, n-1$ where the function \mathcal{S}^- is defined by

$$\mathcal{S}^-(\varepsilon) = \begin{cases} \binom{-}{\alpha} & \text{if } \varepsilon \text{ is an } \mathcal{N}\text{-word } \alpha \\ \mathcal{S}(\varepsilon) & \text{if } \varepsilon \text{ is a DNA expression} \end{cases}. \quad (2.16)$$

Further,

$$\mathcal{S}(\langle \downarrow \varepsilon_1 \varepsilon_2 \dots \varepsilon_n \rangle) = \nu^-(\mathcal{S}^-(\varepsilon_1))y_1\nu^-(\mathcal{S}^-(\varepsilon_2))y_2 \dots y_{n-1}\nu^-(\mathcal{S}^-(\varepsilon_n))$$

with

$$y_i = \begin{cases} \nabla & \text{if } \mathcal{S}^-(\varepsilon_i) \sqsubseteq \mathcal{S}^-(\varepsilon_{i+1}), \text{ i.e., if both } R(\mathcal{S}^-(\varepsilon_i)) \in \mathcal{A}_\pm \\ & \text{and } L(\mathcal{S}^-(\varepsilon_{i+1})) \in \mathcal{A}_\pm \\ \lambda & \text{otherwise, i.e., if either } R(\mathcal{S}^-(\varepsilon_i)) \in \mathcal{A}_- \\ & \text{or } L(\mathcal{S}^-(\varepsilon_{i+1})) \in \mathcal{A}_- \text{ (or both)} \end{cases}$$

$$(i = 1, 2, \dots, n-1).$$

- $\langle \uparrow \varepsilon_1 \rangle$,
where ε_1 is either an \mathcal{N} -word or a DNA expression.

Further,

$$\mathcal{S}(\langle \uparrow \varepsilon_1 \rangle) = \kappa(\mathcal{S}^+(\varepsilon_1)).$$

for the function \mathcal{S}^+ defined above.

An example of a DNA expression which uses all three operators, is

$$\langle \downarrow T \langle \uparrow \langle \uparrow C \rangle AT \langle \downarrow \langle \uparrow G \rangle \langle \uparrow C \rangle \rangle \rangle \langle \uparrow \langle \uparrow A \rangle \langle \uparrow T \rangle \rangle. \quad (2.17)$$

It represents the DNA molecule of Figure 2.5(b).

We call a DNA expression of the form $\langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$ a \uparrow -expression, one of the form $\langle \downarrow \varepsilon_1 \dots \varepsilon_n \rangle$ a \downarrow -expression, and one of the form $\langle \uparrow \varepsilon_1 \rangle$ a \uparrow -expression. Thus, DNA expression (2.17) is a \downarrow -expression.

Note that, in the informal description of the expressions $\langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$ and $\langle \downarrow \varepsilon_1 \dots \varepsilon_n \rangle$, we allowed the ε_i 's to be \mathcal{N} -words of arbitrary length, whereas, in the definition above, we require them to be \mathcal{N} -letters, hence \mathcal{N} -words of length 1 (if they are not DNA expressions). We will eliminate this discrepancy soon.

Note that for each of the three operators, we write the result of its application in a kind of prefix style, i.e., we write the operator *before* its argument(s). This style is only violated by the closing bracket \rangle , which typically belongs to the operator but is written *after* the arguments. This violation is hard to avoid; since the number of arguments of the operators \uparrow and \downarrow is not fixed, we have to explicitly delimit the scope of these operators. And a closing bracket is quite appropriate for that.

The formal DNA molecule $\mathcal{S}^+(\varepsilon)$, occurring in the definition of a DNA expression of the form $\langle \uparrow \varepsilon_1 \varepsilon_2 \dots \varepsilon_n \rangle$, can be considered as a kind of ‘upper semantics’ of the argument ε . Similarly, the formal DNA molecule $\mathcal{S}^-(\varepsilon)$, occurring in the definition of a DNA expression of the form $\langle \downarrow \varepsilon_1 \varepsilon_2 \dots \varepsilon_n \rangle$, can be considered as a kind of ‘lower semantics’ of the argument ε .

If we define functions Exp^+ and Exp^- by

$$\text{Exp}^+(\varepsilon) = \begin{cases} \langle \uparrow \alpha \rangle & \text{if } \varepsilon \text{ is an } \mathcal{N}\text{-word } \alpha \\ \varepsilon & \text{if } \varepsilon \text{ is a DNA expression} \end{cases} \quad (2.18)$$

and

$$\text{Exp}^-(\varepsilon) = \begin{cases} \langle \downarrow \alpha \rangle & \text{if } \varepsilon \text{ is an } \mathcal{N}\text{-word } \alpha \\ \varepsilon & \text{if } \varepsilon \text{ is a DNA expression} \end{cases}, \quad (2.19)$$

then it is easy to see that for every \mathcal{N} -word or DNA expression ε , $\mathcal{S}^+(\varepsilon) = \mathcal{S}(\text{Exp}^+(\varepsilon))$ and $\mathcal{S}^-(\varepsilon) = \mathcal{S}(\text{Exp}^-(\varepsilon))$. The DNA expressions $\text{Exp}^+(\varepsilon)$ and $\text{Exp}^-(\varepsilon)$ can be considered as a kind of ‘upper DNA expression’ and ‘lower DNA expression’ corresponding to ε , respectively.

Note finally that, indeed, the operator \uparrow does not introduce nicks in its argument. This follows from the fact that there could not be a nick next to a missing nucleotide *before* the addition of this nucleotide (by condition 3 of Definition 2.1), and from the fact that the function κ does not introduce new nicks.

We now introduce the possibility for \uparrow -expressions (and \downarrow -expressions) to have arguments that are \mathcal{N} -words α of length $|\alpha| \geq 2$. Consider a DNA expression $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_{i_0-1} \varepsilon_{i_0} \dots \varepsilon_{j_0} \varepsilon_{j_0+1} \dots \varepsilon_n \rangle$ with $1 \leq i_0 < j_0 \leq n$, where the consecutive arguments $\varepsilon_{i_0}, \dots, \varepsilon_{j_0}$ are \mathcal{N} -letters a_{i_0}, \dots, a_{j_0} , respectively. Then, to simplify the notation for E , we might replace the sequence $a_{i_0} \dots a_{j_0}$ by the symbol α , yielding $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_{i_0-1} \alpha \varepsilon_{j_0+1} \dots \varepsilon_n \rangle$. Because Definition 2.8 does not allow \mathcal{N} -words α of length $|\alpha| \geq 2$ to be arguments of the operator \uparrow , α would be merely an abbreviation for the $j_0 - i_0 + 1$ arguments a_{i_0}, \dots, a_{j_0} . For evaluating (i.e., determining the semantics of) E , we would have to consider the arguments a_i separately. Hence, we have

$$\begin{aligned} \mathcal{S}(E) &= \mathcal{S}(\langle \uparrow \varepsilon_1 \dots \varepsilon_{i_0-1} \alpha \varepsilon_{j_0+1} \dots \varepsilon_n \rangle) = \nu^+(\mathcal{S}^+(\varepsilon_1))y_1 \dots y_{i_0-2} \nu^+(\mathcal{S}^+(\varepsilon_{i_0-1})) \cdot \\ & y_{i_0-1} \nu^+(\mathcal{S}^+(\varepsilon_{i_0}))y_{i_0} \dots y_{j_0-1} \nu^+(\mathcal{S}^+(\varepsilon_{j_0}))y_{j_0} \cdot \nu^+(\mathcal{S}^+(\varepsilon_{j_0+1}))y_{j_0+1} \dots y_{n-1} \nu^+(\mathcal{S}^+(\varepsilon_n)), \end{aligned}$$

where the y_i 's are defined by (2.15).

Let us zoom in on the substring $y_{i_0-1} \nu^+(\mathcal{S}^+(\varepsilon_{i_0}))y_{i_0} \dots y_{j_0-1} \nu^+(\mathcal{S}^+(\varepsilon_{j_0}))y_{j_0}$ of $\mathcal{S}(E)$.¹ As $\varepsilon_i = a_i \in \mathcal{N}$ for $i = i_0, \dots, j_0$, we have $\mathcal{S}^+(\varepsilon_i) = \binom{a_i}{-}$ for these i 's. This implies that also $\nu^+(\mathcal{S}^+(\varepsilon_i)) = \binom{a_i}{-}$ for $i = i_0, \dots, j_0$, and, by the definition of y_i , that $y_i = \lambda$

¹If $i_0 = 1$ or $j_0 = n$, then, of course, y_{i_0-1} or y_{j_0} , respectively, does not exist. In our considerations, we may substitute λ for the respective variable then.

for $i = i_0 - 1, \dots, j_0$. Thus, our substring reduces to $\binom{a_{i_0}}{-} \cdots \binom{a_{j_0}}{-} = \binom{a_{i_0} \cdots a_{j_0}}{-} = \binom{\alpha}{-}$ and

$$\mathcal{S}(E) = \mathcal{S}(\langle \uparrow \varepsilon_1 \dots \varepsilon_{i_0-1} \alpha \varepsilon_{j_0+1} \dots \varepsilon_n \rangle) = \\ \nu^+(\mathcal{S}^+(\varepsilon_1))y_1 \dots y_{i_0-2} \nu^+(\mathcal{S}^+(\varepsilon_{i_0-1})) \binom{\alpha}{-} \nu^+(\mathcal{S}^+(\varepsilon_{j_0+1}))y_{j_0+1} \dots y_{n-1} \nu^+(\mathcal{S}^+(\varepsilon_n)).$$

A similar result can be obtained if consecutive arguments $\varepsilon_{i_0}, \dots, \varepsilon_{j_0}$ of the operator \downarrow are \mathcal{N} -letters. This motivates a generalization of the definition of a DNA expression. From now on, we allow the arguments ε_i of the operators \uparrow and \downarrow to be \mathcal{N} -words of arbitrary length. As the definitions of the functions \mathcal{S}^+ and \mathcal{S}^- and the variables y_i are already suited for such ε_i 's, we do not have to make any other modification to Definition 2.8. Thus the new definition of a DNA expression is

Definition 2.9 *A DNA expression is a string in any of the following forms:*

- $\langle \uparrow \varepsilon_1 \varepsilon_2 \dots \varepsilon_n \rangle$,
where $n \geq 1$, ε_i is an \mathcal{N} -word or a DNA expression for $i = 1, 2, \dots, n$, and $\mathcal{S}^+(\varepsilon_i) \sqsubseteq \mathcal{S}^+(\varepsilon_{i+1})$ for $i = 1, 2, \dots, n-1$ and the function \mathcal{S}^+ from (2.13).
- $\langle \downarrow \varepsilon_1 \varepsilon_2 \dots \varepsilon_n \rangle$,
where $n \geq 1$, ε_i is an \mathcal{N} -word or a DNA expression for $i = 1, 2, \dots, n$, and $\mathcal{S}^-(\varepsilon_i) \sqsubseteq \mathcal{S}^-(\varepsilon_{i+1})$ for $i = 1, 2, \dots, n-1$ and the function \mathcal{S}^- from (2.16).
- $\langle \updownarrow \varepsilon_1 \rangle$,
where ε_1 is either an \mathcal{N} -word or a DNA expression.

For each type of DNA expression (i.e., $\langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$, $\langle \downarrow \varepsilon_1 \dots \varepsilon_n \rangle$ or $\langle \updownarrow \varepsilon_1 \rangle$) the semantics is defined by the same equation(s) as in Definition 2.8. One can verify that, indeed, this semantics is always a formal DNA molecule.

Now, the formal language \mathcal{D} is the set of all DNA expressions as defined in Definition 2.9.

Let us return to the DNA expression $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_{i_0-1} \alpha \varepsilon_{j_0+1} \dots \varepsilon_n \rangle$ with $\alpha = a_{i_0} \dots a_{j_0}$ and $j_0 > i_0$. If we consider the \mathcal{N} -word α itself as an argument of \uparrow (which is allowed now), then again we obtain

$$\mathcal{S}(E) = \nu^+(\mathcal{S}^+(\varepsilon_1))y_1 \dots y_{i_0-2} \nu^+(\mathcal{S}^+(\varepsilon_{i_0-1})) \binom{\alpha}{-} \nu^+(\mathcal{S}^+(\varepsilon_{j_0+1}))y_{j_0+1} \dots y_{n-1} \nu^+(\mathcal{S}^+(\varepsilon_n)).$$

Hence, whether we interpret α as one argument or as a sequence of $j_0 - i_0 + 1$ separate arguments, the semantics of our example DNA expression is the same. So the extension of our definition has not caused semantical problems.

By a similar argument, this can be generalized to the following result:

Theorem 2.10 *Let $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_{i_0-1} \varepsilon_{i_0} \dots \varepsilon_{j_0} \varepsilon_{j_0+1} \dots \varepsilon_n \rangle$ be a DNA expression where ε_i is either an \mathcal{N} -word or a DNA expression for $i = 1, \dots, i_0-1, j_0+1, \dots, n$ and $\varepsilon_i = \alpha_i$ is an \mathcal{N} -word for $i = i_0, \dots, j_0$. Let $\alpha = \alpha_{i_0} \dots \alpha_{j_0}$. Then $\mathcal{S}(E)$ is the same, regardless of the interpretation of α as one argument or as a sequence of separate arguments $\alpha_{i_0}, \dots, \alpha_{j_0}$.*

Of course, an analogous result holds for \downarrow -expressions.

Thus, although the admission of \mathcal{N} -words α of length $|\alpha| \geq 2$ as arguments of \uparrow and \downarrow does introduce ambiguity with respect to the question what the arguments of an operator are, there can still be no doubt about the (formal) DNA molecule denoted by a DNA expression.

Note that the interpretation of \mathcal{N} -words α of length $|\alpha| \geq 2$ as argument(s) of an operator is unambiguous for the operator \Downarrow , because this operator can have only one argument.

For example, let $E = \langle \uparrow \text{ACATG} \rangle$. Then under Definition 2.8, we should interpret E as a \uparrow -expression with five arguments $a_1 = \text{A}$, $a_2 = \text{C}$, $a_3 = \text{A}$, $a_4 = \text{T}$ and $a_5 = \text{G}$. Now, there are many more possible interpretations. We may, e.g., interpret E as $\langle \uparrow \alpha_1 \alpha_2 \rangle$ with two arguments $\alpha_1 = \text{AC}$ and $\alpha_2 = \text{ATG}$, as $\langle \uparrow \alpha_1 \alpha_2 \rangle$ with two arguments $\alpha_1 = \text{ACAT}$ and $\alpha_2 = \text{G}$, or as $\langle \uparrow \alpha_1 \rangle$ with only one argument $\alpha_1 = \text{ACATG}$. Whatever interpretation we choose, $\mathcal{S}(E) = \left(\begin{smallmatrix} \text{ACATG} \\ - \end{smallmatrix} \right)$.

In particular, we are free to interpret consecutive \mathcal{N} -words in a DNA expression as one \mathcal{N} -word. This motivates the definition of a *maximal \mathcal{N} -word occurrence in a string X* (for instance a DNA expression E) as an occurrence (X_1, X_2) of an \mathcal{N} -word α in X such that (1) if $X_1 \neq \lambda$ then $R(X_1) \notin \mathcal{N}$ and (2) if $X_2 \neq \lambda$ then $L(X_2) \notin \mathcal{N}$. Hence, the \mathcal{N} -word α ‘cannot be extended either to the left or to the right’.

For example, in the DNA expression

$$\langle \downarrow \text{T} \langle \uparrow \langle \downarrow \text{C} \rangle \text{AT} \langle \downarrow \text{GCAT} \rangle \rangle \rangle$$

the first occurrence of C and the first occurrence of AT are maximal \mathcal{N} -word occurrences. This is, however, not the case with the second occurrences of these \mathcal{N} -words, as they can be extended to GCAT .

We say that an operator *governs* its argument(s) and everything inside its argument(s). In every DNA expression we can identify an outermost operator. This is the operator which has been performed last. It governs the entire DNA expression.

Because of the 1–1 correspondence between a DNA expression and its outermost operator, we will sometimes interchange the terms. In particular, we may speak of the *arguments of a DNA expression*, while we actually mean the arguments of the outermost operator of a DNA expression. For instance, the (three) arguments of DNA expression (2.17) are T , $\langle \uparrow \langle \downarrow \text{C} \rangle \text{AT} \langle \downarrow \langle \downarrow \text{G} \rangle \langle \downarrow \text{C} \rangle \rangle \rangle$ and $\langle \uparrow \langle \downarrow \text{A} \rangle \langle \downarrow \text{T} \rangle \rangle$.

We call (an occurrence of) an operator in a DNA expression E which is not the outermost operator, an *inner occurrence* of this operator in E .

An operator may occur more than once in a DNA expression. To denote a specific occurrence of an operator, we may provide the operator with an index. For example, we may have \uparrow_0 or \downarrow_1 .

A *DNA subexpression E^s* of a DNA expression E is a substring of E which is itself a DNA expression. If $E^s \neq E$, we call E^s a *proper DNA subexpression* of E . Clearly, the outermost operator of a proper DNA subexpression of E is an inner occurrence of this operator in E .

We will use the term *\uparrow -subexpression* of E to refer to a DNA subexpression of E which is a \uparrow -expression. Analogously, we may have a *\downarrow -subexpression* and a *\Downarrow -subexpression* of E .

For every \mathcal{N} -word α occurring in a DNA expression E and for every proper DNA subexpression E^s of E we define its *parent operator* to be the operator which has the \mathcal{N} -word or DNA subexpression as an immediate argument. For instance, in DNA

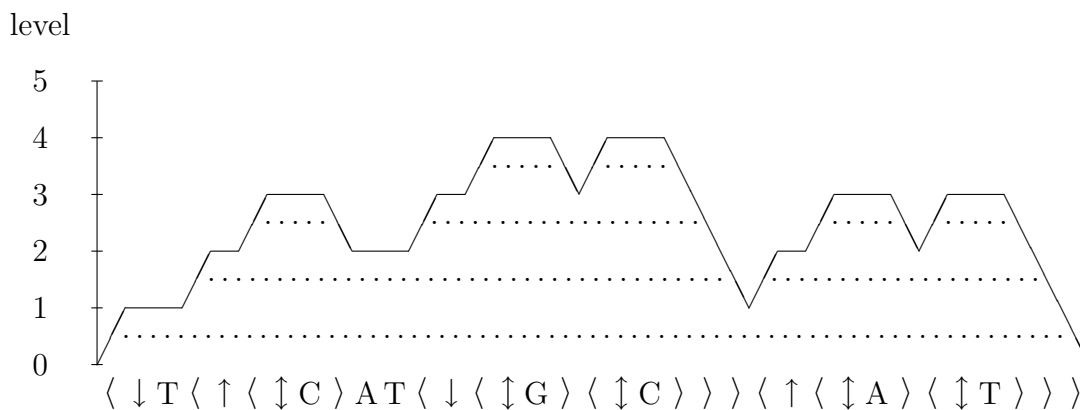


Figure 2.7: Level structure for DNA expression (2.17); horizontal dotted lines connect level changes due to pairs of corresponding brackets.

expression (2.17) the parent operator of the \mathcal{N} -word AT is the first occurrence of the operator \uparrow in the DNA expression; for the last occurrence of the \mathcal{N} -word T it is clearly the operator \downarrow standing in front of it; and the parent operator of the DNA subexpression $\langle \downarrow G \rangle$ is the second occurrence of the operator \downarrow .

If an argument of a certain (occurrence of an) operator is a \uparrow -expression, then we may call this argument a \uparrow -argument of the operator. In an analogous way, we define a \downarrow -argument and a \downarrow -argument of an operator. If an argument of an operator is an \mathcal{N} -word, then we may call it an \mathcal{N} -word-argument of the operator.

2.7 Brackets, arguments and DNA subexpressions

The brackets in a DNA expression can be thought of as determining a structure with different levels. An opening bracket \langle corresponds to an increase of the level by 1, a closing bracket \rangle to a decrease of the level by 1. The resulting levels are also known as the nesting levels of the brackets.

It is natural to start a DNA expression at level 0. Since every opening bracket precedes the corresponding closing bracket, the level in a DNA expression is always non-negative. Further, because the number of opening brackets equals the number of closing brackets, the level is back at 0 at the end of a DNA expression. In Figure 2.7 we show the different levels for DNA expression (2.17).

The notion of the level of a DNA expression can be used for identifying substrings of a DNA expression. This is done in the following two results, which we state without a proof.

Lemma 2.11 *Suppose that the opening bracket of a DNA subexpression E^s of a DNA expression E raises the level of E from $l - 1$ to l for a certain positive integer l . Then the closing bracket of E^s is the first one after this opening bracket to lower the level from l to $l - 1$. In particular, between the opening bracket and the closing bracket of E^s , the level is at least l .*

To illustrate this lemma, we have drawn dotted lines between corresponding increases and decreases of the level in Figure 2.7.

$$\begin{array}{ll}
E: & \langle \dots \dots \dots \rangle \\
E_1^s: & \langle \dots \dots \dots \rangle \\
E_2^s: & \langle \dots \dots \dots \rangle
\end{array}$$

Figure 2.8: Schematic representation of two (hypothetically) overlapping DNA subexpressions E_1^s and E_2^s of a DNA expression E

Theorem 2.12 *Let $|_0$ be an operator at level l of the level structure of a DNA expression E . Then (an occurrence of) a substring between $|_0$ and the closing bracket of $|_0$ is an argument of $|_0$ if and only if*

- *either it is a maximal \mathcal{N} -word occurrence in E at level l*
- *or it starts with an opening bracket raising the level from l to $l+1$ and ends with the corresponding closing bracket.*

Clearly, as every DNA (sub-)expression is of the form $\langle |_0 \varepsilon_1 \dots \varepsilon_n \rangle$ for an operator $|_0$ and arguments $\varepsilon_1, \dots, \varepsilon_n$, the arguments are indeed substrings between $|_0$ and the closing bracket of $|_0$. Hence, this theorem covers all arguments of $|_0$.

The theorem is important, because it enables us to determine the structure of a DNA expression, i.e., how the DNA expression has been built up, even though it is just a sequence of symbols.

The proof of this theorem relies on the observation that, without loss of generality, consecutive \mathcal{N} -words α_i in a DNA expression can be considered as one \mathcal{N} -word (see Theorem 2.10), and on Lemma 2.11. Lemma 2.11 is also useful to prove the following result:

Theorem 2.13 *Two (occurrences of) DNA subexpressions in a DNA expression E cannot overlap. So either one is contained in the other, or they do not have a common (occurrence of a) substring at all.*

Proof: Suppose that two DNA subexpressions E_1^s and E_2^s do overlap. Without loss of generality we assume that the opening bracket of E_2^s is contained in E_1^s and that the closing bracket is not. This situation is depicted in Figure 2.8.

The opening bracket of E_1^s raises the level of the DNA expression from $l_1 - 1$ to l_1 for a certain positive l_1 and the opening bracket of E_2^s raises the level from $l_2 - 1$ to l_2 . By assumption, this latter bracket is between the opening bracket and the closing bracket of E_1^s . Then, by Lemma 2.11, we must have $l_2 - 1 \geq l_1$.

On the other hand, by the same lemma, the closing bracket of E_1^s lowers the level of the DNA expression from l_1 back to $l_1 - 1$. As this bracket is assumed to be between the opening bracket and the closing bracket of E_2^s , Lemma 2.11 also tells us that $l_1 - 1 \geq l_2$. But then we would have $l_2 - 1 \geq l_1 > l_1 - 1 \geq l_2$, which is not possible. \square

Corollary 2.14 *If E^s is a proper DNA subexpression of a DNA expression E , then E^s is contained in an argument of E .*

Proof: Because E^s is a proper DNA subexpression of E , it is a substring of $\varepsilon_1 \dots \varepsilon_n$, the concatenation of the arguments of E . Let ε_i be the first argument that has a

non-empty intersection with E^s . Then ε_i contains the opening bracket of E^s , which implies that ε_i is a DNA expression (and not an \mathcal{N} -word).

If the opening bracket of E^s is the opening bracket of ε_i , then also the closing brackets must match, so E^s is equal to ε_i . In particular, E^s is contained in ε_i . If the opening bracket of E^s is not the opening bracket of ε_i , then ε_i is clearly not contained in E^s . By Theorem 2.13, E^s must be (properly) contained in ε_i then. \square

We conclude this section with a simple, but useful result:

Lemma 2.15 *Let $E \in \mathcal{D}$ be a DNA expression. Every argument of every operator in E contains at least one \mathcal{N} -word α .*

Proof: Straightforward by induction on the number of operators occurring in an argument. \square

2.8 The functions L and R for DNA expressions

An important aspect of the definition of \uparrow -expressions and \downarrow -expressions is that their arguments must fit together by upper strands or lower strands, respectively (see Definition 2.8). This requirement can be reformulated in terms of $L(E_i)$ and $R(E_i)$, where every E_i is a DNA expression corresponding to the argument ε_i .

The functions L and R that occur in these terms have been defined for formal DNA molecules first, and later, via the function \mathcal{S} , also for DNA expressions. However, when we only want to check if two arguments of an operator fit together by upper strands or lower strands, we are not interested in the complete semantics of these arguments. Therefore, it would be convenient if we could compute $L(E)$ and $R(E)$ for a DNA expression E without computing $\mathcal{S}(E)$ explicitly. Actually, we only need to know whether $L(E)$ and $R(E)$ are elements of \mathcal{A}_+ , \mathcal{A}_- or \mathcal{A}_\pm .

This can be decided in a recursive manner, using the following result:

Lemma 2.16 *Let E be a DNA expression.*

1. If $E = \langle \uparrow \alpha \rangle$ for an \mathcal{N} -word α , then $L(E), R(E) \in \mathcal{A}_+$.
2. If $E = \langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α , then $L(E), R(E) \in \mathcal{A}_-$.
3. If $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$ where $n \geq 1$ and ε_i is an \mathcal{N} -word or a DNA expression for $i = 1, \dots, n$, then $L(E) = L(E_1)$ and $R(E) = R(E_n)$ where $E_1 = \text{Exp}^+(\varepsilon_1)$ and $E_n = \text{Exp}^+(\varepsilon_n)$.
4. If $E = \langle \downarrow \varepsilon_1 \dots \varepsilon_n \rangle$ where $n \geq 1$ and ε_i is an \mathcal{N} -word or a DNA expression for $i = 1, \dots, n$, then $L(E) = L(E_1)$ and $R(E) = R(E_n)$ where $E_1 = \text{Exp}^-(\varepsilon_1)$ and $E_n = \text{Exp}^-(\varepsilon_n)$.
5. If $E = \langle \updownarrow \varepsilon_1 \rangle$ where ε_1 is an \mathcal{N} -word or a DNA expression, then $L(E), R(E) \in \mathcal{A}_\pm$.

Proof:

1. The claim follows immediately from the observation that $\mathcal{S}(\langle \uparrow \alpha \rangle) = \binom{\alpha}{-}$.
2. Analogous to the proof of the previous case.

3. According to the definition of a DNA expression and its semantics,

$$\mathcal{S}(E) = \nu^+(\mathcal{S}^+(\varepsilon_1))y_1 \dots y_{n-1}\nu^+(\mathcal{S}^+(\varepsilon_n))$$

for the y_i 's from (2.15). Consequently,

$$L(E) = L(\nu^+(\mathcal{S}^+(\varepsilon_1))) = L(\mathcal{S}^+(\varepsilon_1)) = L(\mathcal{S}(\text{Exp}^+(\varepsilon_1))) = L(\text{Exp}^+(\varepsilon_1)) = L(E_1).$$

The second equality in this derivation follows from Lemma 2.7.

In a similar way, we find $R(E) = R(E_n)$.

4. Analogous to the proof of the previous case.

5. By the definition of the semantics of a \downarrow -expression, $\mathcal{S}(E) = \kappa(\mathcal{S}^+(\varepsilon_1))$. Hence, $L(E) = L(\kappa(\mathcal{S}^+(\varepsilon_1)))$ and $R(E) = R(\kappa(\mathcal{S}^+(\varepsilon_1)))$. By Lemma 2.7, these are in \mathcal{A}_\pm .

□

2.9 Recognition of DNA expressions

The preceding two sections provide the tools we need to decide whether or not an arbitrary string E over $\Sigma_{\mathcal{D}}$, hence consisting of \mathcal{N} -words α , operators and brackets, is a DNA expression.

According to Definition 2.8, a DNA expression is a string of the form $\langle |_0 \dots \rangle$, where $|_0 \in \mathcal{O}$ is an operator. In particular, the first symbol of the string has to be an opening bracket \langle and the last symbol of the string has to be the corresponding closing bracket \rangle .

A first thing we can do now, is to determine the level structure of the string, like we did in Figure 2.7 for DNA expression (2.17). The first symbol of the string (an opening bracket) raises the level of the string from 0 to 1. By Lemma 2.11, the last symbol of the string (the corresponding closing bracket) must lower the level back from 1 to 0, and the level of the string between these two symbols has to be strictly positive.

There have to be as many operators in the string as there are opening brackets (and closing brackets). Each operator must be immediately preceded by an opening bracket.

Next, by using Theorem 2.12, we can determine the arguments ε_i of the outermost operator $|_0$ of the string. If $|_0$ is \downarrow , then there has to be exactly one argument; if it is either \uparrow or \downarrow , then the number of arguments has to be positive. In particular, we cannot have $E = \langle \uparrow \rangle$, $E = \langle \downarrow \rangle$ or $E = \langle \downarrow \uparrow \rangle$.

For those arguments that are no (maximal) \mathcal{N} -word occurrences, we can check recursively whether they are DNA expressions.

If, up to here, all requirements are met and $|_0$ has only one argument, then the string is a DNA expression. If the number of arguments n is greater than 1 (which implies that $|_0$ is \uparrow or \downarrow), then we have to do some more work. We use Lemmas 2.16, to compute $L(E_i)$ and $R(E_i)$ where the E_i 's are defined by

$$E_i = \begin{cases} \text{Exp}^+(\varepsilon_i) & \text{if } |_0 \text{ is } \uparrow \\ \text{Exp}^-(\varepsilon_i) & \text{if } |_0 \text{ is } \downarrow \end{cases} \quad (i = 1, 2, \dots, n). \quad (2.20)$$

Then, with the $L(E_i)$'s and $R(E_i)$'s we check whether or not the arguments fit together by upper strands (if $|_0$ is \uparrow) or lower strands (if $|_0$ is \downarrow). If so, then the string is a DNA expression; otherwise, it is not.

The entire algorithm for the recognition of a DNA expression takes a time linear in the length of the string. In principle, it can be implemented with one pass through the string. In Appendix A, we give a two-phase implementation. First, we check the positioning of the brackets and the corresponding operators in the string in a straightforward iterative way. Then we recursively check if all arguments of the operators are indeed \mathcal{N} -words or DNA expressions and, if so, if they fit together by upper or lower strands for operators \uparrow and \downarrow , respectively.

2.10 Concatenation of DNA expressions

We have seen that the concatenation of two formal DNA molecules is not necessarily a formal DNA molecule itself. For DNA expressions, the situation is even worse. The mere concatenation of two DNA expressions E_1 and E_2 is *never* a DNA expression, not even if E_1 and E_2 fit together.

This conclusion follows immediately from an examination of the brackets. The first and the last symbol of a DNA expression have to be corresponding opening and closing brackets. However, although the first and the last symbol of the string E_1E_2 are an opening and a closing bracket, respectively, they are not *corresponding* opening and closing brackets.

Thus, E_1E_2 is just a string consisting of two separate DNA expressions. This links up with the (natural) interpretation of DNA expressions as DNA molecules. By putting two DNA molecules in each other's vicinity, we do not automatically get a new DNA molecule. It requires a chemical reaction to achieve that. In the world of DNA expressions, the analogue of such a chemical reaction is an operator. In particular, the operators \uparrow and \downarrow that we have defined can be used to combine two or more DNA expressions into one new DNA expression.

2.11 DNA expressions and trees

Expressions in a context free language have a convenient representation as ordered, directed trees with labelled nodes. Before we discuss the trees that can be used to represent DNA expressions, we want to fix some terminology concerning directed trees in general.

A *directed tree* is a tree with one designated node, which is called the *root* of the tree. A *non-root* in a tree is a node that is not the root of the tree. Let X be a non-root in a directed tree. The nodes on the path from the root of the tree to X (including the root, but excluding X) are the *ancestors* of X . The last node on this path is the *parent* of X . X is called a *child* of its parent. All nodes 'below' X in the tree, i.e., nodes that X is an ancestor of, are called *descendants* of X . The *subtree rooted in X* is the subtree of t consisting of X and *all* descendants of X , together with the arcs connecting these nodes. A *leaf* in a directed tree is a node without descendants. Nodes that do have descendants are called *internal nodes*. We thus have two ways to partition the nodes in a directed tree: either in a root and non-roots, or in leaves and internal nodes.

A directed tree is *ordered* if for each internal node X , the children of X are linearly ordered ('from left to right'). Finally, an *ordered, directed, node-labelled tree* is an

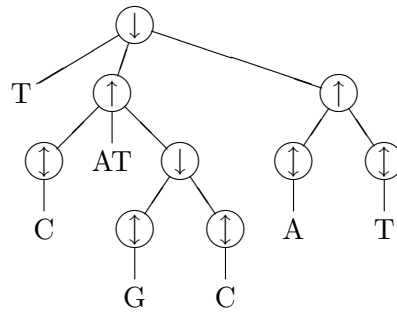


Figure 2.9: The tree of DNA expression (2.17).

ordered directed tree with labels at the nodes.

We now define *the tree of DNA expression* E as follows. For each \mathcal{N} -word α and each operator occurring in E we have a node. The label of this node is the corresponding \mathcal{N} -word or operator. Recall that there is a 1–1 correspondence between (occurrences of) DNA subexpressions and operators in E . Therefore, every node labelled by an operator corresponds to a DNA subexpression of E .

Because of the close relationship (through the labelling function) between nodes on the one hand, and \mathcal{N} -words and operators (or DNA subexpressions) occurring in E on the other hand, we will often say ‘ \mathcal{N} -word’ or ‘operator’ (‘DNA subexpression’) when we actually mean the corresponding node in the tree. This meaning will be clear from the context then.

In the tree we draw arcs from operators to their arguments. By definition, these arguments are \mathcal{N} -words or DNA subexpressions of E . Indeed, as we just noticed, for every occurrence of an \mathcal{N} -word or a DNA subexpression of E , there is a corresponding node. Hence, the arcs are well defined.

Clearly, the node corresponding to an operator is the parent of (the nodes corresponding to) its arguments. These arguments are the children of the operator. If an operator has two or more arguments, its children in the tree are arranged from left to right in the same order as the corresponding arguments in the DNA expression.

Because every \mathcal{N} -word and every proper DNA subexpression of E has exactly one parent operator, we indeed obtain a tree. The leaves of the tree are labelled by the \mathcal{N} -words α in E , the internal nodes by the operators, and, in particular, the root by the outermost operator of E . As an example, in Figure 2.9 we have drawn the tree of DNA expression (2.17).

We just recalled the correspondence between DNA subexpressions and operators. In fact, in the tree of a DNA expression E , a DNA subexpression of E is stored in the subtree rooted in its outermost operator.

The transformation from DNA expressions to trees is injective. This means that when we are given a ‘syntactically correct’ ordered, directed, node-labelled tree, we can perform the inverse transformation. The syntactic demands we impose on the trees are similar to those for a string to be a DNA expression:

- internal nodes are labelled by operators and leaves by \mathcal{N} -words
- a node labelled by \uparrow has only one child
- if a node labelled by \uparrow or \downarrow has two or more children, the DNA subexpressions

corresponding to these children fit together by upper strands or lower strands, respectively.

This final requirement is in fact a recursive one, as it presupposes that the subtrees rooted in the children can be interpreted as DNA expressions.

If this assumption is valid, the prefitting requirement can be checked in a way similar to that for DNA expressions. Suppose that the i^{th} child of a node labelled by \uparrow corresponds to an \mathcal{N} -word or a DNA expression ε_i , which has to prefit the $(i+1)^{\text{st}}$ child by upper strands. Then, e.g., $R(\mathcal{S}^+(\varepsilon_i))$ must be an element of $\mathcal{A}_\pm \cup \mathcal{A}_+$.

We can check this condition by walking the rightmost path in the subtree rooted in ε_i . This path ends in a certain \mathcal{N} -word α . If the parent of α is a node labelled by \downarrow or \uparrow , $R(\mathcal{S}^+(\varepsilon_i))$ certainly belongs to $\mathcal{A}_\pm \cup \mathcal{A}_+$. If not (hence, if the parent of α is labelled by \downarrow), the composite symbol $R(\mathcal{S}^+(\varepsilon_i))$ cannot be an element of \mathcal{A}_+ , as the lower part of the symbol is not equal to $-$. In order for $R(\mathcal{S}^+(\varepsilon_i))$ to be in \mathcal{A}_\pm , there has to be a node labelled by \uparrow on the path from ε_i down to α (including ε_i), and this is easy to verify.

If (and only if) a tree satisfies the three requirements mentioned, it represents a DNA expression: *the DNA expression of the tree*.

In order to obtain this DNA expression, we have to perform a depth first search walk through the tree. In this walk, when entering an internal node X for the first time, we collect the opening bracket \langle and the operator of the node. Next, we collect the argument(s) of the operator by recursively visiting the child(ren) of the node, and finally, when returning to X , we obtain the closing bracket \rangle . Apart from this closing bracket, the walk can be considered as a preorder walk.

2.12 Equivalent DNA expressions

Different DNA expressions may correspond to the same DNA molecule. It is, for instance, easy to verify that the DNA expressions $\langle \uparrow \alpha \rangle$ and $\langle \uparrow \langle \uparrow \alpha \rangle \rangle$ have the same semantics. It is also possible that different DNA expressions denote ‘almost the same’ DNA molecule for a certain interpretation of ‘almost the same’. To express these things, we give a number of definitions. Before that, however, we recall some general notions.

A *binary relation* R on a set X is a subset of $X \times X = \{(x, y) \mid x, y \in X\}$. If $(x, y) \in R$, we also write xRy ; if $(x, y) \notin R$, we may write $x\not R y$. A binary relation R on X is

- *reflexive* if for every $x \in X$, xRx
- *symmetric* if for every $x, y \in X$, xRy implies yRx
- *antisymmetric* if for every $x, y \in X$, $(xRy$ and $yRx)$ implies $x = y$
- *transitive* if for every $x, y, z \in X$, $(xRy$ and $yRz)$ implies xRz

If a relation R is reflexive, symmetric and transitive, R is called an *equivalence relation*; if R is reflexive, antisymmetric and transitive, we call R a *partial order*.

Given a binary relation R , the set $R^\partial = \{(y, x) \mid (x, y) \in R\}$ is the *dual relation* of R . A binary relation R_1 is a *refinement* of a binary relation R_2 if $R_1 \subseteq R_2$, in other words: if xR_1y implies xR_2y . In this case R_2 is called an *extension* of R_1 .

We return to the world of DNA. We define four binary relations on \mathcal{D} .

Definition 2.17 Two DNA expressions E_1 and E_2 are strictly equivalent, or equivalent for short, if $\mathcal{S}(E_1) = \mathcal{S}(E_2)$. We write $E_1 \equiv E_2$ then.

Hence two DNA expressions are equivalent if they denote exactly the same DNA molecule.

A somewhat weaker version of this relation is

Definition 2.18 Two DNA expressions E_1 and E_2 are equivalent modulo nicks, if $\nu(\mathcal{S}(E_1)) = \nu(\mathcal{S}(E_2))$. We write $E_1 \overline{\equiv} E_2$ then.

Intuitively, E_1 and E_2 are equivalent modulo nicks, if they denote DNA molecules with the same nucleotides at the same positions; the DNA molecules may, however, have nicks at different positions. E_1 may have nicks not occurring in E_2 and/or the other way round. For instance, if $\mathcal{S}(E_1) = \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \nabla \begin{pmatrix} \text{C} \\ \text{G} \end{pmatrix} \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \triangle \begin{pmatrix} \text{T} \\ \text{A} \end{pmatrix} \begin{pmatrix} \text{G} \\ \text{C} \end{pmatrix}$, and $\mathcal{S}(E_2) = \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \begin{pmatrix} \text{C} \\ \text{G} \end{pmatrix} \triangle \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \begin{pmatrix} \text{T} \\ \text{A} \end{pmatrix} \begin{pmatrix} \text{G} \\ \text{C} \end{pmatrix}$ then $E_1 \overline{\equiv} E_2$ ²

We further define a variant of this last relation.

Definition 2.19 A DNA expression E_1 is equivalent to a DNA expression E_2 pre-modulo nicks, if there are strings X_1, \dots, X_r with $r \geq 1$ over $\mathcal{A}_{\nabla\triangle}$ and symbols $c_1, \dots, c_{r-1} \in \{\nabla, \triangle\}$ such that $\mathcal{S}(E_1) = X_1 c_1 \dots c_{r-1} X_r$ and $\mathcal{S}(E_2) = X_1 \dots X_r$. We write $E_1 \nabla \equiv E_2$ then.

Hence, if $E_1 \nabla \equiv E_2$, then $E_1 \overline{\equiv} E_2$ with the restriction that the DNA molecule denoted by E_2 does not contain nicks not occurring in the DNA molecule denoted by E_1 . For instance: if $\mathcal{S}(E_1) = \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \nabla \begin{pmatrix} \text{C} \\ \text{G} \end{pmatrix} \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \triangle \begin{pmatrix} \text{T} \\ \text{A} \end{pmatrix} \begin{pmatrix} \text{G} \\ \text{C} \end{pmatrix}$, and $\mathcal{S}(E_2) = \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \begin{pmatrix} \text{C} \\ \text{G} \end{pmatrix} \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \triangle \begin{pmatrix} \text{T} \\ \text{A} \end{pmatrix} \begin{pmatrix} \text{G} \\ \text{C} \end{pmatrix}$ then $E_1 \nabla \equiv E_2$. On the other hand, if $\mathcal{S}(E_1)$ is as before and $\mathcal{S}(E_2) = \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \begin{pmatrix} \text{C} \\ \text{G} \end{pmatrix} \triangle \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \begin{pmatrix} \text{T} \\ \text{A} \end{pmatrix} \begin{pmatrix} \text{G} \\ \text{C} \end{pmatrix}$, then $E_1 \nabla \not\equiv E_2$.

If $E_1 \nabla \equiv E_2$, we may also write $E_2 \equiv_{\nabla} E_1$ and say that E_2 is *equivalent post-modulo nicks* to E_1 . Thus, the relations $\nabla \equiv$ and \equiv_{∇} are each other's duals: $\nabla \equiv = \equiv_{\nabla}^{\partial}$.

It is easy to verify that each of the binary relations \equiv , $\overline{\equiv}$, $\nabla \equiv$ and \equiv_{∇} is reflexive and transitive. Further, \equiv and $\overline{\equiv}$ are symmetric, so these relations are (indeed) equivalence relations.

The relations $\nabla \equiv$ and \equiv_{∇} are not symmetric. Hence, despite of their names, they are no equivalence relations. At first glance, one might think that $\nabla \equiv$ and \equiv_{∇} are antisymmetric: 'if a formal DNA molecule $\mathcal{S}(E_1)$ has more nicks than another formal DNA molecule $\mathcal{S}(E_2)$, then certainly $\mathcal{S}(E_2)$ does not have more nicks than $\mathcal{S}(E_1)$ '. However, if $E_1 \equiv E_2$, then both $E_1 \nabla \equiv E_2$ and $E_2 \nabla \equiv E_1$ (and analogously for \equiv_{∇}). Since equivalent DNA expressions E_1 and E_2 are not necessarily the same, $\nabla \equiv$ and \equiv_{∇} are not antisymmetric. Consequently, they are no partial orders, either. At most we can say that they are antisymmetric (and thus partial orders) *up to equivalence*.

It follows immediately from the definition that $E_1 \nabla \equiv E_2$ implies $E_1 \overline{\equiv} E_2$ and that $E_1 \equiv_{\nabla} E_2$ implies $E_1 \overline{\equiv} E_2$, so $\nabla \equiv$ and \equiv_{∇} are refinements of $\overline{\equiv}$. On the other hand, the equivalence relation \equiv is a refinement both of $\nabla \equiv$ and of \equiv_{∇} (and thus certainly of $\overline{\equiv}$). Finally, the equivalence relation $=$ on \mathcal{D} (containing only (x, x) for $x \in \mathcal{D}$), is a trivial refinement of \equiv .

²Actually, this example is not really appropriate. As we will see in § 3.1, a DNA expression with the semantics attributed to E_1 does not exist. At the level of formal DNA molecules, however, this example is a good illustration of the notion of equivalence modulo nicks.

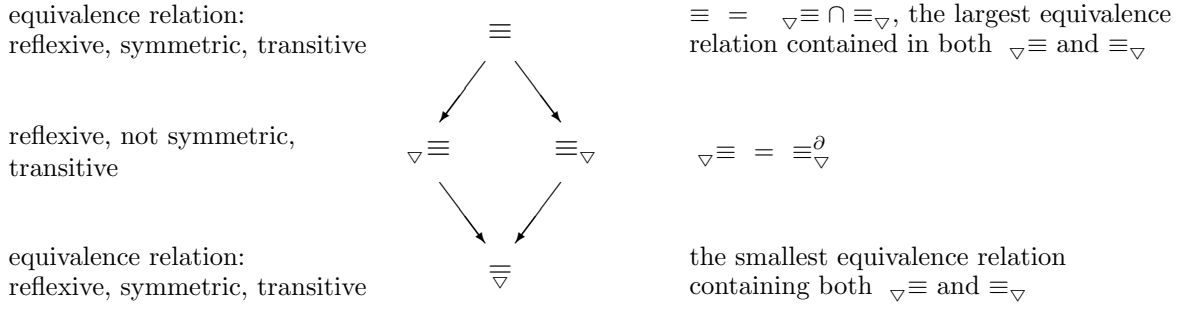


Figure 2.10: Properties of and relations between four different binary relations on \mathcal{D} .

We can combine the notions of transitivity and refinement, for instance: if $E_1 \equiv E_2$ and $E_2 \equiv \nabla E_3$, then $E_1 \equiv \nabla E_3$. The following is also clear: if $E_1 \nabla \equiv E_2$ and $E_1 \equiv \nabla E_2$ then $E_1 \equiv E_2$. Thus, the equivalence relation \equiv is the intersection of the relations $\nabla \equiv$ and $\equiv \nabla$. Clearly, \equiv is the largest equivalence relation contained in both $\nabla \equiv$ and $\equiv \nabla$.

We finally have the following result:

Lemma 2.20 *The relation $\overline{\nabla}$ is the smallest equivalence relation containing both $\nabla \equiv$ and $\equiv \nabla$.*

Note that there is indeed a unique *smallest* equivalence relation containing both $\nabla \equiv$ and $\equiv \nabla$, namely the intersection of all equivalence relations containing $\nabla \equiv$ and $\equiv \nabla$.

Proof: Let R_0 be the smallest equivalence relation containing both $\nabla \equiv$ and $\equiv \nabla$. We just observed that $\overline{\nabla}$ is an equivalence relation containing $\nabla \equiv$ and $\equiv \nabla$. Hence, R_0 must be a subset of $\overline{\nabla}$.

Consider two arbitrary DNA expressions E_1 and E_2 such that $E_1 \overline{\nabla} E_2$. By definition, $\nu(\mathcal{S}(E_1)) = \nu(\mathcal{S}(E_2))$, or, in words, E_1 and E_2 denote DNA molecules that have the same nucleotides at the same positions, but may have different nicks.

If we let $E_3 = \langle \downarrow \langle \uparrow E_1 \rangle \rangle$, then $E_3 \in \mathcal{D}$ and $\mathcal{S}(E_3) = \nu^-(\nu^+(\mathcal{S}(E_1))) = \nu(\mathcal{S}(E_1))$ by (2.12). Thus, $\mathcal{S}(E_3) = \nu(\mathcal{S}(E_1)) = \nu(\mathcal{S}(E_2))$, or, in words, E_3 denotes a DNA molecule with the same nucleotides at the same positions as E_1 and E_2 , but without nicks.

We have $E_1 \nabla \equiv E_3$ and $E_3 \equiv \nabla E_2$, implying $E_1 R_0 E_3$ and $E_3 R_0 E_2$. The transitivity of R_0 yields that also $E_1 R_0 E_2$. Because E_1 and E_2 were arbitrary DNA expressions with $E_1 \overline{\nabla} E_2$, the equivalence relation $\overline{\nabla}$ must be a subset of R_0 .

Thus, we can conclude that R_0 is equal to $\overline{\nabla}$. □

The results of this section are summarized in Figure 2.10.

Note that, because $\nabla \equiv$ and $\equiv \nabla$ are each other's duals and symmetry is an inherent property of equivalence relations, every equivalence relation contained in $\nabla \equiv$ is also contained in $\equiv \nabla$, and vice versa. Similarly, every equivalence relation containing $\nabla \equiv$ also contains $\equiv \nabla$, and vice versa. Therefore, we may rephrase two statements we made, as follows: (1) the relation \equiv is the largest equivalence relation contained in $\nabla \equiv$, and (2) the relation $\overline{\nabla}$ is the smallest equivalence relation containing $\nabla \equiv$. Of course, in the rephrased statements, we may replace the relation $\nabla \equiv$ by $\equiv \nabla$.

Chapter 3

Basic results on DNA expressions

In this chapter, we present some basic results on DNA expressions, which will be used in the next chapter of this report. We first discuss which formal DNA molecules can be denoted by a DNA expression. After that, we derive a number of results on equivalence (modulo nicks) between different DNA expressions.

3.1 Expressible formal DNA molecules

Many formal DNA molecules can be denoted by DNA expressions. We call such formal DNA molecules *expressible*. In particular, there exist DNA expressions which denote molecules with gaps and nicks. An example of this was DNA expression (2.17), which denotes the molecule of Figure 2.5(b), with two gaps and a nick.

Unfortunately, there also exist formal DNA molecules that are not expressible. We will see that the presence of nick letters in a formal DNA molecule determines whether or not it is expressible. We have a number of results concerning nicks in DNA expressions.

Lemma 3.1 *Let E be a \uparrow -expression, $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$, where $n \geq 1$ and ε_i is an \mathcal{N} -word or a DNA expression for $i = 1, \dots, n$. Then*

1. *the upper strand of E is nick free;*
2. *the lower strand of E is nick free if and only if*
 - (a) *for $i = 1, \dots, n$, the lower strand of $\mathcal{S}^+(\varepsilon_i)$ is nick free, and*
 - (b) *for $i = 1, \dots, n - 1$, either $R(\mathcal{S}^+(\varepsilon_i)) \in \mathcal{A}_+$ or $L(\mathcal{S}^+(\varepsilon_{i+1})) \in \mathcal{A}_+$ (or both).*

Proof: By definition,

$$\mathcal{S}(E) = \nu^+(\mathcal{S}^+(\varepsilon_1))y_1 \dots y_{n-1}\nu^+(\mathcal{S}^+(\varepsilon_n))$$

with the y_i 's from (2.15).

1. Because the function ν^+ removes all upper nick letters from its arguments, and y_i is either \triangle or λ (in particular, y_i is not an upper nick letter), the upper strand of $\mathcal{S}(E)$ is nick free.

2. “ \implies ” Assume that the lower strand of E is nick free.

Suppose then that condition 2(a) is not valid. Hence, for some $i \in \{1, \dots, n\}$, $\mathcal{S}^+(\varepsilon_i)$ contains a lower nick letter. Then also $\nu^+(\mathcal{S}^+(\varepsilon_i))$ and thus $\mathcal{S}(E)$ contains a lower nick letter, which contradicts the assumption.

Suppose that condition 2(b) is not valid. Hence, for some $i \in \{1, \dots, n-1\}$, both $R(\mathcal{S}^+(\varepsilon_i)) \in \mathcal{A}_\pm$ and $L(\mathcal{S}^+(\varepsilon_{i+1})) \in \mathcal{A}_\pm$. By definition, $y_i = \triangle$ and the lower strand of E is not nick free then, which again contradicts the assumption.

“ \impliedby ” Assume that conditions 2(a) and 2(b) hold for the arguments of E . Because $\mathcal{S}^+(\varepsilon_i)$ does not contain lower nick letters by condition 2(a), and the function ν^+ certainly does not introduce lower nick letters, the lower strand of $\nu^+(\mathcal{S}^+(\varepsilon_i))$ is nick free for $i = 1, \dots, n$.

Further, condition 2(b) ensures that $y_i = \lambda$ for $i = 1, \dots, n-1$.

As a result, the lower strand of $\mathcal{S}(E)$ is nick free.

□

In an analogous way we prove

Lemma 3.2 *Let E be a \downarrow -expression, $E = \langle \downarrow \varepsilon_1 \dots \varepsilon_n \rangle$, where $n \geq 1$ and ε_i is an \mathcal{N} -word or a DNA expression for $i = 1, \dots, n$. Then*

1. *the lower strand of E is nick free;*
2. *the upper strand of E is nick free if and only if*
 - (a) *for $i = 1, \dots, n$, the upper strand of $\mathcal{S}^-(\varepsilon_i)$ is nick free, and*
 - (b) *for $i = 1, \dots, n-1$, either $R(\mathcal{S}^-(\varepsilon_i)) \in \mathcal{A}_-$ or $L(\mathcal{S}^-(\varepsilon_{i+1})) \in \mathcal{A}_-$ (or both).*

We finally have

Lemma 3.3 *Let E be a \uparrow -expression, $E = \langle \uparrow \varepsilon_1 \rangle$, where ε_1 is an \mathcal{N} -word or a DNA expression. Then*

1. *the upper strand of E is nick free if and only if either ε_1 is an \mathcal{N} -word α or ε_1 is a DNA expression with a nick free upper strand;*
2. *the lower strand of E is nick free if and only if either ε_1 is an \mathcal{N} -word α or ε_1 is a DNA expression with a nick free lower strand.*

Proof: If ε_1 is an \mathcal{N} -word α , then $\mathcal{S}(E) = \binom{\alpha}{c(\alpha)}$ and E is altogether nick free.

If, on the other hand ε_1 is a DNA expression E_1 , then $\mathcal{S}(E) = \kappa(\mathcal{S}(E_1))$. As the function κ does not introduce and does not repair nicks, the upper (or lower) strand of E is nick free if and only if the upper (lower, respectively) strand of E_1 is nick free. □

Lemmas 3.1 and 3.2 are useful for proving the following result:

Theorem 3.4 *There do not exist DNA expressions denoting molecules with nicks in both strands. In other words: if E is a DNA expression, then either the upper strand or the lower strand (or both) of E is nick free.*

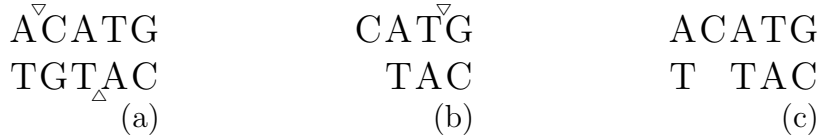


Figure 3.1: (a) A molecule which cannot be denoted by a DNA expression, because it has nicks in both strands; (b) a molecule which can be denoted by a DNA expression, but not by a DNA expression in \mathcal{D}' , because it contains a nick; (c) a molecule which can be denoted by a DNA expression in \mathcal{D}' , because it is nick free.

Proof: By Lemmas 3.1 and 3.2, every \uparrow -expression and every \downarrow -expression has at least one strand without nicks.

For \downarrow -expressions $E = \langle \downarrow \varepsilon_1 \rangle$, with ε_1 an \mathcal{N} -word or a DNA expression, we prove the claim by induction on the number p of operators occurring in E .

- If $p = 1$, then $E = \langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α and $\mathcal{S}(E) = \binom{\alpha}{c(\alpha)}$. Clearly, E is altogether nick free.
- Let $p \geq 1$, and suppose that the claim holds for all \downarrow -expressions containing p operators (induction hypothesis). Then consider an arbitrary \downarrow -expression $E = \langle \downarrow \varepsilon_1 \rangle$ with $p + 1$ operators.

As $p + 1 \geq 2$, ε_1 must be a DNA expression E_1 containing p operators. If E_1 is a \uparrow -expression or a \downarrow -expression, at least one of the strands of E_1 is nick free by Lemmas 3.1 and 3.2. If E_1 is a \downarrow -expression, we know by the induction hypothesis that at least one of the strands of E_1 is nick free.

Because $\mathcal{S}(E) = \kappa(\mathcal{S}(E_1))$ and the function κ does not introduce (nor repair) nicks in its argument, the claim holds also for E .

□

Consequently, there is, e.g., no DNA expression for the molecule depicted in Figure 3.1(a).

Given Theorem 3.4, we may wonder if there are other limitations on the DNA molecules with gaps and nicks that can be expressed in \mathcal{D} . Does there exist a DNA expression for every DNA molecule with nicks in at most one strand? In Chapter 4, we will see that indeed there is. In particular, in Theorem 4.65 and Theorem 4.67, we describe constructions of DNA expressions denoting nick free formal DNA molecules and formal DNA molecules containing lower nick letters, respectively. We thus have

Theorem 3.5 *A formal DNA molecule X is expressible, if and only if X does not contain both upper nick letters and lower nick letters.*

Hence, some DNA molecules with nicks are expressible, whereas others are not.

3.2 Some equivalences

There are many general rules concerning equivalence between different DNA expressions. Some of them follow immediately from the definition of the semantics of a DNA expression. For example, for every \mathcal{N} -word α ,

$$\langle \downarrow \alpha \rangle \equiv \langle \downarrow \langle \uparrow \alpha \rangle \rangle \equiv \langle \downarrow \langle \downarrow c(\alpha) \rangle \rangle. \quad (3.1)$$

Another example is: for every DNA expression $\langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$, where $n \geq 1$ and for $i = 1, \dots, n$, ε_i is an \mathcal{N} -word or a DNA expression,

$$\langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle \equiv \langle \uparrow E_1 E_2 \dots E_n \rangle, \quad (3.2)$$

where for $i = 1, \dots, n$, $E_i = \text{Exp}^+(\varepsilon_i)$.

Other rules are intuitively clear, but a bit less easy to prove. To demonstrate how such rules are proved, we state one rule as a lemma here and give its formal proof.

Lemma 3.6 *Let $1 \leq i_0 \leq j_0 \leq n$, and let ε_i for $i = 1, \dots, n$ be an \mathcal{N} -word or a DNA expression. Then*

$$\langle \uparrow \varepsilon_1 \dots \varepsilon_{i_0-1} \langle \uparrow \varepsilon_{i_0} \dots \varepsilon_{j_0} \rangle \varepsilon_{j_0+1} \dots \varepsilon_n \rangle \equiv \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle \quad (3.3)$$

if either the left-hand side or the right-hand side of the equivalence is a DNA expression.

Proof: Let us (again) consider $E_i = \text{Exp}^+(\varepsilon_i)$ for $i = 1, \dots, n$ and let $E_{i_0 j_0} = \langle \uparrow \varepsilon_{i_0} \dots \varepsilon_{j_0} \rangle$.

First, we need to prove that if either side of the equivalence in the claim is a DNA expression, then so is the other. If, e.g., the left-hand side is a DNA expression, then in particular $E_{i_0 j_0} = \langle \uparrow \varepsilon_{i_0} \dots \varepsilon_{j_0} \rangle$ is a DNA expression. This implies that $E_i \sqsubseteq E_{i+1}$ ($i = i_0, \dots, j_0 - 1$). We further know that $E_i \sqsubseteq E_{i+1}$ for $i = 1, \dots, i_0 - 2, j_0 + 1, \dots, n - 1$. Finally, we have $E_{i_0-1} \sqsubseteq E_{i_0 j_0}$ and $E_{i_0 j_0} \sqsubseteq E_{j_0+1}$.

The last two relations are equivalent to $R(E_{i_0-1}), L(E_{i_0 j_0}) \in \mathcal{A}_\pm \cup \mathcal{A}_+$ and to $R(E_{i_0 j_0}), L(E_{j_0+1}) \in \mathcal{A}_\pm \cup \mathcal{A}_+$, respectively. Now, by Lemma 2.16(3), $L(E_{i_0 j_0}) = L(E_{i_0})$ and $R(E_{i_0 j_0}) = R(E_{j_0})$. Hence, $L(E_{i_0}), R(E_{j_0}) \in \mathcal{A}_\pm \cup \mathcal{A}_+$. We already knew that $R(E_{i_0-1}), L(E_{j_0+1}) \in \mathcal{A}_\pm \cup \mathcal{A}_+$. Thus, $E_{i_0-1} \sqsubseteq E_{i_0}$ and $E_{j_0} \sqsubseteq E_{j_0+1}$.

We can conclude that $E_i \sqsubseteq E_{i+1}$ for $i = 1, 2, \dots, n - 1$, so that $\langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$ is a DNA expression. The proof in the other direction proceeds along the same lines.

Now, we can concentrate on the claim itself. By definition,

$$\mathcal{S}^+(E_{i_0 j_0}) = \mathcal{S}(E_{i_0 j_0}) = \nu^+(\mathcal{S}^+(\varepsilon_{i_0})) y_{i_0} \dots y_{j_0-1} \nu^+(\mathcal{S}^+(\varepsilon_{j_0}))$$

and

$$\begin{aligned} \mathcal{S}(\langle \uparrow \varepsilon_1 \dots \varepsilon_{i_0-1} \langle \uparrow \varepsilon_{i_0} \dots \varepsilon_{j_0} \rangle \varepsilon_{j_0+1} \dots \varepsilon_n \rangle) = \\ \nu^+(\mathcal{S}^+(\varepsilon_1)) y_1 \dots y_{i_0-2} \nu^+(\mathcal{S}^+(\varepsilon_{i_0-1})) y_{i_0-1} \cdot \nu^+(\mathcal{S}^+(E_{i_0 j_0})) \cdot \\ y_{j_0} \nu^+(\mathcal{S}^+(\varepsilon_{j_0+1})) y_{j_0+1} \dots y_{n-1} \nu^+(\mathcal{S}^+(\varepsilon_n)), \end{aligned} \quad (3.4)$$

where the y_i 's are defined by

$$y_i = \begin{cases} \triangle & \text{if } E_i \sqsubseteq E_{i+1}, \text{ i.e., if both } R(E_i) \in \mathcal{A}_\pm \\ & \text{and } L(E_{i+1}) \in \mathcal{A}_\pm \\ \lambda & \text{otherwise, i.e., if } R(E_i) \in \mathcal{A}_+ \\ & \text{or } L(E_{i+1}) \in \mathcal{A}_+ \text{ (or both)} \end{cases}. \quad (3.5)$$

for $i = 1, \dots, i_0 - 2, i_0, \dots, j_0 - 1, j_0 + 1, \dots, n - 1$,

$$y_{i_0-1} = \begin{cases} \triangle & \text{if } E_{i_0-1} \sqsubseteq E_{i_0 j_0}, \text{ i.e., if both } R(E_{i_0-1}) \in \mathcal{A}_\pm \\ & \text{and } L(E_{i_0 j_0}) \in \mathcal{A}_\pm \\ \lambda & \text{otherwise, i.e., if } R(E_{i_0-1}) \in \mathcal{A}_+ \\ & \text{or } L(E_{i_0 j_0}) \in \mathcal{A}_+ \text{ (or both)} \end{cases}$$

$$y_{j_0} = \begin{cases} \triangle & \text{if } E_{i_0 j_0} \sqsubseteq E_{j_0+1}, \text{ i.e., if both } R(E_{i_0 j_0}) \in \mathcal{A}_\pm \\ & \text{and } L(E_{j_0+1}) \in \mathcal{A}_\pm \\ \lambda & \text{otherwise, i.e., if } R(E_{i_0 j_0}) \in \mathcal{A}_+ \\ & \text{or } L(E_{j_0+1}) \in \mathcal{A}_+ \text{ (or both)} \end{cases}$$

We already observed that $L(E_{i_0 j_0}) = L(E_{i_0})$ and $R(E_{i_0 j_0}) = R(E_{j_0})$. But then the definitions of y_{i_0-1} and y_{j_0} fit exactly into the general framework of definition (3.5). Hence, definition (3.5) is valid for $i = 1, \dots, n-1$.

Now we will elaborate on the middle term of the right-hand side of (3.4). Because ν^+ is a homomorphism,

$$\begin{aligned} \nu^+(\mathcal{S}^+(E_{i_0 j_0})) &= \nu^+(\nu^+(\mathcal{S}^+(\varepsilon_{i_0}))y_{i_0} \dots y_{j_0-1}\nu^+(\mathcal{S}^+(\varepsilon_{j_0}))) = \\ & \nu^+(\nu^+(\mathcal{S}^+(\varepsilon_{i_0}))\nu^+(y_{i_0}) \dots \nu^+(y_{j_0-1})\nu^+(\nu^+(\mathcal{S}^+(\varepsilon_{j_0})))) \end{aligned} \quad (3.6)$$

For every i , y_i is either \triangle or λ . Consequently, $\nu^+(y_i) = y_i$ for every i , and in particular for $i = i_0, \dots, j_0-1$. Combining this with property (2.11), we can rewrite the result of (3.6) into

$$\nu^+(\mathcal{S}^+(\varepsilon_{i_0}))y_{i_0} \dots y_{j_0-1}\nu^+(\mathcal{S}^+(\varepsilon_{j_0}))$$

We can substitute this into (3.4), which yields

$$\begin{aligned} \mathcal{S}(\langle \uparrow \varepsilon_1 \dots \varepsilon_{i_0-1} \langle \uparrow \varepsilon_{i_0} \dots \varepsilon_{j_0} \rangle \varepsilon_{j_0+1} \dots \varepsilon_n \rangle) &= \\ \nu^+(\mathcal{S}^+(\varepsilon_1))y_1 \dots y_{i_0-2}\nu^+(\mathcal{S}^+(\varepsilon_{i_0-1}))y_{i_0-1} & \nu^+(\mathcal{S}^+(\varepsilon_{i_0}))y_{i_0} \dots y_{j_0-1}\nu^+(\mathcal{S}^+(\varepsilon_{j_0})) \\ y_{j_0}\nu^+(\mathcal{S}^+(\varepsilon_{j_0+1}))y_{j_0+1} \dots y_{n-1}\nu^+(\mathcal{S}^+(\varepsilon_n)) & \end{aligned}$$

with y_i as in (3.5) for $i = 1, \dots, n-1$. But this exactly equals $\mathcal{S}(\langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle)$. \square

In fact, (3.2) is a special case of Lemma 3.6. Another special case is

$$\langle \uparrow \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle \rangle \equiv \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle \quad (3.7)$$

if either side of the equivalence is a DNA expression. Under the same condition, we find

$$\langle \uparrow \langle \uparrow \varepsilon_1 \rangle \langle \uparrow \varepsilon_2 \rangle \rangle \equiv \langle \uparrow \varepsilon_1 \varepsilon_2 \rangle \quad (3.8)$$

by applying the lemma twice.

For every result on \uparrow -expressions there exists an analogous result for \downarrow -expressions (and vice versa). For example, the analogous version of Lemma 3.6 is

Let $1 \leq i_0 \leq j_0 \leq n$, and let ε_i for $i = 1, \dots, n$ be an \mathcal{N} -word or a DNA expression. Then

$$\langle \downarrow \varepsilon_1 \dots \varepsilon_{i_0-1} \langle \downarrow \varepsilon_{i_0} \dots \varepsilon_{j_0} \rangle \varepsilon_{j_0+1} \dots \varepsilon_n \rangle \equiv \langle \downarrow \varepsilon_1 \dots \varepsilon_n \rangle$$

if either the left-hand side or the right-hand side of the equivalence is a DNA expression.

Often, we will not formulate the analogous result explicitly. As a matter of fact, we will refer to the result for \uparrow -expressions, even if we need the analogous version of it.

The analogue of (3.7) for \uparrow -expressions is clear from the definition of the operator \uparrow (see Definition 2.8) and from property (2.11):

$$\langle \uparrow \langle \uparrow \varepsilon \rangle \rangle \equiv \langle \uparrow \varepsilon \rangle \quad (3.9)$$

for every \mathcal{N} -word or DNA expression ε .

We proceed with three results concerning the substitution of (occurrences of) \mathcal{N} -words or DNA subexpressions in a DNA expression by \mathcal{N} -words or DNA subexpressions which are equivalent ((pre/post-)modulo nicks).

Lemma 3.7 *Let E be a DNA expression and let E^s be (an occurrence of) a DNA subexpression in E . Let $E^{s'}$ be a DNA expression such that $E^s \equiv E^{s'}$.*

When we substitute (the occurrence of) E^s in E by $E^{s'}$, the resulting string E' is again a DNA expression, and $E \equiv E'$.

Proof: By induction on the number p of operators in E which are not in E^s .

- If $p = 0$, then $E = E^s$, and the claim is trivially valid.
- Let $p \geq 0$, and suppose that the claim holds for every DNA expression E and (occurrence of a) DNA subexpression E^s of E such that the number of operators in E which are not in E^s is at most p (induction hypothesis). Now let E be a DNA expression and let E^s be (an occurrence of) a DNA subexpression of E such that there are $p + 1$ operators in E which are not in E^s .

Because $p + 1 \geq 1$, E^s is a proper DNA subexpression of E , and E^s is the immediate argument of a DNA subexpression $E^\sigma = \langle |_0 \varepsilon_1 \dots \varepsilon_{i_0-1} E^s \varepsilon_{i_0+1} \dots \varepsilon_n \rangle$ of E , for some operator $|_0$, i_0 and n with $1 \leq i_0 \leq n$, and \mathcal{N} -words or DNA expressions ε_i .

Let us define $E^{\sigma'} = \langle |_0 \varepsilon_1 \dots \varepsilon_{i_0-1} E^{s'} \varepsilon_{i_0+1} \dots \varepsilon_n \rangle$. Then by condition 2 of Definition 2.1, $L(E^s)$, $R(E^s)$, $L(E^{s'})$ and $R(E^{s'})$ are not nick letters, and thus $L(E^s) = L(E^{s'})$ and $R(E^s) = R(E^{s'})$. Consequently, the arguments of $E^{\sigma'}$ fit together just like those of E^σ , so that $E^{\sigma'}$ is a DNA expression. Now it follows from the definition of the semantics of a DNA expression that $E^\sigma \equiv E^{\sigma'}$.

Substituting E^s in E by $E^{s'}$ produces the same overall string E' as substituting E^σ by $E^{\sigma'}$. Because the number of operators in E which are not in E^σ is at most p , it follows by induction that E' is a DNA expression satisfying $E \equiv E'$.

□

It is easy to see that this result remains valid if we replace every occurrence of the relation \equiv by \equiv , \equiv_∇ or \equiv_{∇} .

Lemma 3.8 *Let E be a DNA expression and let ε be (an occurrence of) an \mathcal{N} -word or a proper DNA subexpression in E , such that the parent operator of ε is \uparrow . Let ε' be an \mathcal{N} -word or a DNA expression satisfying $\text{Exp}^+(\varepsilon) \equiv \text{Exp}^+(\varepsilon')$.*

When we substitute (the occurrence of) ε in E by ε' , the resulting string E' is again a DNA expression, and $E \equiv E'$.

Proof: If both ε and ε' are DNA expressions, then we simply have a special case of Lemma 3.7.

If both ε and ε' are \mathcal{N} -words, then they must be equal, because $\text{Exp}^+(\varepsilon) = \left(\begin{smallmatrix} \varepsilon \\ - \end{smallmatrix}\right)$ and $\text{Exp}^+(\varepsilon') = \left(\begin{smallmatrix} \varepsilon' \\ - \end{smallmatrix}\right)$ are assumed to be equivalent modulo nicks. Then also $E = E'$ and the claim follows immediately.

If ε is an \mathcal{N} -word and ε' is a DNA expression, then let E^s be the DNA subexpression of E which ε is an immediate argument of: $E^s = \langle \uparrow \varepsilon_1 \dots \varepsilon_{i_0-1} \varepsilon \varepsilon_{i_0+1} \dots \varepsilon_n \rangle$ for some i_0 and n with $1 \leq i_0 \leq n$ and \mathcal{N} -words or DNA expressions ε_i . Now, by Lemma 3.6, $E^s \equiv \langle \downarrow \varepsilon_1 \dots \varepsilon_{i_0-1} \langle \uparrow \varepsilon \rangle \varepsilon_{i_0+1} \dots \varepsilon_n \rangle$. Let us use $E^{s'}$ to denote the right-hand side of this equivalence.

By Lemma 3.7, we can replace E^s in E by $E^{s'}$ and the overall result E'' is a DNA expression equivalent to E . In E'' we can replace $\langle \uparrow \varepsilon \rangle$ by the DNA expression ε' , and again by Lemma 3.7, the resulting overall string E' is a DNA expression satisfying $E'' \equiv E'$. By the transitivity of the relation \equiv , we also have $E \equiv E'$.

For the case that ε is a DNA expression and ε' is an \mathcal{N} -word, the claim can be proved analogously. \square

When we apply a special case of Lemma 3.8 n times, we obtain

Corollary 3.9 *Let $n \geq 1$, and let for $i = 1, \dots, n$, ε_i and ε'_i be an \mathcal{N} -word or a DNA expression, $E_i = \text{Exp}^+(\varepsilon_i)$ and $E'_i = \text{Exp}^+(\varepsilon'_i)$.*

Then

$$\text{if } E_i \equiv E'_i \text{ for } i = 1, \dots, n, \text{ then } \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle \equiv \langle \uparrow \varepsilon'_1 \dots \varepsilon'_n \rangle$$

if either of $\langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$ and $\langle \uparrow \varepsilon'_1 \dots \varepsilon'_n \rangle$ is a DNA expression, i.e. if, e.g. $\varepsilon_i \equiv \varepsilon_{i+1}$ for $i = 1, \dots, n-1$.

Both in Lemma 3.8 and in Corollary 3.9, we might also replace every occurrence of the relation \equiv by \equiv_{∇} , \equiv_{∇} or \equiv_{∇} , and the operator \uparrow by \downarrow (in which case we must use the function Exp^- instead of Exp^+) or \uparrow (in which case n must be equal to 1 in Corollary 3.9).

Finally, we will give four results that deal with the exchange of outermost operators between a DNA expression and its argument(s). We will need them when we prove the correctness of an algorithm for rewriting DNA expressions. Again, we will state (and prove) only one of two possible versions of each of the results. There exist analogous results in which every occurrence of the operator \uparrow is replaced by \downarrow and (if applicable) vice versa.

Lemma 3.10 *Let $E = \langle \uparrow \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle \rangle$ with $n \geq 1$ be a \uparrow -expression, such that for $i = 1, \dots, n$, ε_i is a DNA expression (and not an \mathcal{N} -word). Then $E \equiv_{\nabla} \langle \uparrow \langle \uparrow \varepsilon_1 \rangle \dots \langle \uparrow \varepsilon_n \rangle \rangle$.*

Note that the right-hand side of the equivalence in the claim is indeed a DNA expression. By Lemma 2.16, case 5, $L(\langle \uparrow \varepsilon_i \rangle), R(\langle \uparrow \varepsilon_i \rangle) \in \mathcal{A}_{\pm}$ for $i = 1, \dots, n$, and thus the arguments of the operator \uparrow in the right-hand side fit together by upper strands.

Proof: By definition,

$$\begin{aligned} \mathcal{S}(E) &= \mathcal{S}(\langle \uparrow \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle \rangle) = \kappa(\mathcal{S}(\langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle)) = \\ &= \kappa(\nu^+(\mathcal{S}^+(\varepsilon_1))y_1 \dots y_{n-1}\nu^+(\mathcal{S}^+(\varepsilon_n))) = \kappa(\nu^+(\mathcal{S}^+(\varepsilon_1))y_1 \dots y_{n-1}\kappa(\nu^+(\mathcal{S}^+(\varepsilon_n))), \end{aligned}$$

where for $i = 1, \dots, n-1$, $y_i \in \{\Delta, \lambda\}$, and the actual value of y_i depends on ε_i and ε_{i+1} (see (2.15)). Because ε_i is a DNA expression, $\mathcal{S}^+(\varepsilon_i) = \mathcal{S}(\varepsilon_i)$ for $i = 1, \dots, n$, and because of the commutativity of κ and ν^+ (see (2.10)), these functions may be interchanged. Hence, we get:

$$\mathcal{S}(E) = \nu^+(\kappa(\mathcal{S}(\varepsilon_1)))y_1 \dots y_{n-1}\nu^+(\kappa(\mathcal{S}(\varepsilon_n))).$$

On the other hand,

$$\begin{aligned} \mathcal{S}(\langle \uparrow \langle \downarrow \varepsilon_1 \rangle \dots \langle \downarrow \varepsilon_n \rangle \rangle) &= \nu^+(\mathcal{S}^+(\langle \downarrow \varepsilon_1 \rangle))y'_1 \dots y'_{n-1}\nu^+(\mathcal{S}^+(\langle \downarrow \varepsilon_n \rangle)) = \\ &\nu^+(\kappa(\mathcal{S}(\varepsilon_1)))y'_1 \dots y'_{n-1}\nu^+(\kappa(\mathcal{S}(\varepsilon_n))), \end{aligned}$$

where, for $i = 1, \dots, n-1$, $y'_i \in \{\Delta, \lambda\}$, and the value of y'_i is determined by the arguments $\langle \downarrow \varepsilon_i \rangle$ and $\langle \downarrow \varepsilon_{i+1} \rangle$. However, by Lemma 2.16, case 5, $L(\langle \downarrow \varepsilon_i \rangle), R(\langle \downarrow \varepsilon_i \rangle) \in \mathcal{A}_\pm$ for $i = 1, \dots, n$, so that every y'_i is equal to Δ .

Consequently, $E \equiv_{\nabla} \langle \uparrow \langle \downarrow \varepsilon_1 \rangle \dots \langle \downarrow \varepsilon_n \rangle \rangle$. \square

If, for instance, $n = 2$, and $\mathcal{S}(\varepsilon_1) = \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \nabla \begin{pmatrix} \text{C} \\ \text{G} \end{pmatrix}$ and $\mathcal{S}(\varepsilon_2) = \begin{pmatrix} \text{A} \\ - \end{pmatrix} \begin{pmatrix} \text{T} \\ \text{A} \end{pmatrix} \Delta \begin{pmatrix} \text{G} \\ \text{C} \end{pmatrix}$, then $\mathcal{S}(\langle \uparrow \langle \downarrow \varepsilon_1 \varepsilon_2 \rangle \rangle) = \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \begin{pmatrix} \text{C} \\ \text{G} \end{pmatrix} \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \begin{pmatrix} \text{T} \\ \text{A} \end{pmatrix} \Delta \begin{pmatrix} \text{G} \\ \text{C} \end{pmatrix}$, while $\langle \uparrow \langle \downarrow \varepsilon_1 \rangle \langle \downarrow \varepsilon_2 \rangle \rangle = \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \begin{pmatrix} \text{C} \\ \text{G} \end{pmatrix} \Delta \begin{pmatrix} \text{A} \\ \text{T} \end{pmatrix} \begin{pmatrix} \text{T} \\ \text{A} \end{pmatrix} \Delta \begin{pmatrix} \text{G} \\ \text{C} \end{pmatrix}$.

Lemma 3.10 cannot always be reversed. For instance, if we have a DNA expression $\langle \uparrow \langle \downarrow \varepsilon_1 \rangle \dots \langle \downarrow \varepsilon_n \rangle \rangle$, we do not a priori know that $\langle \downarrow \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle \rangle$ is a DNA expression, because the arguments $\varepsilon_1, \dots, \varepsilon_n$ of \uparrow may not fit together by upper strands. Only if they do, we can say that $\langle \uparrow \langle \downarrow \varepsilon_1 \rangle \dots \langle \downarrow \varepsilon_n \rangle \rangle \nabla \equiv \langle \downarrow \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle \rangle$.

Sometimes, however, reversing is possible without worrying about syntactic constraints.

Corollary 3.11 *For all \mathcal{N} -words $\alpha_1, \dots, \alpha_n$ with $n \geq 1$, we have*

$$\langle \uparrow \langle \downarrow \alpha_1 \rangle \dots \langle \downarrow \alpha_n \rangle \rangle \nabla \equiv \langle \downarrow \alpha_1 \dots \alpha_n \rangle.$$

Note that the concatenation of $n \geq 1$ \mathcal{N} -words α_i is itself a (one) \mathcal{N} -word, so that the right-hand side of the claim is indeed a DNA expression.

Proof: We can rewrite $\langle \downarrow \alpha_1 \dots \alpha_n \rangle$ backwards as follows:

$$\begin{aligned} \langle \downarrow \alpha_1 \dots \alpha_n \rangle &\equiv \langle \downarrow \langle \uparrow \alpha_1 \dots \alpha_n \rangle \rangle \equiv \langle \downarrow \langle \uparrow \langle \uparrow \alpha_1 \rangle \dots \langle \uparrow \alpha_n \rangle \rangle \rangle \equiv_{\nabla} \\ &\langle \uparrow \langle \downarrow \langle \uparrow \alpha_1 \rangle \rangle \dots \langle \downarrow \langle \uparrow \alpha_n \rangle \rangle \rangle \equiv \langle \uparrow \langle \downarrow \alpha_1 \rangle \dots \langle \downarrow \alpha_n \rangle \rangle \end{aligned}$$

The first and the last equivalence follow from (3.1), the second one from Lemma 3.6 and the third one from Lemma 3.10. \square

Theorem 3.12 *Let $\varepsilon_1, \dots, \varepsilon_m, \hat{\varepsilon}_2, \dots, \hat{\varepsilon}_n$ with $m \geq 0$ and $n \geq 1$ be DNA expressions or \mathcal{N} -words, and let \hat{E}_1 be a DNA expression, such that*

- $\mathcal{S}^+(\varepsilon_i) \sqsubseteq \mathcal{S}^+(\varepsilon_{i+1})$ for $i = 1, \dots, m-1$,
- $\mathcal{S}^+(\varepsilon_m) \sqsubseteq \mathcal{S}(\hat{E}_1)$,
- $\mathcal{S}(\hat{E}_1) \sqsubseteq \mathcal{S}^-(\hat{\varepsilon}_2)$ and
- $\mathcal{S}^-(\hat{\varepsilon}_i) \sqsubseteq \mathcal{S}^-(\hat{\varepsilon}_{i+1})$ for $i = 2, \dots, n-1$.

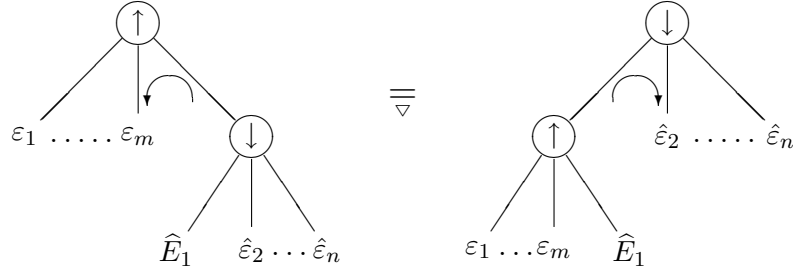


Figure 3.2: Analogue of Theorem 3.12 for trees of DNA expressions.

Then the strings $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_m \langle \downarrow \widehat{E}_1 \widehat{\varepsilon}_2 \dots \widehat{\varepsilon}_n \rangle \rangle$ and $E' = \langle \downarrow \langle \uparrow \varepsilon_1 \dots \varepsilon_m \widehat{E}_1 \rangle \widehat{\varepsilon}_2 \dots \widehat{\varepsilon}_n \rangle$ are DNA expressions satisfying $E \stackrel{\equiv}{=} E'$.

What we actually do in this theorem, is moving the outermost operator \downarrow of the last argument $\langle \downarrow \widehat{E}_1 \widehat{\varepsilon}_2 \dots \widehat{\varepsilon}_n \rangle$ of the DNA expression E to the left of the DNA expression. To ensure that the arguments of the two operators \uparrow and \downarrow still fit together by upper or lower strands, respectively, i.e., that the resulting string is still a DNA expression, we also have to shift one of the closing brackets.

For the tree of the DNA expression E , this action corresponds to a rotation to the left on the root of the tree. If we want to transform the tree of E' back into the tree of E , then we have to perform a rotation to the right on the root of the tree. This is depicted in Figure 3.2.

Note that our requirement that \widehat{E}_1 be a DNA expression (and not an \mathcal{N} -word) is quite natural. If $m \geq 1$ (or $n \geq 2$), it simply *has* to be a DNA expression, in order for E (or E' , respectively) to be a DNA expression. If \widehat{E}_1 were an \mathcal{N} -word α here, the lower strand of $\langle \downarrow \alpha \widehat{\varepsilon}_2 \dots \widehat{\varepsilon}_n \rangle$ would strictly cover the upper strand to the left, and thus ε_m and $\langle \downarrow \alpha \widehat{\varepsilon}_2 \dots \widehat{\varepsilon}_n \rangle$ would not fit together by upper strands in E (and similarly for E' if $n \geq 2$).

Proof: By Definition 2.9 and Lemma 2.16, E and E' are indeed DNA expressions. Now by definition,

$$\begin{aligned} \mathcal{S}(E) &= \nu^+(\mathcal{S}^+(\varepsilon_1))y_1 \dots y_{m-1} \nu^+(\mathcal{S}^+(\varepsilon_m))y_m \cdot \\ &\quad \nu^+(\nu^-(\mathcal{S}(\widehat{E}_1))\widehat{y}_1 \nu^-(\mathcal{S}^-(\widehat{\varepsilon}_2))\widehat{y}_2 \dots \widehat{y}_{n-1} \nu^-(\mathcal{S}^-(\widehat{\varepsilon}_n))) \end{aligned}$$

and

$$\begin{aligned} \mathcal{S}(E') &= \nu^-(\nu^+(\mathcal{S}^+(\varepsilon_1))y_1 \dots y_{m-1} \nu^+(\mathcal{S}^+(\varepsilon_m))y_m \nu^+(\mathcal{S}(\widehat{E}_1))) \cdot \\ &\quad \widehat{y}_1 \nu^-(\mathcal{S}^-(\widehat{\varepsilon}_2))\widehat{y}_2 \dots \widehat{y}_{n-1} \nu^-(\mathcal{S}^-(\widehat{\varepsilon}_n)) \end{aligned}$$

where the y_i 's are either \triangle or λ and the \widehat{y}_i 's are either ∇ or λ (depending on the formal DNA molecules preceding and succeeding them). It is not hard to see that each y_i in $\mathcal{S}(E)$ is equal to the corresponding y_i in $\mathcal{S}(E')$, and that the same property holds for each \widehat{y}_i .

When we observe that $\nu^+(\nabla) = \nu^-(\triangle) = \lambda$ and that for each $X \in \mathcal{A}_{\nabla\triangle}^*$, $\nu^-(\nu^+(X)) = \nu^+(\nu^-(X)) = \nu(X)$, we can rewrite the expressions for $\mathcal{S}(E)$ and $\mathcal{S}(E')$ into:

$$\mathcal{S}(E) = \nu^+(\mathcal{S}^+(\varepsilon_1))y_1 \dots y_{m-1} \nu^+(\mathcal{S}^+(\varepsilon_m))y_m \nu(\mathcal{S}(\widehat{E}_1))\nu(\mathcal{S}^-(\widehat{\varepsilon}_2)) \dots \nu(\mathcal{S}^-(\widehat{\varepsilon}_n))$$

and

$$\mathcal{S}(E') = \nu(\mathcal{S}^+(\varepsilon_1)) \dots \nu(\mathcal{S}^+(\varepsilon_m)) \nu(\mathcal{S}(\hat{E}_1)) \hat{y}_1 \nu^-(\mathcal{S}^-(\hat{\varepsilon}_2)) \hat{y}_2 \dots \hat{y}_{n-1} \nu^-(\mathcal{S}^-(\hat{\varepsilon}_n)).$$

Indeed, $S(E)$ and $S(E')$ can differ only in the occurrences of nicks. Hence, $E \equiv E'$. \square

For a special case we can combine Theorem 3.12 with Corollary 3.11:

Corollary 3.13 *Let $\varepsilon_1, \dots, \varepsilon_m$ with $m \geq 0$ be DNA expressions or \mathcal{N} -words, and $\hat{\alpha}_1$ and $\hat{\alpha}_2$ be \mathcal{N} -words, such that*

- $\mathcal{S}^+(\varepsilon_i) \bar{\square} \mathcal{S}^+(\varepsilon_{i+1})$ for $i = 1, \dots, m-1$ and
- $\mathcal{S}^+(\varepsilon_m) \bar{\square} \mathcal{S}(\langle \downarrow \hat{\alpha}_1 \rangle)$.

Then the strings $E' = \langle \downarrow \langle \uparrow \varepsilon_1 \dots \varepsilon_m \langle \downarrow \hat{\alpha}_1 \rangle \rangle \langle \downarrow \hat{\alpha}_2 \rangle \rangle$ and $E'' = \langle \uparrow \varepsilon_1 \dots \varepsilon_m \langle \downarrow \hat{\alpha}_1 \hat{\alpha}_2 \rangle \rangle$ are DNA expressions satisfying $E' \equiv E''$.

Proof: By Theorem 3.12, E' and $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_m \langle \downarrow \langle \downarrow \hat{\alpha}_1 \rangle \langle \downarrow \hat{\alpha}_2 \rangle \rangle \rangle$ are DNA expressions for which $E' \equiv E$. By Corollary 3.11, the DNA subexpression $E^s = \langle \downarrow \langle \downarrow \hat{\alpha}_1 \rangle \langle \downarrow \hat{\alpha}_2 \rangle \rangle$ of E satisfies $E^s \sphericalangle \equiv \langle \downarrow \hat{\alpha}_1 \hat{\alpha}_2 \rangle$. Consequently, by Lemma 3.7, also E'' is a DNA expression and $E' \equiv E''$. \square

Chapter 4

The length of a DNA expression

The complexity of an algorithm can often be considered as a function of the length of its input. Hence, when we want to analyse the complexity of algorithms that operate on DNA expressions, it is important to know the length of the DNA expressions at hand. Apart from this application, it is also *intrinsically* interesting how long a DNA expression denoting a certain formal DNA molecule may be.

In this chapter, therefore, we examine the length of a DNA expression. We start with a basic observation.

Lemma 4.1 *Let E be a DNA expression denoting a formal DNA molecule X , and let p be the number of operators occurring in E . Then*

$$|E| = 3 \cdot p + |\nu(X)|.$$

As a DNA expression consists of operators and corresponding brackets on the one hand, and \mathcal{N} -letters on the other hand, the term $|\nu(X)|$ apparently counts the number of \mathcal{N} -letters occurring in E .

Proof: By induction on p .

- If $p = 1$, then E is $\langle \uparrow \alpha_1 \rangle$, $\langle \downarrow \alpha_1 \rangle$ or $\langle \downarrow \uparrow \alpha_1 \rangle$ for an \mathcal{N} -word α_1 . The corresponding formal DNA molecule X is $\binom{\alpha_1}{-}$, $\binom{-}{\alpha_1}$ or $\binom{\alpha_1}{c(\alpha_1)}$, respectively. In each of the cases,

$$|E| = 3 + |\alpha_1| = 3 \cdot p + |X| = 3 \cdot p + |\nu(X)|.$$

- Let $p \geq 1$, and suppose that the claim holds for all DNA expressions containing at most p operators (induction hypothesis). Now assume that E contains $p + 1$ operators. E is either a \uparrow -expression, or a \downarrow -expression or a $\downarrow \uparrow$ -expression.
 - If E is a \uparrow -expression, hence $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$ for some $n \geq 1$ and \mathcal{N} -words and DNA expressions $\varepsilon_1, \dots, \varepsilon_n$, then by definition,

$$X = \mathcal{S}(E) = \nu^+(\mathcal{S}^+(\varepsilon_1))y_1\nu^+(\mathcal{S}^+(\varepsilon_2))y_2 \dots y_{n-1}\nu^+(\mathcal{S}^+(\varepsilon_n)),$$

where the y_i 's are Δ or λ (see (2.15)). The function ν is a homomorphism that removes all nick letters occurring in its argument. This implies that

$$\begin{aligned} |\nu(X)| &= |\nu(\mathcal{S}^+(\varepsilon_1))\nu(\mathcal{S}^+(\varepsilon_2)) \dots \nu(\mathcal{S}^+(\varepsilon_n))| \\ &= |\nu(\mathcal{S}^+(\varepsilon_1))| + |\nu(\mathcal{S}^+(\varepsilon_2))| + \dots + |\nu(\mathcal{S}^+(\varepsilon_n))|. \end{aligned}$$

Apart from the outermost operator, all operators in E occur in the arguments $\varepsilon_1, \dots, \varepsilon_n$. For $i = 1, \dots, n$, let p_i be the number of operators occurring in ε_i . Then

$$p_1 + p_2 + \dots + p_n = p.$$

If an argument ε_i is an \mathcal{N} -word α_i , then $\mathcal{S}^+(\varepsilon_i) = \binom{\alpha_i}{-}$, $p_i = 0$ and

$$|\varepsilon_i| = |\alpha_i| = 3 \cdot p_i + |\nu(\mathcal{S}^+(\varepsilon_i))|.$$

If, on the other hand, an argument ε_i is a DNA expression, then $\mathcal{S}^+(\varepsilon_i) = \mathcal{S}(\varepsilon_i)$, $1 \leq p_i \leq p$ and by the induction hypothesis,

$$|\varepsilon_i| = 3 \cdot p_i + |\nu(\mathcal{S}(\varepsilon_i))| = 3 \cdot p_i + |\nu(\mathcal{S}^+(\varepsilon_i))|.$$

When we combine all equations, we obtain

$$\begin{aligned} |E| &= 3 + |\varepsilon_1| + \dots + |\varepsilon_n| \\ &= 3 + (3 \cdot p_1 + |\nu(\mathcal{S}^+(\varepsilon_1))|) + \dots + (3 \cdot p_n + |\nu(\mathcal{S}^+(\varepsilon_n))|) \\ &= 3 \cdot (p + 1) + |\nu(X)|. \end{aligned}$$

- If E is a \downarrow -expression containing $p + 1$ operators, then the proof is analogous.
- Finally, if E is a \uparrow -expression, then $E = \langle \uparrow E_1 \rangle$ for a DNA expression E_1 containing $p \geq 1$ operators. Hence,

$$X = \mathcal{S}(E) = \kappa(\mathcal{S}(E_1)).$$

Because the functions ν and κ are commutative and κ does not change the length of its argument, we have

$$|\nu(X)| = |\nu(\kappa(\mathcal{S}(E_1)))| = |\kappa(\nu(\mathcal{S}(E_1)))| = |\nu(\mathcal{S}(E_1))|.$$

We can apply the induction hypothesis to E_1 :

$$|E_1| = 3 \cdot p + |\nu(\mathcal{S}(E_1))|.$$

We then find

$$|E| = 3 + |E_1| = 3 + 3 \cdot p + |\nu(\mathcal{S}(E_1))| = 3 \cdot (p + 1) + |\nu(X)|.$$

Hence, the claim is also valid for every DNA expression E that contains $p + 1$ operators. □

We will concentrate on the minimal length of a DNA expression E denoting a given (expressible) formal DNA molecule X . First we derive lower bounds for the length of a DNA expression with the desired semantics. Subsequently, we demonstrate that these lower bounds are tight, i.e., that they can actually be achieved. We describe and analyse the DNA expressions that achieve the lower bounds.

Obviously, there does not exist an upper bound on the length of DNA expressions denoting a certain formal DNA molecule. Indeed, consider an arbitrary DNA expression $E = \langle |_0 \varepsilon_1 \dots \varepsilon_n \rangle$, where $|_0$ is an operator, $n \geq 1$ and $\varepsilon_1, \dots, \varepsilon_n$ are the arguments of E . Then $E' = \langle |_0 E \rangle$ is an equivalent DNA expression, but $|E'| = |E| + 3$. This way, we can extend a DNA expression infinitely.

4.1 Lower bounds for the length of a DNA expression

Obviously, the minimal length of a DNA expression denoting a certain (expressible) formal DNA molecule X depends on X . We will see that it particularly depends on a few simple counting functions of X , to be defined in Definition 4.8 below. Apart from one, each of these functions counts the number of components of X of a certain type.

A formal DNA molecule may have double components, upper components, lower components, upper nick letters and lower nick letters. The last four types of components are categorized as follows:

Definition 4.2 *Let X be a formal DNA molecule and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .*

- *A \uparrow -component x'_i of X is an upper component or a lower nick letter occurring in X .*
- *A \downarrow -component x'_i of X is a lower component or an upper nick letter occurring in X .*

Recall that if $X = \mathcal{S}(E)$ for a DNA expression E , then upper components and lower nick letters occurring in X are the products of an operator \uparrow . Similarly, lower components and upper nick letters are produced by an operator \downarrow . This explains the terms \uparrow -component and \downarrow -component.

We are not interested in all \uparrow -components and \downarrow -components, but only in \uparrow -components and \downarrow -components with an additional property:

Definition 4.3 *Let X be a formal DNA molecule and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .*

- *An opening \uparrow -component of X is a \uparrow -component x'_i of X , such that either $i \leq 2$, or x'_{i-2} is a \downarrow -component of X .*
- *A closing \uparrow -component of X is a \uparrow -component x'_i of X , such that either $i \geq k-1$, or x'_{i+2} is a \downarrow -component of X .*
- *An opening \downarrow -component of X is a \downarrow -component x'_i of X , such that either $i \leq 2$, or x'_{i-2} is a \uparrow -component of X .*
- *A closing \downarrow -component of X is a \downarrow -component x'_i of X , such that either $i \geq k-1$, or x'_{i+2} is a \uparrow -component of X .*

An opening \uparrow -component can be intuitively understood as the first of a series of \uparrow -components of X at the beginning of X or following a \downarrow -component. It represents a transition (from \downarrow) to \uparrow . There are analogous interpretations of a closing upper component and of opening and closing lower components.

In the DNA molecule depicted in Figure 4.1, the opening \uparrow -components are $\binom{\alpha_4}{-}$, $\binom{\alpha_8}{-}$ and $\binom{\alpha_{13}}{-}$, the closing \uparrow -components are $\binom{\alpha_4}{-}$, $\binom{\alpha_{10}}{-}$ and $\binom{\alpha_{13}}{-}$, the opening \downarrow -components are the first nick letter, $\binom{-}{\alpha_6}$, the third nick letter and the fourth nick letter, and the closing \downarrow -components are the second nick letter, $\binom{-}{\alpha_6}$, the third nick

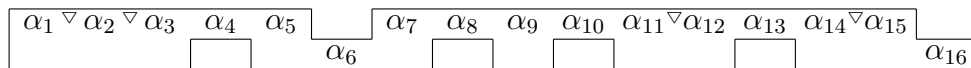


Figure 4.1: Pictorial representation of an example formal DNA molecule X that contains (upper) nick letters.

letter and $\binom{-}{\alpha_{16}}$. In this example, some, but not all opening \uparrow -components are also closing \uparrow -components, and some, but not all opening \downarrow -components are also closing \downarrow -components.

Lemma 4.4 *Let X be a formal DNA molecule.*

1. *The number of opening \uparrow -components of X is equal to the number of closing \uparrow -components of X .*
2. *The number of opening \downarrow -components of X is equal to the number of closing \downarrow -components of X .*

Proof: By Lemma 2.4, X is an alternating sequence of double components and other types of components. The other types of components can be categorized as \uparrow -components and \downarrow -components. Let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .

1. Let x'_{i_0} with $1 \leq i_0 \leq k$ be an arbitrary opening \uparrow -component of X . Then we define j_0 as the largest index with $i_0 \leq j_0 \leq k$ and $i_0 \equiv j_0 \pmod{2}$ such that each of $x'_{i_0}, x'_{i_0+2}, x'_{i_0+4}, \dots, x'_{j_0}$ is a \uparrow -component. Clearly, j_0 is well defined. If $j_0 \leq k-2$, then x'_{j_0+2} must be a \downarrow -component, which implies that x'_{j_0} is a closing \uparrow -component. If $j_0 \geq k-1$, then by definition x'_{j_0} is a closing \uparrow -component.

This way, for each opening \uparrow -component x'_{i_0} of X , we find a corresponding closing \uparrow -component x'_{j_0} . It is easy to verify that different opening \uparrow -components x'_{i_0} correspond to different closing \uparrow -components x'_{j_0} . Hence, the number of opening \uparrow -components is at most as large as the number of closing \uparrow -components.

In a completely analogous way, we can prove that the number of closing \uparrow -components is at most as large as the number of opening \uparrow -components. Thus, these numbers are equal.

2. The proof of this claim is analogous to that of the previous claim.

□

The above proof is based on a natural correspondence between opening \uparrow -components and closing \uparrow -components. This correspondence will appear to be useful at several other places in this chapter. Therefore, we also define it formally:

Definition 4.5 *Let X be a formal DNA molecule and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .*

- *For an opening \uparrow -component x'_{i_0} , the corresponding closing \uparrow -component is the component x'_{j_0} for which j_0 is the largest index with $i_0 \leq j_0 \leq k$ and $i_0 \equiv j_0 \pmod{2}$, such that each of $x'_{i_0}, x'_{i_0+2}, \dots, x'_{j_0-2}, x'_{j_0}$ is a \uparrow -component. Conversely, x'_{i_0} is the corresponding opening \uparrow -component for the closing \uparrow -component x'_{j_0} .*

- For an opening \downarrow -component x'_{i_0} , the corresponding closing \downarrow -component is the component x'_{j_0} for which j_0 is the largest index with $i_0 \leq j_0 \leq k$ and $i_0 \equiv j_0 \pmod{2}$, such that each of $x'_{i_0}, x'_{i_0+2}, \dots, x'_{j_0-2}, x'_{j_0}$ is a \downarrow -component. Conversely, x'_{i_0} is the corresponding opening \downarrow -component for the closing \downarrow -component x'_{j_0} .

It is useful to realize what type of components may occur between an opening \uparrow -component (or opening \downarrow -component) and the corresponding closing \uparrow -component (or closing \downarrow -component, respectively).

Lemma 4.6 *Let X be a formal DNA molecule and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .*

1. Let x'_{i_0} be an opening \uparrow -component and let x'_{j_0} be the corresponding closing \uparrow -component. Then for $i = i_0 + 2, i_0 + 4, \dots, j_0 - 2, j_0$, x'_i is a \uparrow -component, but not an opening \uparrow -component. For $i = i_0 + 1, i_0 + 3, \dots, j_0 - 3, j_0 - 1$, x'_i is a double component.
2. Let x'_{i_0} be an opening \downarrow -component and let x'_{j_0} be the corresponding closing \downarrow -component. Then for $i = i_0 + 2, i_0 + 4, \dots, j_0 - 2, j_0$, x'_i is a \downarrow -component, but not an opening \downarrow -component. For $i = i_0 + 1, i_0 + 3, \dots, j_0 - 3, j_0 - 1$, x'_i is a double component.

Proof:

1. By the definition of a corresponding closing \uparrow -component, for $i = i_0 + 2, i_0 + 4, \dots, j_0 - 2, j_0$, both x'_{i-2} and x'_i are \uparrow -components. Hence, none of $x'_{i_0+2}, x'_{i_0+4}, \dots, x'_{j_0-2}, x'_{j_0}$ is an opening \uparrow -component.

For $i = i_0 + 1, i_0 + 3, \dots, j_0 - 3, j_0 - 1$, $i \equiv i_0 + 1 \pmod{2}$. Hence, by Lemma 2.4, x'_i is a double component of X .

2. The proof of this claim is analogous to that of the previous claim.

□

Lemma 4.4 allows us to concentrate either on the opening \uparrow -components and the opening \downarrow -components, or on the closing \uparrow -components and the closing \downarrow -components. We arbitrarily choose for the first alternative.

Lemma 4.7 *Let X be a formal DNA molecule. The opening \uparrow -components and opening \downarrow -components occur in X alternately.*

Proof: Let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .

Let x'_{i_0} with $1 \leq i_0 \leq k$ be an arbitrary opening \uparrow -component of X and let x'_{j_0} with $i_0 \leq j_0 \leq k$ be the corresponding closing \uparrow -component.

By Lemma 2.4, the components x'_i with $i \equiv i_0 + 1 \pmod{2}$ are double components. Hence, none of them is an opening \uparrow -component or an opening \downarrow -component. By Lemma 4.6(1), none of the components $x'_{i_0+2}, x'_{i_0+4}, \dots, x'_{j_0}$ is an opening \uparrow -component or an opening \downarrow -component, either.

If $j_0 \geq k - 1$, then apparently, X does not contain any opening \uparrow -component or opening \downarrow -component after x'_{i_0} . If, on the other hand, $j_0 \leq k - 2$, then x'_{j_0+2} is a \downarrow -component. In particular, it is an opening \downarrow -component.

Consequently, for each opening \uparrow -component x'_{i_0} of X , if X contains opening \uparrow -components or opening \downarrow -components after x'_{i_0} , then the first one of them (x'_{j_0+2}) is an opening \downarrow -component.

In a completely analogous way, we can prove: for each opening \downarrow -component x'_{i_0} of X , if X contains opening \uparrow -components or opening \downarrow -components after x'_{i_0} , then the first one of them is an opening \uparrow -component.

Hence, the opening \uparrow -components and opening \downarrow -components of X alternate. \square

We now define functions that count the opening \uparrow -components, the opening \downarrow -components and the double components occurring in a formal DNA molecule X .

Definition 4.8 *Let X be a formal DNA molecule. Then*

- $T_{\uparrow}(X)$ is the number of opening \uparrow -components of X ,
- $T_{\downarrow}(X)$ is the number of opening \downarrow -components of X ,
- $n_{\uparrow}(X)$ is the number of double components of X .

For the formal DNA molecule X from Figure 4.1, we have $T_{\uparrow}(X) = 3$, $T_{\downarrow}(X) = 4$ and $n_{\uparrow}(X) = 10$.

We are interested in the values of the functions T_{\uparrow} , T_{\downarrow} and n_{\uparrow} for formal DNA molecules X . Sometimes, however, it will be convenient to have the possibility to provide an argument $X = \lambda$. In line with the intuition, we define

$$T_{\uparrow}(\lambda) = T_{\downarrow}(\lambda) = n_{\uparrow}(\lambda) = 0.$$

In addition to the three new counting functions, we will frequently use $\#\nabla(X)$, $\#\triangle(X)$, $\#\nabla,\triangle(X)$, $\#\uparrow,\downarrow(E)$ and $\#\uparrow(E)$. Here, X and E may be arbitrary strings, but often X will be a formal DNA molecule and E will be a DNA expression.

The following result is immediate from Lemma 2.4:

Lemma 4.9 *Let X be a formal DNA molecule and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .*

If x'_1 is a double component, then $n_{\uparrow}(X) = \lceil \frac{k}{2} \rceil$. If x'_1 is not a double component, then $n_{\uparrow}(X) = \lfloor \frac{k}{2} \rfloor$.

Another simple result is

Lemma 4.10 *Let X be a nick free formal DNA molecule.*

1. $T_{\uparrow}(X) = 0$ if and only if X does not contain any upper component.
2. $T_{\downarrow}(X) = 0$ if and only if X does not contain any lower component.

Proof: Let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .

1. \implies Assume that $T_{\uparrow}(X) = 0$ and suppose that X contains at least one upper component. Let i_0 be the smallest index such that x'_{i_0} is an upper component of X .

If $i_0 \leq 2$, then by definition, x'_{i_0} is an opening \uparrow -component. If, on the other hand, $i_0 \geq 3$, then by Corollary 2.5, x'_{i_0-2} is either an upper component, or a lower component of X . By the definition of i_0 , x'_{i_0-2} has to be a lower component. Then, however, x'_{i_0} is an opening \uparrow -component.

In both cases, $T_{\uparrow}(X)$ would be greater than 0, which contradicts our assumption. Hence, X does not contain any upper component.

\longleftarrow Because X is nick free, each opening \uparrow -component is an upper component. If X does not contain any upper component, then X certainly does not contain any opening \uparrow -component. Thus, $T_{\uparrow}(X) = 0$.

2. The proof of this claim is analogous to that of the previous claim.

□

The different counting functions on formal DNA molecules can be related to each other:

Lemma 4.11 *Let X be a formal DNA molecule.*

1. $T_{\uparrow}(X) \leq T_{\downarrow}(X) + 1$.
2. $T_{\downarrow}(X) \leq T_{\uparrow}(X) + 1$.
3. If X does not contain upper components, nor lower components, then $n_{\uparrow}(X) = \#_{\nabla, \Delta}(X) + 1$.
4. If X contains at least one nick letter, then $n_{\uparrow}(X) \geq \#_{\nabla, \Delta}(X) + 1$.
5. $n_{\uparrow}(X) \geq \#_{\nabla, \Delta}(X)$.
6. $\#_{\nabla, \Delta}(X) = \#_{\nabla}(X) + \#_{\Delta}(X)$.

Proof: Because, by Lemma 4.7, the opening \uparrow -components and opening \downarrow -components occur in X alternately, the difference between their numbers of occurrences can be at most 1. Now Claims 1 and 2 follow immediately.

We proceed with the other claims.

3. This claim follows immediately from Corollary 2.6.
4. Let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X . By the definition of a formal DNA molecule, every nick letter occurring in X is preceded and succeeded by a double component. Hence, for each component x'_i of X that is a nick letter, x'_{i-1} is a double component. Further, if x'_{i_0} is the last nick letter occurring in X , then also x'_{i_0+1} is a double component. Obviously, all these double components are different, and thus $n_{\uparrow}(X) \geq \#_{\nabla, \Delta}(X) + 1$.
5. If X is nick free (hence $\#_{\nabla, \Delta}(X) = 0$), then the claim holds because $n_{\uparrow}(X) \geq 0$. If X is not nick free, then the claim follows from Claim 4.

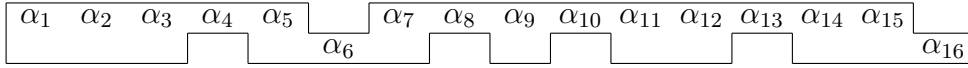


Figure 4.2: Pictorial representation of $\nu^+(X)$ for the formal DNA molecule X from Figure 4.1.

6. This claim is valid by definition. □

For nick free formal DNA molecules, we study some specific cases of Lemma 4.11(1) and (2). Note that for such formal DNA molecules, each opening \uparrow -component is an upper component and each opening \downarrow -component is a lower component.

Lemma 4.12 *Let X be a nick free formal DNA molecule that contains at least one single-stranded component.*

1. *If the first single-stranded component of X is an upper component and the last single-stranded component of X is an upper component, then $T_\uparrow(X) = T_\downarrow(X) + 1$.*
2. *If the first single-stranded component of X is a lower component and the last single-stranded component of X is an upper component, then $T_\uparrow(X) = T_\downarrow(X)$.*
3. *If the first single-stranded component of X is an upper component and the last single-stranded component of X is a lower component, then $T_\uparrow(X) = T_\downarrow(X)$.*
4. *If the first single-stranded component of X is a lower component and the last single-stranded component of X is a lower component, then $T_\uparrow(X) = T_\downarrow(X) - 1$.*

Proof: Let $x'_1 \dots x'_k$ be the decomposition of X .

By Corollary 2.5, there exists $i \in \{1, 2\}$ such that x'_i is a single-stranded component. If x'_i is an upper component, then it is an opening \uparrow -component. If it is a lower component, then it is an opening \downarrow -component.

Analogously, there exists $j \in \{k-1, k\}$ such that x'_j is a single-stranded component. If x'_j is an upper component, then it is a closing \uparrow -component. By Lemma 4.6(1) (and Corollary 2.5), the corresponding opening \uparrow -component is not succeeded in X by any other opening \uparrow -component or opening \downarrow -component. If, on the other hand, x'_j is a lower component, then it is a closing \downarrow -component. The corresponding opening \downarrow -component is not succeeded in X by any other opening \uparrow -component or opening \downarrow -component.

Now, all claims follow from the observation that, by Lemma 4.7, the opening \uparrow -components and the opening \downarrow -components of X alternate. □

Next, we investigate the effect of applying the function ν^+ to a formal DNA molecule, i.e., the effect of removing the upper nick letters. Of course, we can achieve analogous results for ν^- .

In Figure 4.2, we have depicted $\nu^+(X)$ for the formal DNA molecule X from Figure 4.1. As we have established before,

$$T_\uparrow(X) = 3, \quad T_\downarrow(X) = 4, \quad n_\uparrow(X) = 10 \quad \text{and} \quad \#_{\nabla}(X) = 4$$

Now,

$$T_\uparrow(\nu^+(X)) = 2, \quad T_\downarrow(\nu^+(X)) = 2, \quad n_\uparrow(\nu^+(X)) = 6 \quad \text{and} \quad \#_{\nabla}(\nu^+(X)) = 0.$$

In general, we have:

Lemma 4.13 *Let X be a formal DNA molecule.*

1. $T_{\uparrow}(\nu^+(X)) \leq T_{\uparrow}(X)$.
2. $T_{\downarrow}(\nu^+(X)) \leq T_{\downarrow}(X)$.
3. $T_{\downarrow}(\nu^+(X)) \geq T_{\downarrow}(X) - \#\nabla(X)$.
4. $n_{\uparrow}(\nu^+(X)) = n_{\uparrow}(X) - \#\nabla(X)$.
5. $\#\nabla(\nu^+(X)) = 0$.
6. $\#\triangle(\nu^+(X)) = \#\triangle(X)$.

Proof: We will only prove Claims 1 – 4, because the other two claims are immediate from the definition of the function ν^+ .

Let $x'_1 \dots x'_{k_1}$ for some $k_1 \geq 1$ be the decomposition of X and let $\hat{x}'_1 \dots \hat{x}'_{k_2}$ for some $k_2 \geq 1$ be the decomposition of $\nu^+(X)$.

When the function ν^+ is applied to X , the only effect is that upper nick letters occurring in X are removed. By the definition of a formal DNA molecule (Definition 2.1), (upper) nick letters may occur only between two double components. When an upper nick letter is removed, the two double components become adjacent and thus form one larger double component.

Upper components, lower components and lower nick letters are not affected by the application of ν^+ . Hence, every upper component x'_i of X uniquely corresponds to an (equal) upper component \hat{x}'_j of $\nu^+(X)$, and vice versa, and there exist analogous correspondences between lower components and between lower nick letters.

1. Let \hat{x}'_{j_0} be an opening \uparrow -component of $\nu^+(X)$. Because \hat{x}'_{j_0} is an upper component or a lower nick letter, it corresponds to an equal component x'_{i_0} of X .

If $i_0 \leq 2$, then by definition x'_{i_0} is an opening \uparrow -component of X .

If $i_0 \geq 3$, then by Lemma 2.4, x'_{i_0-1} is a double component and x'_{i_0-2} is an upper component, a lower component or a nick letter. If x'_{i_0-2} is an upper component or a lower nick letter, then so is x'_{j_0-2} , because such components are not affected by the application of ν^+ (and the double component x'_{i_0-1} is not removed). As a result, x'_{j_0} would not be an opening \uparrow -component of $\nu^+(X)$. Because this contradicts our prior assumption, x'_{i_0-2} must be a lower component or an upper nick letter, which implies that x'_{i_0} is an opening \uparrow -component of X .

Hence, for each opening \uparrow -component \hat{x}'_{j_0} of $\nu^+(X)$, the corresponding component x'_{i_0} of X (an upper component or a lower nick letter) is also an opening \uparrow -component. Because of the 1–1 correspondence between upper components of X and upper components of $\nu^+(X)$ and between lower nick letters occurring in X and lower nick letters occurring in $\nu^+(X)$, different opening \uparrow -components of $\nu^+(X)$ correspond to different opening \uparrow -components of X . Thus, $T_{\uparrow}(\nu^+(X)) \leq T_{\uparrow}(X)$.

2. Let \hat{x}'_{j_0} be an opening \downarrow -component of $\nu^+(X)$. Because $\nu^+(X)$ is free of upper nick letters, \hat{x}'_{j_0} is a lower component of $\nu^+(X)$. Let x'_{i_0} be the corresponding lower component of X .

If $i_0 \leq 2$, then by definition x'_{i_0} is an opening \downarrow -component of X .

If $i_0 \geq 3$, then we examine the possibilities for the component x'_{i_0-2} . First, it cannot be a lower component of X . Otherwise, also \hat{x}'_{j_0-2} would be a lower component and \hat{x}'_{j_0} would not be an opening \downarrow -component of $\nu^+(X)$. Next, if x'_{i_0-2} is an upper component or a lower nick letter, then x'_{i_0} is an opening \downarrow -component of X .

Finally, if x'_{i_0-2} is an upper nick letter (which is removed by ν^+), then the double component \hat{x}'_{j_0-1} is the concatenation of at least two double components x'_i of X , which are separated in X by upper nick letters. Let x'_{i_1} be the first of these upper nick letters, i.e., the leftmost one.

If $i_1 \leq 2$, then x'_{i_1} is an opening \downarrow -component. If $i_1 \geq 3$, then by definition x'_{i_1-2} is not another upper nick letter. Further, x'_{i_1-2} cannot be a lower component, for the same reason that x'_{i_0-2} could not be a lower component. Hence, x'_{i_1-2} must be an upper component or a lower nick letter and thus x'_{i_1} is an opening \downarrow -component.

Hence, for each opening \downarrow -component \hat{x}'_{j_0} of $\nu^+(X)$ (a lower component), we have found a corresponding opening \downarrow -component of X (either the corresponding lower component x'_{i_0} or the upper nick letter x'_{i_1}).

Because, by Lemma 4.7, opening \uparrow -components and opening \downarrow -components of a formal DNA molecule alternate, different opening \downarrow -components of $\nu^+(X)$ are separated by at least an upper component or a lower nick letter. It follows from the construction that the corresponding opening \downarrow -components of X are also separated by at least an (equal) upper component, or a lower nick letter, respectively. In particular, these opening \downarrow -components of X are different. Therefore, $T_{\downarrow}(\nu^+(X)) \leq T_{\downarrow}(X)$.

3. Let x'_{i_0} be an opening \downarrow -component of X which is not an upper nick letter. Then it must be a lower component of X , which corresponds to a(n equal) lower component \hat{x}'_{j_0} of $\nu^+(X)$.

If $i_0 \leq 2$, then, because the function ν^+ does not add components to its argument, certainly $j_0 \leq 2$, so that by definition \hat{x}'_{j_0} is an opening \downarrow -component.

If $i_0 \geq 3$, then x'_{i_0-2} is an upper component of X or a lower nick letter. Because such components are not affected by ν^+ , also $j_0 \geq 3$ and \hat{x}'_{j_0-2} is an (equal) upper component of $\nu^+(X)$ or a lower nick letter, respectively. Also in this case, \hat{x}'_{j_0} is an opening \downarrow -component.

Hence, for each opening \downarrow -component x'_{i_0} of X which is a lower component, the corresponding lower component \hat{x}'_{j_0} of $\nu^+(X)$ is an opening \downarrow -component of $\nu^+(X)$. Because of the 1–1 correspondence between lower components of X and lower components of $\nu^+(X)$, $T_{\downarrow}(\nu^+(X)) \geq T_{\downarrow}(X) - \#\nabla(X)$.

4. As we have mentioned at the beginning of the proof, when an upper nick letter x'_{i_0} is removed, the two double components x'_{i_0-1} and x'_{i_0+1} become adjacent and thus form one larger double component. That way, both the number of upper nick letters and the number of double components are decreased by 1. Because all upper nick letters occurring in X are removed and this is the only effect of the application of ν^+ , the number of double components is decreased by $\#\nabla(X)$.

□

By the above result, we can derive invariants for the functions ν^+ , ν^- and ν : the function $n_{\uparrow}(\cdot) - \#_{\nabla}(\cdot)$ is an invariant for the function ν^+ , the function $n_{\uparrow}(\cdot) - \#_{\Delta}(\cdot)$ is an invariant for the function ν^- , and the function $n_{\uparrow}(\cdot) - \#_{\nabla, \Delta}(\cdot)$ is an invariant for the function ν .

Corollary 4.14 *Let X be a formal DNA molecule*

1. $n_{\uparrow}(\nu^+(X)) - \#_{\nabla}(\nu^+(X)) = n_{\uparrow}((X)) - \#_{\nabla}((X))$
2. $n_{\uparrow}(\nu^-(X)) - \#_{\Delta}(\nu^-(X)) = n_{\uparrow}((X)) - \#_{\Delta}((X))$
3. $n_{\uparrow}(\nu(X)) - \#_{\nabla, \Delta}(\nu(X)) = n_{\uparrow}((X)) - \#_{\nabla, \Delta}((X))$

Proof: Claim 1 follows immediately from Lemma 4.13(4) and (5). Then by analogy, we also have Claim 2. Finally, Claim 3 follows from the other two and from Lemma 4.13(6), because the function ν is the composition of ν^+ and ν^- :

$$\begin{aligned}
& n_{\uparrow}(\nu(X)) - \#_{\nabla, \Delta}(\nu(X)) \\
&= n_{\uparrow}(\nu^+(\nu^-(X))) - \#_{\nabla}(\nu^+(\nu^-(X))) - \#_{\Delta}(\nu^+(\nu^-(X))) \\
&= n_{\uparrow}(\nu^-(X)) - \#_{\nabla}(\nu^-(X)) - \#_{\Delta}(\nu^-(X)) \\
&= n_{\uparrow}(X) - \#_{\nabla}(X) - \#_{\Delta}(X) \\
&= n_{\uparrow}(X) - \#_{\nabla, \Delta}(X).
\end{aligned}$$

□

We also study the effect of applying the function κ to a formal DNA molecule, i.e., the effect of making the molecule double stranded.

Lemma 4.15 *Let X be a formal DNA molecule.*

1. $T_{\uparrow}(\kappa(X)) \leq T_{\uparrow}(X)$.
2. $T_{\downarrow}(\kappa(X)) \leq T_{\downarrow}(X)$.
3. $\#_{\nabla, \Delta}(\kappa(X)) = \#_{\nabla, \Delta}(X)$.
4. $n_{\uparrow}(\kappa(X)) \leq n_{\uparrow}(X) + 1$.

Proof: The function κ replaces each occurrence of an upper \mathcal{A} -letter and a lower \mathcal{A} -letter in X by the corresponding double \mathcal{A} -letter. This is all the function does. Consequently, $\kappa(X)$ does not contain upper components, nor lower components. Because κ does not introduce, nor remove nick letters, there exist a 1–1 correspondence between the nick letters occurring in X and the (same) nick letters occurring in $\kappa(X)$.

Let $x'_1 \dots x'_{k_1}$ for some $k_1 \geq 1$ be the decomposition of X and let $\hat{x}'_1 \dots \hat{x}'_{k_2}$ for some $k_2 \geq 1$ be the decomposition of $\kappa(X)$.

1. Let \hat{x}'_{j_0} be an opening \uparrow -component of $\kappa(X)$. By the above, it must be a lower nick letter Δ . Let x'_{i_0} be the corresponding (occurrence of a) lower nick letter in X . By Lemma 2.4, each component x'_i of X with $i \equiv i_0 \pmod{2}$ is either an upper component, or a lower component, or a nick letter.

Now, we define i_1 as the smallest index with $1 \leq i_1 \leq i_0$ and $i_1 \equiv i_0 \pmod{2}$ such that each of $x'_{i_1}, x'_{i_1+2}, x'_{i_1+4}, \dots, x'_{i_0}$ is a \uparrow -component of X .

If $i_1 \leq 2$, then by definition x'_{i_1} is an opening \uparrow -component of X . If $i_1 \geq 3$, then apparently x'_{i_1-2} is a \downarrow -component. Also in this case, x'_{i_1} is an opening \uparrow -component of X .

Hence, for each opening \uparrow -component \hat{x}'_{j_0} of $\kappa(X)$ (a lower nick letter), we find an opening \uparrow -component x'_{i_1} of X . By Lemma 4.7, different opening \uparrow -components of $\kappa(X)$ are separated by at least an opening \downarrow -component, which must be an upper nick letter. Then it follows from the construction that the corresponding opening \uparrow -components of X are also separated by at least an upper nick letter. In particular, these opening \uparrow -components of X are different. Consequently, $T_{\uparrow}(\kappa(X)) \leq T_{\uparrow}(X)$.

2. The proof of this claim is entirely analogous to that of the previous claim.
3. This claim follows immediately from the 1–1 correspondence between the nick letters occurring in X and the nick letters occurring in $\kappa(X)$.
4. When we combine Lemma 4.11(3), Claim 3 and Lemma 4.11(5), we obtain:

$$n_{\uparrow}(\kappa(X)) = \#_{\nabla, \Delta}(\kappa(X)) + 1 = \#_{\nabla, \Delta}(X) + 1 \leq n_{\uparrow}(X) + 1.$$

□

When a formal DNA molecule is denoted by a DNA expression, the counting numbers for the full molecule can be related to the numbers for the arguments of the DNA expression.

Lemma 4.16 *Let E be a DNA expression, and let $X = \mathcal{S}(E)$.*

1. *If $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$, where $n \geq 1$, the arguments $\varepsilon_1, \dots, \varepsilon_n$ are \mathcal{N} -words or DNA expressions and for $i = 1, \dots, n$, $X_i = \mathcal{S}^+(\varepsilon_i)$, then*

$$T_{\uparrow}(X) - \#_{\Delta}(X) \leq T_{\uparrow}(X_1) + \dots + T_{\uparrow}(X_n) - \#_{\Delta}(X_1) - \dots - \#_{\Delta}(X_n), \quad (4.1)$$

$$T_{\downarrow}(X) \leq T_{\downarrow}(X_1) + \dots + T_{\downarrow}(X_n), \quad (4.2)$$

$$T_{\downarrow}(X) \geq T_{\downarrow}(X_1) + \dots + T_{\downarrow}(X_n) - \#_{\nabla}(X_1) - \dots - \#_{\nabla}(X_n), \quad (4.3)$$

$$n_{\uparrow}(X) = n_{\uparrow}(X_1) + \dots + n_{\uparrow}(X_n) - \#_{\nabla}(X_1) - \dots - \#_{\nabla}(X_n), \quad (4.4)$$

$$\text{and} \quad n_{\downarrow}(X) \leq n_{\downarrow}(X_1) + \dots + n_{\downarrow}(X_n). \quad (4.5)$$

2. *If $E = \langle \downarrow \varepsilon_1 \dots \varepsilon_n \rangle$, where $n \geq 1$, the arguments $\varepsilon_1, \dots, \varepsilon_n$ are \mathcal{N} -words or DNA expressions and for $i = 1, \dots, n$, $X_i = \mathcal{S}^-(\varepsilon_i)$, then*

$$T_{\downarrow}(X) - \#_{\nabla}(X) \leq T_{\downarrow}(X_1) + \dots + T_{\downarrow}(X_n) - \#_{\nabla}(X_1) - \dots - \#_{\nabla}(X_n),$$

$$T_{\uparrow}(X) \leq T_{\uparrow}(X_1) + \dots + T_{\uparrow}(X_n),$$

$$T_{\uparrow}(X) \geq T_{\uparrow}(X_1) + \dots + T_{\uparrow}(X_n) - \#_{\Delta}(X_1) - \dots - \#_{\Delta}(X_n),$$

$$n_{\downarrow}(X) = n_{\downarrow}(X_1) + \dots + n_{\downarrow}(X_n) - \#_{\Delta}(X_1) - \dots - \#_{\Delta}(X_n),$$

$$\text{and} \quad n_{\uparrow}(X) \leq n_{\uparrow}(X_1) + \dots + n_{\uparrow}(X_n).$$

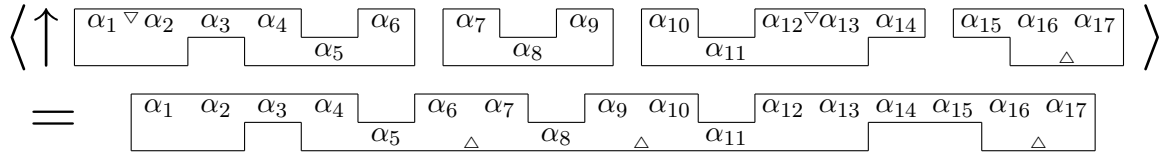


Figure 4.3: Pictorial representation of the example \uparrow -expression E with $\mathcal{S}(E) = X$ for which $T_\uparrow(X)$, $T_\downarrow(X)$ and $n_\uparrow(X)$ are calculated.

3. If $E = \langle \uparrow E_1 \rangle$ for a DNA expression E_1 with $X_1 = \mathcal{S}(E_1)$, then

$$T_\uparrow(X) \leq T_\uparrow(X_1), \quad (4.6)$$

$$T_\downarrow(X) \leq T_\downarrow(X_1), \quad (4.7)$$

$$\#_{\nabla, \triangle}(X) = \#_{\nabla, \triangle}(X_1) \text{ and} \quad (4.8)$$

$$n_\uparrow(X) \leq n_\uparrow(X_1) + 1. \quad (4.9)$$

As an example of Claim 1, consider

$$E = \langle \uparrow \langle \downarrow \langle \uparrow \alpha_1 \rangle \langle \uparrow \langle \downarrow \alpha_2 \rangle \alpha_3 \langle \downarrow \alpha_4 \rangle \rangle \alpha_5 \langle \downarrow \alpha_6 \rangle \rangle \langle \downarrow \langle \downarrow \alpha_7 \rangle \alpha_8 \langle \downarrow \alpha_9 \rangle \rangle \langle \downarrow \langle \downarrow \alpha_{10} \rangle \alpha_{11} \langle \downarrow \alpha_{12} \rangle \langle \uparrow \langle \downarrow \alpha_{13} \rangle \alpha_{14} \rangle \rangle \langle \uparrow \alpha_{15} \langle \downarrow \alpha_{16} \rangle \langle \downarrow \alpha_{17} \rangle \rangle \rangle. \quad (4.10)$$

Here $n = 4$ and

$$\begin{aligned} X_1 &= \binom{\alpha_1}{c(\alpha_1)} \nabla \binom{\alpha_2}{c(\alpha_2)} \binom{\alpha_3}{-} \binom{\alpha_4}{c(\alpha_4)} \binom{-}{\alpha_5} \binom{\alpha_6}{c(\alpha_6)}, \\ X_2 &= \binom{\alpha_7}{c(\alpha_7)} \binom{-}{\alpha_8} \binom{\alpha_9}{c(\alpha_9)}, \\ X_3 &= \binom{\alpha_{10}}{c(\alpha_{10})} \binom{-}{\alpha_{11}} \binom{\alpha_{12}}{c(\alpha_{12})} \nabla \binom{\alpha_{13}}{c(\alpha_{13})} \binom{\alpha_{14}}{-}, \text{ and} \\ X_4 &= \binom{\alpha_{15}}{-} \binom{\alpha_{16}}{c(\alpha_{16})} \triangle \binom{\alpha_{17}}{c(\alpha_{17})}. \end{aligned}$$

Then

$$X = \binom{\alpha_1 \alpha_2}{c(\alpha_1 \alpha_2)} \binom{\alpha_3}{-} \binom{\alpha_4}{c(\alpha_4)} \binom{-}{\alpha_5} \binom{\alpha_6}{c(\alpha_6)} \triangle \binom{\alpha_7}{c(\alpha_7)} \binom{-}{\alpha_8} \binom{\alpha_9}{c(\alpha_9)} \triangle \binom{\alpha_{10}}{c(\alpha_{10})} \binom{-}{\alpha_{11}} \binom{\alpha_{12} \alpha_{13}}{c(\alpha_{12} \alpha_{13})} \binom{\alpha_{14} \alpha_{15}}{-} \binom{\alpha_{16}}{c(\alpha_{16})} \triangle \binom{\alpha_{17}}{c(\alpha_{17})}$$

and

$$\begin{aligned} T_\uparrow(X) - \#_{\triangle}(X) &= 4 - 3 < (1 + 0 + 1 + 1) - (0 + 0 + 0 + 1), \\ T_\downarrow(X) &= 3 < 2 + 1 + 1 + 0, \\ T_\downarrow(X) &= 3 > (2 + 1 + 1 + 0) - (1 + 0 + 1 + 0), \\ n_\uparrow(X) &= 9 = (4 + 2 + 3 + 2) - (1 + 0 + 1 + 0) < (4 + 2 + 3 + 2). \end{aligned}$$

This example is depicted in Figure 4.3.

Note that for \uparrow -expressions as described in Claim 1, the inequality

$$T_\uparrow(X) \leq T_\uparrow(X_1) + \cdots + T_\uparrow(X_n)$$

does not hold in general. Lower nick letters added between the arguments of \uparrow may induce an increase of the number of opening \uparrow -components. For example, for the \uparrow -expression we considered above, $T_{\uparrow}(X) = 4$, whereas $T_{\uparrow}(X_1) + T_{\uparrow}(X_2) + T_{\uparrow}(X_3) + T_{\uparrow}(X_4) = 3$.

Proof of Lemma 4.16:

1. Let E be a \uparrow -expression as described in the claim. By the definition of the semantics of a DNA expression,

$$X = \nu^+(X_1)y_1\nu^+(X_2)y_2 \dots y_{n-1}\nu^+(X_n), \quad (4.11)$$

where for $i = 1, \dots, n-1$, $y_i = \triangle$ if $R(X_i), L(X_{i+1}) \in \mathcal{A}_{\pm}$, and $y_i = \lambda$ otherwise.

Before we prove the separate inequalities, we first analyse the consequences of (4.11) on the components of the formal DNA (sub)molecules.

For $i = 1, \dots, n$, by Lemma 2.7, the first (last) component of $\nu^+(X_i)$ is of the same type (double component, upper component or lower component) as the first (last) component of X_i . Further, $\nu^+(X_i)$ does not contain upper nick letters ∇ . Because only lower nick letters y_i may be added by the application of \uparrow , the entire formal DNA molecule X is free of upper nick letters.

For $i = 1, \dots, n-1$, if both the last component of $\nu^+(X_i)$ and the first component of $\nu^+(X_{i+1})$ are upper components, then they merge into one new upper component of X . If both the last component of $\nu^+(X_i)$ and the first component of $\nu^+(X_{i+1})$ are double components, then a lower nick letter $y_i = \triangle$ is inserted between them. Hence, they are not joined and there is a 1–1 correspondence between the double components of the $\nu^+(X_i)$'s and the (same) double components of X . Because the arguments of \uparrow have to fit together by upper strands, there cannot be a lower component at the end of any of X_1, \dots, X_{n-1} , nor at the beginning of any of X_2, \dots, X_n . Hence, there is also a 1–1 correspondence between the lower components of the $\nu^+(X_i)$'s and the (same) lower components of X .

Now let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X . Further, for $i = 1, \dots, n$, let $x'_{i,1} \dots x'_{i,k_i}$ for some $k_i \geq 1$ be the decomposition of $\nu^+(X_i)$.

- In order to show (4.1), we first prove

$$\begin{aligned} T_{\uparrow}(X) - \#\triangle(X) &\leq T_{\uparrow}(\nu^+(X_1)) + \dots + T_{\uparrow}(\nu^+(X_n)) \\ &\quad - \#\triangle(\nu^+(X_1)) - \dots - \#\triangle(\nu^+(X_n)). \end{aligned} \quad (4.12)$$

Let x'_j be an opening \uparrow -component of X , which is *not* a lower nick letter y_i introduced by the outermost operator \uparrow . Hence, either x'_j is an upper component of X , or x'_j is a lower nick letter which already occurred in some $\nu^+(X_i)$.

In the former case, as we have just seen, x'_j may be the concatenation of several upper components of different, consecutive $\nu^+(X_i)$'s. Assume that the first (and possibly only) upper component that has merged into x'_j is the j_0^{th} component of $\nu^+(X_{i_0})$: x'_{i_0,j_0} . In the latter case, let x'_{i_0,j_0} be the lower nick letter corresponding to x'_j .

If $j_0 \leq 2$, then by definition x'_{i_0,j_0} is an opening \uparrow -component of $\nu^+(X_{i_0})$.

If $j_0 \geq 3$, then by Lemma 2.4, x'_{i_0, j_0-1} is a double component and x'_{i_0, j_0-2} is an upper component, a lower component or a nick letter. It cannot be an upper nick letter, because $\nu^+(X_{i_0})$ does not contain upper nick letters. If x'_{i_0, j_0-2} were an upper component or a lower nick letter, then x'_{j-2} would also be an upper component (possibly extended) or a lower nick letter, respectively. This would, however, contradict the assumption that x'_j is an opening \uparrow -component of X . Hence, x'_{i_0, j_0-2} is a lower component, which implies that x'_{i_0, j_0} is an opening \uparrow -component of $\nu^+(X_{i_0})$.

Hence, for each opening \uparrow -component of X which is not a lower nick letter y_i introduced by the outermost operator \uparrow , there is at least one opening \uparrow -component of $\nu^+(X_i)$ for a certain i . Because, obviously, each upper component of an $\nu^+(X_i)$ merges into exactly one upper component of X , different opening \uparrow -components of X correspond to different opening \uparrow -components of the $\nu^+(X_i)$'s.

The number of opening \uparrow -components of X which *are* lower nick letters y_i introduced by the outermost operator \uparrow , is at most as large as the total number of lower nick letters y_i introduced. The latter number can be written as $|y_1 \dots y_{n-1}|$. We thus have

$$\begin{aligned} T_{\uparrow}(X) - |y_1 \dots y_{n-1}| \\ &\leq T_{\uparrow}(X) - |\{y_i | (1 \leq i \leq n-1) \wedge (y_i \text{ is an opening } \uparrow\text{-component of } X)\}| \\ &\leq T_{\uparrow}(\nu^+(X_1)) + \dots + T_{\uparrow}(\nu^+(X_n)). \end{aligned}$$

When we subtract the number of lower nick letters occurring in the $\nu^+(X_i)$'s ($\#_{\Delta}(\nu^+(X_1)) + \dots + \#_{\Delta}(\nu^+(X_n))$), we obtain (4.12).

Now, (4.1) follows from Lemma 4.13(1) and (6).

- In order to show (4.2) and (4.3), we first prove

$$T_{\downarrow}(X) = T_{\downarrow}(\nu^+(X_1)) + \dots + T_{\downarrow}(\nu^+(X_n)). \quad (4.13)$$

Let x'_j be an opening \downarrow -component of X . Because X does not contain upper nick letters, x'_j must be a lower component. The lower components of X are exactly the ones from $\nu^+(X_1), \dots, \nu^+(X_n)$. Hence, x'_j is equal to a lower component of $\nu^+(X_{i_0})$ for a certain i_0 , say to x'_{i_0, j_0} .

If $j_0 \leq 2$, then by definition x'_{i_0, j_0} is an opening \downarrow -component of $\nu^+(X_{i_0})$.

If $j_0 \geq 3$, then by Lemma 2.4, x'_{i_0, j_0-2} is an upper component, a lower component or a nick letter. It obviously cannot be an upper nick letter. It cannot be a lower component either, because otherwise x'_{j-2} would be a lower component of X and x'_j would not be an opening \downarrow -component of X . Hence, x'_{i_0, j_0-2} is an upper component or a lower nick letter. In both cases, x'_{i_0, j_0} is an opening \downarrow -component of $\nu^+(X_{i_0})$.

Hence, for each opening \downarrow -component of X (a lower component) the corresponding lower component x'_{i_0, j_0} of $\nu^+(X_{i_0})$ is also an opening \downarrow -component.

On the other hand, let x'_{i_0, j_0} for some i_0 and j_0 be an opening \downarrow -component of $\nu^+(X_{i_0})$. Then x'_{i_0, j_0} is a lower component and is equal to a lower component x'_j of X .

If $j_0 \leq 2$, then we consider two subcases. If $i_0 = 1$ (hence $\varepsilon_{i_0} = \varepsilon_1$ is the first argument of \uparrow), then $j = j_0 \leq 2$ and x'_j is an opening \downarrow -component of X . If $i_0 \geq 2$, then j_0 must be 2, because X_{i_0-1} and X_{i_0} have to fit together by upper strands. By Lemma 2.4, $x'_{i_0, j_0-1} = x'_{i_0, 1}$ is a double component. Now let us consider the last component of $\nu^+(X_{i_0-1})$, $x'_{i_0-1, k_{i_0-1}}$. Again because X_{i_0-1} and X_{i_0} have to fit together by upper strands, $x'_{i_0-1, k_{i_0-1}}$ must be an upper component or a double component. In the former case, $y_{i_0-1} = \lambda$ and also x'_{j-2} is an upper component (ending with $x'_{i_0-1, k_{i_0-1}}$, but possibly longer); in the latter case, a lower nick letter is added by the application of \uparrow : $y_{i_0-1} = \triangle = x'_{j-2}$. In both cases, x'_j is an opening \downarrow -component of X .

If $j_0 \geq 3$, then by definition x'_{i_0, j_0-2} is an upper component or a lower nick letter. Then obviously, also x'_{j-2} is an upper component (possibly longer than x'_{i_0, j_0-2}) or a lower nick letter, respectively. Again x'_j is an opening \downarrow -component of X .

Hence, for each opening \downarrow -component of an $\nu^+(X_{i_0})$ (a lower component), the corresponding lower component of X is an opening \downarrow -component.

Because the correspondence between the lower components of the $\nu^+(X_i)$'s and the lower components of X is 1–1, we have thus proved (4.13). Now, (4.2) and (4.3) follow from Lemma 4.13(2). and Lemma 4.13(3), respectively.

- Because of the 1–1 correspondence between the double components of the $\nu^+(X_i)$'s and the double components of X ,

$$n_{\uparrow}(X) = n_{\uparrow}(\nu^+(X_1)) + \dots + n_{\uparrow}(\nu^+(X_n)).$$

Now, (4.4) follows from Lemma 4.13(4).

- Because, obviously, for each formal DNA molecule X_i , $\#_{\nabla}(X_i) \geq 0$. inequality (4.5) follows from (4.4).

2. The proof of this claim is analogous to that of the previous claim.
3. Let E be a \downarrow -expression $\langle \downarrow E_1 \rangle$ for a DNA expression E_1 with $X_1 = \mathcal{S}(E_1)$. By definition, $X = \kappa(X_1)$. Now, equations (4.6)–(4.9) are just special cases of the claims from Lemma 4.15.

□

At several places in the proof of Lemma 4.16(1), we used Lemma 4.13. In fact, for *expressible* formal DNA molecules X , Claims 1 – 4 of Lemma 4.13 are special cases of equations (4.1) – (4.4). If we take $n = 1$ and let ε_1 be a DNA expression denoting X , then the first four claims of Lemma 4.13 follow from the observation that $\mathcal{S}(\langle \uparrow \varepsilon_1 \rangle) = \nu^+(\mathcal{S}^+(\varepsilon_1)) = \nu^+(X)$.

In the proof of Lemma 4.16(3), we observed that equations (4.6)–(4.9) are special cases of the four claims from Lemma 4.15. For *expressible* formal DNA molecules X , we can also walk the opposite direction; when we substitute for E_1 in Lemma 4.16(3) a DNA expression denoting X , we obtain Lemma 4.15.

By inequalities (4.6) and (4.7) from Lemma 4.16(3), the values of the functions T_{\uparrow} and T_{\downarrow} do not increase when we apply the operator \downarrow to a DNA expression E_1 . There exists, however, a much stronger result concerning T_{\uparrow} and T_{\downarrow} for \downarrow -expressions:

Lemma 4.17 *Let E be a \updownarrow -expression, and let $X = \mathcal{S}(E)$. Then $T_{\uparrow}(X) + T_{\downarrow}(X) \leq 1$.*

Proof: Let $E = \langle \updownarrow \varepsilon_1 \rangle$, where ε_1 is an \mathcal{N} -word or a DNA expression. By the definition of the semantics of a \updownarrow -expression, $X = \kappa(\mathcal{S}^+(\varepsilon_1))$. In particular, X does not contain upper components, nor lower components.

Hence, each opening \uparrow -component of X has to be a lower nick letter and each opening \downarrow -component of X has to be an upper nick letter. By Theorem 3.4, X does not both contain lower nick letters and upper nick letters. This implies that either $T_{\uparrow}(X) = 0$, or $T_{\downarrow}(X) = 0$ (or both). Now, the claim follows from Lemma 4.11(1) and (2). \square

After all this introductory work, we are ready to calculate lower bounds for the number of occurrences of the operators \uparrow and \downarrow and for the number of occurrences of the operator \updownarrow in a DNA expression.

Theorem 4.18 *Let E be a DNA expression, and let $X = \mathcal{S}(E)$.*

1. *If E is a \uparrow -expression, then*

$$\begin{aligned} \#\updownarrow(E) &\geq 1 + T_{\downarrow}(X) \quad \text{and} \\ \#\uparrow(E) &\geq n_{\uparrow}(X). \end{aligned}$$

2. *If E is a \downarrow -expression, then*

$$\begin{aligned} \#\updownarrow(E) &\geq 1 + T_{\uparrow}(X) \quad \text{and} \\ \#\uparrow(E) &\geq n_{\uparrow}(X). \end{aligned}$$

3. *If E is a \updownarrow -expression, then*

$$\begin{aligned} \#\updownarrow(E) &\geq T_{\uparrow}(X), \\ \#\updownarrow(E) &\geq T_{\downarrow}(X) \quad \text{and} \\ \#\uparrow(E) &\geq n_{\uparrow}(X). \end{aligned} \tag{4.14}$$

Proof: By induction on the number p of operators occurring in E .

- If $p = 1$, then E is $\langle \uparrow \alpha_1 \rangle$, $\langle \downarrow \alpha_1 \rangle$ or $\langle \updownarrow \alpha_1 \rangle$ for an \mathcal{N} -word α_1 .

If $E = \langle \uparrow \alpha_1 \rangle$, then $\#\updownarrow(E) = 1$, $\#\uparrow(E) = 0$, $X = \binom{\alpha_1}{-}$ and $T_{\downarrow}(X) = n_{\downarrow}(X) = 0$. Hence, the inequalities in Claim 1 are valid.

If $E = \langle \downarrow \alpha_1 \rangle$, then Claim 2 is applicable and the inequalities in this claim are verified analogously.

If $E = \langle \updownarrow \alpha_1 \rangle$, then $\#\updownarrow(E) = 0$, $\#\uparrow(E) = 1$, $X = \binom{\alpha_1}{c(\alpha_1)}$, $T_{\uparrow}(X) = T_{\downarrow}(X) = 0$ and $n_{\uparrow}(X) = 1$. Indeed, these values satisfy the inequalities in Claim 3.

- Let $p \geq 1$, and suppose that the lower bounds have been proved to hold for all DNA expressions containing at most p operators (induction hypothesis). Now let E be an arbitrary DNA expression that contains $p + 1$ operators. E is either a \uparrow -expression, or a \downarrow -expression or a \updownarrow -expression. We consider each of these cases separately.

— If E is a \uparrow -expression $\langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$, where $n \geq 1$ and $\varepsilon_1, \dots, \varepsilon_n$ are the arguments of E , then let for $i = 1, \dots, n$, $X_i = \mathcal{S}^+(\varepsilon_i)$. The arguments are \mathcal{N} -words, \uparrow -expressions, \downarrow -expressions and \updownarrow -expressions.

By definition, if an argument ε_i is an \mathcal{N} -word α , then $\#_{\uparrow, \downarrow}(\varepsilon_i) = \#_{\updownarrow}(\varepsilon_i) = 0$, $X_i = \mathcal{S}^+(\varepsilon_i) = \binom{\alpha}{-}$ and $T_{\downarrow}(X_i) = n_{\updownarrow}(X_i) = 0$. If, on the other hand, an argument ε_i is a DNA expression, then $X_i = \mathcal{S}^+(\varepsilon_i) = \mathcal{S}(\varepsilon_i)$. Because such an argument contains at most p operators, the induction hypothesis provides us with lower bounds for $\#_{\uparrow, \downarrow}(\varepsilon_i)$ and $\#_{\updownarrow}(\varepsilon_i)$. For \updownarrow -expressions ε_i , we use lower bound (4.14) for $\#_{\uparrow, \downarrow}(\varepsilon_i)$.

We first consider $\#_{\uparrow, \downarrow}(E)$:

$$\begin{aligned}
\#_{\uparrow, \downarrow}(E) &= 1 + \sum_{i=1}^n \#_{\uparrow, \downarrow}(\varepsilon_i) \\
&= 1 + \sum_{\mathcal{N}\text{-words } \varepsilon_i} \#_{\uparrow, \downarrow}(\varepsilon_i) + \sum_{\uparrow\text{-expr. } \varepsilon_i} \#_{\uparrow, \downarrow}(\varepsilon_i) \\
&\quad + \sum_{\downarrow\text{-expr. } \varepsilon_i} \#_{\uparrow, \downarrow}(\varepsilon_i) + \sum_{\updownarrow\text{-expr. } \varepsilon_i} \#_{\uparrow, \downarrow}(\varepsilon_i) \\
&\geq 1 + \sum_{\mathcal{N}\text{-words } \varepsilon_i} T_{\downarrow}(X_i) + \sum_{\uparrow\text{-expr. } \varepsilon_i} (1 + T_{\downarrow}(X_i)) \\
&\quad + \sum_{\downarrow\text{-expr. } \varepsilon_i} (1 + T_{\uparrow}(X_i)) + \sum_{\updownarrow\text{-expr. } \varepsilon_i} T_{\downarrow}(X_i).
\end{aligned}$$

Obviously, the term $1 + T_{\downarrow}(X_i)$ for a \uparrow -expression ε_i is greater than $T_{\downarrow}(X_i)$. For the \downarrow -expressions ε_i , we use Lemma 4.11(2) to replace the terms $1 + T_{\uparrow}(X_i)$ by $T_{\downarrow}(X_i)$. When subsequently, we apply inequality (4.2) from Lemma 4.16(1), we obtain the desired lower bound for $\#_{\uparrow, \downarrow}(E)$:

$$\#_{\uparrow, \downarrow}(E) \geq 1 + \sum_{i=1}^n T_{\downarrow}(X_i) \geq 1 + T_{\downarrow}(X).$$

The lower bound for $\#_{\updownarrow}(E)$ is easy to calculate. First, we observe that for each argument ε_i , either by definition (if ε_i is an \mathcal{N} -word), or by the induction hypothesis (if ε_i is a DNA expression), $\#_{\updownarrow}(\varepsilon_i) \geq n_{\updownarrow}(X_i)$. Next, we apply inequality (4.5) from Lemma 4.16(1):

$$\#_{\updownarrow}(E) = \sum_{i=1}^n \#_{\updownarrow}(\varepsilon_i) \geq \sum_{i=1}^n n_{\updownarrow}(X_i) \geq n_{\updownarrow}(X).$$

- If E is a \downarrow -expression, then the proof is analogous.
- If E is a \updownarrow -expression, then its only argument must be a DNA expression E_1 : $E = \langle \updownarrow E_1 \rangle$. Let $X_1 = \mathcal{S}(E_1)$. Because E_1 contains p operators, we can apply the induction hypothesis to it.

If E_1 is a \uparrow -expression, then we additionally apply inequality (4.7) from Lemma 4.16(3) and Lemma 4.11(1):

$$\#_{\uparrow,\downarrow}(E) = \#_{\uparrow,\downarrow}(E_1) \geq 1 + T_{\downarrow}(X_1) \geq 1 + T_{\downarrow}(X) \geq T_{\uparrow}(X).$$

Analogously, if E_1 is a \downarrow -expression, then

$$\#_{\uparrow,\downarrow}(E) = \#_{\uparrow,\downarrow}(E_1) \geq 1 + T_{\uparrow}(X_1) \geq 1 + T_{\uparrow}(X) \geq T_{\downarrow}(X).$$

Finally, if E_1 is a \updownarrow -expression, then $X = \kappa(X_1) = X_1$. Hence, by the induction hypothesis,

$$\begin{aligned} \#_{\uparrow,\downarrow}(E) &= \#_{\uparrow,\downarrow}(E_1) \geq T_{\uparrow}(X_1) = T_{\uparrow}(X) \quad \text{and} \\ \#_{\uparrow,\downarrow}(E) &= \#_{\uparrow,\downarrow}(E_1) \geq T_{\downarrow}(X_1) = T_{\downarrow}(X). \end{aligned}$$

To calculate a lower bound for $\#_{\uparrow}(E)$, we do not have to distinguish different cases. By the induction hypothesis, $\#_{\uparrow}(E_1) \geq n_{\uparrow}(X_1)$, regardless of the outermost operator of E_1 . Then by inequality (4.9) from Lemma 4.16(3),

$$\#_{\uparrow}(E) = 1 + \#_{\uparrow}(E_1) \geq 1 + n_{\uparrow}(X_1) \geq n_{\uparrow}(X).$$

□

It is only a small step from the number of operators occurring in a DNA expression to the length of that DNA expression:

Corollary 4.19 *Let E be a DNA expression, and let $X = \mathcal{S}(E)$.*

1. *If E is a \uparrow -expression, then $|E| \geq 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|$.*
2. *If E is a \downarrow -expression, then $|E| \geq 3 + 3 \cdot T_{\uparrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|$.*
3. *If E is a \updownarrow -expression, then*

$$\begin{aligned} |E| &\geq 3 \cdot T_{\uparrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)| \quad \text{and} \\ |E| &\geq 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|. \end{aligned} \tag{4.15}$$

4. *If $E = \langle \updownarrow \alpha_1 \rangle$ for an \mathcal{N} -word α_1 , then $|E| = 3 \cdot n_{\uparrow}(X) + |\nu(X)|$.*
5. *If $E = \langle \updownarrow E_1 \rangle$ for a DNA expression E_1 , then $|E| \geq 3 + 3 \cdot n_{\uparrow}(X) + |\nu(X)|$.*
6. *Unless $E = \langle \updownarrow \alpha_1 \rangle$ for an \mathcal{N} -word α_1 , $|E| \geq 3 + 3 \cdot n_{\uparrow}(X) + |\nu(X)|$.*

Proof: Claims 1, 2 and 3 follow immediately from Lemma 4.1 and Theorem 4.18.

4. If $E = \langle \updownarrow \alpha_1 \rangle$ for an \mathcal{N} -word α_1 , then $\nu(X) = X = \binom{\alpha_1}{c(\alpha_1)}$ and $n_{\uparrow}(X) = 1$. Hence, both sides of the equality in the claim evaluate to $3 + |\alpha_1|$.

5. Assume that $E = \langle \uparrow E_1 \rangle$ for a DNA expression E_1 , and let $X_1 = \mathcal{S}(E_1)$. By definition, $X = \kappa(X_1)$ and hence $|\nu(X)| = |\nu(X_1)|$. We distinguish three cases.

If E_1 is a \uparrow -expression, then by Claim 1 and inequality (4.9) from Lemma 4.16(3),

$$\begin{aligned} |E| &= 3 + |E_1| \\ &\geq 3 + 3 + 3 \cdot T_1(X_1) + 3 \cdot n_{\uparrow}(X_1) + |\nu(X_1)| \\ &\geq 3 + 3 + 0 + 3 \cdot n_{\uparrow}(X_1) + |\nu(X_1)| \\ &\geq 3 + 3 \cdot n_{\uparrow}(X) + |\nu(X)|. \end{aligned}$$

If E_1 is a \downarrow -expression, then the inequality $|E| \geq 3 + 3 \cdot n_{\uparrow}(X) + |\nu(X)|$ is obtained in an analogous way.

Finally, if E_1 is a \downarrow -expression, then $X = X_1$. Hence, by Claim 3,

$$|E| = 3 + |E_1| \geq 3 + 0 + 3 \cdot n_{\uparrow}(X_1) + |\nu(X_1)| = 3 + 3 \cdot n_{\uparrow}(X) + |\nu(X)|.$$

For each type of DNA expression E_1 , we obtain the inequality from the claim. Hence, the claim is valid.

6. This claim follows immediately from Claims 1, 2 and 5.

□

4.2 Minimal DNA expressions

If a formal DNA molecule is expressible, then there exist (infinitely) many DNA expressions denoting it. We are interested in the one(s) with the smallest length.

Definition 4.20 *A DNA expression E is minimal if for every DNA expression E' with $E' \equiv E$, $|E'| \geq |E|$.*

Hence, when a DNA expression is minimal, we cannot find a shorter DNA expression with the same semantics.

In principle, there may be different minimal DNA expressions for the same formal DNA molecule. For example, both

$$E = \langle \uparrow \alpha_1 \langle \downarrow \langle \uparrow \alpha_2 \rangle \alpha_3 \rangle \rangle \quad \text{and} \quad E' = \langle \downarrow \langle \uparrow \alpha_1 \langle \uparrow \alpha_2 \rangle \rangle \alpha_3 \rangle \quad (4.16)$$

denote $X = \binom{\alpha_1}{-} \binom{\alpha_2}{c(\alpha_2)} \binom{-}{\alpha_3}$, and $|E| = |E'|$. It is easy to verify that E and E' achieve the lower bounds given in Corollary 4.19 for \uparrow -expressions and \downarrow -expressions, respectively. Hence, there do not exist shorter \uparrow -expressions or \downarrow -expressions for X . Because X contains an upper component and a lower component, it cannot be denoted by a \downarrow -expression. Consequently, E and E' are indeed minimal.

The following result is immediately deduced from Lemma 4.1:

Corollary 4.21 *A DNA expression E containing p operators is minimal if and only if every DNA expression E' with $E' \equiv E$ contains at least p operators.*

The next result is so natural that we will not refer to it when we use it. Nevertheless, it is good to state it explicitly:

Lemma 4.22 *A DNA expression E is minimal if and only if each DNA subexpression of E is minimal.*

Proof: Let E be an arbitrary DNA expression.

From right to left, the claim is obvious, because by definition E is a DNA subexpression of itself.

Now suppose that a DNA subexpression E^s of E is not minimal. Then there must exist a DNA expression $E^{s'}$ such that $E^{s'} \equiv E^s$ and $|E^{s'}| < |E^s|$. Let us substitute E^s in E by $E^{s'}$. By Lemma 3.7, the resulting DNA expression E' is equivalent to E . Because $|E'| < |E|$, E cannot be minimal. \square

In this section, we first describe how to construct minimal DNA expressions for nick free formal DNA molecules. We then do the same for expressible formal DNA molecules that do contain nicks. We subsequently demonstrate that each minimal DNA expression satisfies the description we have given, i.e., that there do not exist other minimal DNA expressions. After that, we calculate the number of different minimal DNA expressions for a given expressible formal DNA molecule. We also present a characterization of minimal DNA expressions. By this, we can recognize a minimal DNA expression without determining its length. Finally, we consider the trees that correspond to minimal DNA expressions.

4.2.1 Minimal DNA expressions for a nick free formal DNA molecule

Minimal \updownarrow -expressions are limited to one simple type of formal DNA molecules:

Theorem 4.23 *A \updownarrow -expression E is minimal if and only if $E = \langle \updownarrow \alpha_1 \rangle$ for an \mathcal{N} -word α_1 .*

In that case, E is the unique minimal DNA expression denoting $\mathcal{S}(E) = \binom{\alpha_1}{c(\alpha_1)}$.

Proof: Let $E = \langle \updownarrow \varepsilon_1 \rangle$ be a \updownarrow -expression for an \mathcal{N} -word or DNA expression ε_1 and let $X = \mathcal{S}(E)$. We consider the two possibilities for ε_1 separately.

- If ε_1 is an \mathcal{N} -word α_1 , hence $E = \langle \updownarrow \alpha_1 \rangle$, then $X = \binom{\alpha_1}{c(\alpha_1)}$.

Now let E' be another, equivalent DNA expression. E' is not equal to $\langle \updownarrow \alpha \rangle$ for an \mathcal{N} -word α , because the only \mathcal{N} -word α such that $\mathcal{S}(\langle \updownarrow \alpha \rangle) = \binom{\alpha_1}{c(\alpha_1)}$ is $\alpha = \alpha_1$.

Because $n_{\updownarrow}(X) = 1$, Corollary 4.19(6) implies that E' is longer than E :

$$|E'| \geq 3 + 3 \cdot 1 + |\nu(X)| = 6 + |X| > 3 + |X| = |E|.$$

Thus, E is the unique minimal DNA expression denoting X .

- If $\varepsilon_1 = E_1$ for a DNA expression E_1 , hence $E = \langle \updownarrow E_1 \rangle$, then by definition $X = \kappa(X_1)$ with $X_1 = \mathcal{S}(E_1)$.

Consequently, X does not contain upper components, nor lower components. Because X is expressible, the nick letters occurring in X must be all of the same type: either they are all upper nick letters, or they are all lower nick letters (see

Theorem 3.4). Hence, by Corollary 2.6, there exist \mathcal{N} -words $\alpha_1, \dots, \alpha_k$ for some $k \geq 1$ and a nick letter $y \in \{\nabla, \triangle\}$ such that

$$X = \left(\begin{smallmatrix} \alpha_1 \\ c(\alpha_1) \end{smallmatrix} \right) y \left(\begin{smallmatrix} \alpha_2 \\ c(\alpha_2) \end{smallmatrix} \right) y \cdots y \left(\begin{smallmatrix} \alpha_k \\ c(\alpha_k) \end{smallmatrix} \right).$$

If $k = 1$, hence $X = \left(\begin{smallmatrix} \alpha_1 \\ c(\alpha_1) \end{smallmatrix} \right)$ is nick free, then E is not minimal, because we have just observed that the unique minimal DNA expression denoting $\left(\begin{smallmatrix} \alpha_1 \\ c(\alpha_1) \end{smallmatrix} \right)$ is $\langle \downarrow \alpha_1 \rangle$.

If $k \geq 2$, then E_1 cannot be a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α , because otherwise $\mathcal{S}(\langle \downarrow E_1 \rangle) = \mathcal{S}(\langle \downarrow \langle \downarrow \alpha \rangle \rangle) = \left(\begin{smallmatrix} \alpha \\ c(\alpha) \end{smallmatrix} \right)$ would be nick free. Hence, by Corollary 4.19(6),

$$|E| = 3 + |E_1| \geq 3 + 3 + 3 \cdot n_{\downarrow}(X_1) + |\nu(X_1)| \quad (4.17)$$

The fact that $X = \kappa(X_1)$ implies that $|\nu(X_1)| = |\nu(X)|$ and that $\#_{\nabla, \triangle}(X_1) = \#_{\nabla, \triangle}(X) \geq 1$. Combining the latter observation with Lemma 4.11(4) and Lemma 4.11(3), we find

$$n_{\downarrow}(X_1) \geq \#_{\nabla, \triangle}(X_1) + 1 = \#_{\nabla, \triangle}(X) + 1 = n_{\downarrow}(X).$$

When we substitute everything into (4.17), we obtain:

$$|E| \geq 3 + 3 + 3 \cdot n_{\downarrow}(X) + |\nu(X)| = 6 + 3k + |\nu(X)|.$$

Now without loss of generality, assume that $y = \triangle$ and consider the DNA expression $E' = \langle \uparrow \langle \downarrow \alpha_1 \rangle \langle \downarrow \alpha_2 \rangle \cdots \langle \downarrow \alpha_k \rangle \rangle$. It is easily verified that

$$\mathcal{S}(E') = \left(\begin{smallmatrix} \alpha_1 \\ c(\alpha_1) \end{smallmatrix} \right)_{\triangle} \left(\begin{smallmatrix} \alpha_2 \\ c(\alpha_2) \end{smallmatrix} \right)_{\triangle} \cdots \left(\begin{smallmatrix} \alpha_k \\ c(\alpha_k) \end{smallmatrix} \right)_{\triangle} = X$$

and that

$$|E'| = 3 + 3k + |\nu(X)| < |E|$$

Thus, also in this case, $E = \langle \downarrow E_1 \rangle$ is not minimal. □

Formal DNA molecules of the form $\left(\begin{smallmatrix} \alpha_1 \\ c(\alpha_1) \end{smallmatrix} \right)$ for an \mathcal{N} -word α_1 will come back frequently in the remainder of this section. Often, we are not interested in the actual \mathcal{N} -letters occurring in such a molecule (hence in α_1), but only in the shape of the molecule, for instance when we want to except molecules of this type from a certain statement. In order not to burden the text with unnecessary details, we may speak of a *double-complete* formal DNA molecule, when we mean a formal DNA molecule of the form $\left(\begin{smallmatrix} \alpha_1 \\ c(\alpha_1) \end{smallmatrix} \right)$ for an \mathcal{N} -word α_1 .

The fact that the only minimal \downarrow -expressions are $\langle \downarrow \alpha_1 \rangle$ for \mathcal{N} -words α_1 , implies the following:

Corollary 4.24 *Let X be an expressible formal DNA molecule which is not double-complete. Then each minimal DNA expression denoting X is either a \uparrow -expression or a \downarrow -expression.*

Theorem 4.23 describes the (unique) minimal DNA expression denoting a double-complete formal DNA molecule. This result is quite straightforward. For other nick free formal DNA molecules, the construction of minimal DNA expressions and the corresponding proof of correctness is more involved. We start, however, with a simple observation, which is immediate from Corollary 2.5.

Lemma 4.25 *Let X be a nick free formal DNA molecule. Then X is not double-complete, if and only if X contains at least one single-stranded component.*

In the previous section, we defined the opening and closing \uparrow -components and the opening and closing \downarrow -components of a formal DNA molecule. Here, these notions appear to be useful, again. For a nick free formal DNA molecule, however, each \uparrow -component is an upper component and each \downarrow -component is a lower component. To reflect this in our terminology, we will speak of opening and closing *upper components* instead of opening and closing \uparrow -components, and of opening and closing *lower components* instead of opening and closing \downarrow -components. We will use the new, but equivalent terminology only in the context of nick free formal DNA molecules.

We will find that, in general, there may be different minimal DNA expressions denoting the same nick free formal DNA molecule. Sometimes, there exist minimal DNA expressions for a formal DNA molecule with different outermost operators, \uparrow or \downarrow , as is the case for the example we have seen in (4.16). Moreover, given an outermost operator, there may be different ways to partition a formal DNA molecule into parts that are (intuitively speaking) generated by the outermost operator and parts that are generated by other operators in the DNA expression, at a higher nesting level. Each of these ways may lead to one or (many) more minimal DNA expressions.

In order to determine what a useful partitioning of a certain formal DNA molecule is, we need a number of definitions and auxiliary results.

Definition 4.26 *Let X be a nick free formal DNA molecule and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .*

A maximal upper sequence of X is an occurrence (X_1, X_2) of a non-empty substring Y of X such that $X_1 = x'_1 \dots x'_{a-1}$ and $X_2 = x'_{b+1} \dots x'_k$ for some a and b with $1 \leq a \leq b \leq k$ (hence $Y = x'_a \dots x'_b$), and

- for $i = a, \dots, b$, x'_i is either a double component or an upper component, and
- – either $a = 1$ (hence X_1 is empty),
– or $a \geq 3$ and x'_a is an opening upper component,
and
- – either $b = k$ (hence X_2 is empty),
– or $b \leq k - 2$ and x'_b is a closing upper component.

A maximal upper sequence of a nick free formal DNA molecule X is formally defined as an *occurrence* (X_1, X_2) of a substring Y of X satisfying certain conditions. However, when the occurrence is clear from the context, we will often refer to a maximal upper sequence by the substring Y itself.

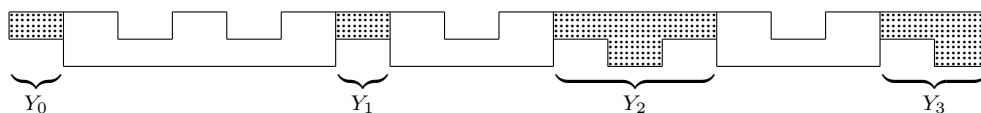


Figure 4.4: The maximal upper sequences of a formal DNA molecule X . Y_0 is the maximal upper prefix and Y_3 is the maximal upper suffix of X .

As an example, Figure 4.4 shows all maximal upper sequences of a certain formal DNA molecule.

As another example, consider $X = \binom{\alpha_1}{c(\alpha_1)}$ for a certain \mathcal{N} -word α_1 . It is easily verified that $Y = X = \binom{\alpha_1}{c(\alpha_1)}$ satisfies all conditions of a maximal upper sequence. Note that in this case, Y does not contain any upper component.

Intuitively, a maximal upper sequence Y of X is ‘maximal’ in the sense that it cannot be extended either to the left or to the right by a ‘block’ \square or by a ‘block’ \square . We make this intuition more formal in the next result.

Lemma 4.27 *Let X be a nick free formal DNA molecule, let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X , and let $Y = x'_a \dots x'_b$ for some a and b with $1 \leq a \leq b \leq k$ be a maximal upper sequence of X .*

1. *If $a \geq 2$, then $a \geq 3$, x'_{a-2} is a closing lower component, x'_{a-1} is a double component and x'_a is an opening upper component.*
2. *If $b \leq k-1$, then $b \leq k-2$, x'_{b+2} is an opening lower component, x'_{b+1} is a double component and x'_b is a closing upper component.*

Proof:

1. Assume that $a \geq 2$. Then by definition $a \geq 3$ and x'_a is an opening upper component. By Corollary 2.5, x'_{a-1} is a double component and by the definition of an opening upper component, x'_{a-2} is a lower component. In particular, x'_{a-2} is a closing lower component.
2. The proof of this claim is analogous to that of the previous claim.

□

As expected, the opening and closing upper components occurring in the definition of a maximal upper sequence are not just arbitrary opening and closing upper components. They are related, as follows:

Lemma 4.28 *Let X be a nick free formal DNA molecule, let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X , and let $Y = x'_a \dots x'_b$ for some a and b with $1 \leq a \leq b \leq k$ be a maximal upper sequence of X .*

If Y contains at least one upper component, then the first upper component occurring in Y is an opening upper component and the last upper component occurring in Y is the corresponding closing upper component.

In particular, each maximal upper sequence $Y = x'_a \dots x'_b$ with $a \geq 2$ and $b \leq k - 1$ starts with an opening upper component and ends with the *corresponding* closing upper component.

Proof: Assume that Y contains at least one upper component. Then ‘the first upper component’ and ‘the last upper component’ occurring in Y are well defined.

If $a \geq 2$, then by the definition of a maximal upper sequence, x'_a is an opening upper component of X . Obviously, x'_a is the first upper component occurring in Y .

If $a = 1$ and x'_a is an upper component, then by definition x'_a is an opening upper component.

If $a = 1$ and x'_a is not an upper component, then x'_a must be a double component, because the maximal upper sequence Y does not contain lower components. Because Y contains at least one upper component, we must have $a < b$, and by Corollary 2.5, $x'_{a+1} = x'_2$ is an upper component. In particular, it is an opening upper component of X .

In every case, the first upper component occurring in Y (either x'_a , or x'_{a+1} with $a = 1$) is an opening upper component. Analogously, we can prove that the last upper component occurring in Y (either x'_b or x'_{b-1}) is a closing upper component. Let us call these opening and closing upper components x'_{a_0} and x'_{b_0} , respectively.

By Corollary 2.5, $a_0 \equiv b_0 \pmod{2}$, and because Y does not contain lower components, each of $x'_{a_0}, x'_{a_0+2}, \dots, x'_{b_0-2}, x'_{b_0}$ is an upper component. Consequently, x'_{a_0} and x'_{b_0} are *corresponding* opening and closing upper components. \square

For the case that X has more than one maximal upper sequence, we have the following result:

Lemma 4.29 *Let X be a nick free formal DNA molecule. Then the maximal upper sequences of X are pairwise disjoint.*

Proof: Let $Y_1 = x'_{a_1} \dots x'_{b_1}$ and $Y_2 = x'_{a_2} \dots x'_{b_2}$ be two different maximal upper sequences of X . Without loss of generality, assume that $a_1 \neq a_2$ (otherwise consider the b_i 's and mirror the arguments). In particular, assume that $a_1 < a_2$. By definition, $a_1 \geq 1$, and thus $a_2 \geq 2$. By Lemma 4.27(1), $a_2 \geq 3$ and x'_{a_2-2} is a lower component.

Suppose that $a_1 = a_2 - 1$ (hence $a_1 - 1 = a_2 - 2$). Then $a_1 \geq 2$, and thus $a_1 \geq 3$ and x'_{a_1-2} would be a lower component. Now, X would have two consecutive lower components x'_{a_1-2} and $x'_{a_1-1} = x'_{a_2-2}$, which would contradict Corollary 2.5.

Consequently, $a_1 \leq a_2 - 2$. Because (the maximal upper sequence) $Y_1 = x'_{a_1} \dots x'_{b_1}$ does not contain lower components, we must have $b_1 < a_2 - 2$. This implies in particular that $b_1 \leq k - 1$. Then by Lemma 4.27(2), $b_1 \leq k - 2$ and x'_{b_1+2} is a lower component. Again, because X cannot have two consecutive lower components, we must have $b_1 + 2 \leq a_2 - 2$.

We thus have $a_1 \leq b_1 < b_1 + 2 \leq a_2 - 2 < a_2 \leq b_2$. Clearly, Y_1 and Y_2 are disjoint. \square

Because of this result, we can unambiguously order the different maximal upper sequences of a nick free formal DNA molecule X according to their occurrence in X . Hence, we can speak of the first, the second, \dots , the last maximal upper sequence of X .

Upper components are important in the definition of a maximal upper sequence. The next result deals with the relation between upper components and maximal upper

sequences.

Lemma 4.30 *Let X be a nick free formal DNA molecule.*

1. *If X is not double-complete, then each maximal upper sequence of X contains at least one upper component.*
2. *Each upper component of X occurs in a (exactly one) maximal upper sequence.*

Proof: Let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .

1. Assume that X is not double-complete. Then by Lemma 4.25, X contains at least one single-stranded component. Let $Y = x'_a \dots x'_b$ with $1 \leq a \leq b \leq k$ be an arbitrary maximal upper sequence of X .

If $a \geq 2$, then by definition, x'_a is an (opening) upper component. Similarly, if $b \leq k - 1$, then x'_b is a (closing) upper component. In particular, in these cases, Y contains at least one upper component.

If $a = 1$ and $b = k$, then $Y = X$, and hence, Y contains at least one single-stranded component. Because Y does not contain lower components, this has to be an upper component.

2. Let x'_{i_0} be an arbitrary upper component of X . By Corollary 2.5, each component x'_i with $i \equiv i_0 \pmod{2}$ is an upper component or a lower component, and each component x'_i with $i \equiv i_0 + 1 \pmod{2}$ is a double component.

Now, let a_0 be the smallest index with $1 \leq a_0 \leq i_0$ and $a_0 \equiv i_0 \pmod{2}$ such that each of $x'_{a_0}, x'_{a_0+2}, \dots, x'_{i_0-2}, x'_{i_0}$ is an upper component. Further, let b_0 be the largest index with $i_0 \leq b_0 \leq k$ and $b_0 \equiv i_0 \pmod{2}$ such that each of $x'_{i_0}, x'_{i_0+2}, \dots, x'_{b_0-2}, x'_{b_0}$ is an upper component. Because x'_{i_0} itself is an upper component, a_0 and b_0 are well defined.

We subsequently define indices a and b by

$$a = \begin{cases} 1 & \text{if } a_0 = 2 \\ a_0 & \text{otherwise} \end{cases} \quad \text{and} \quad b = \begin{cases} k & \text{if } b_0 = k - 1 \\ b_0 & \text{otherwise} \end{cases},$$

and let $Y = x'_a \dots x'_b$. It is easy to see that $1 \leq a \leq a_0 \leq i_0 \leq b_0 \leq b \leq k$. Indeed, Y contains x'_{i_0} .

It follows from the foregoing that for $i = a, \dots, b$, x'_i is an upper component or a double component. Further, either $a = 1$, or $a \geq 3$. In the latter case, $a = a_0$, and hence x'_{a-2} is a lower component and x'_a is an opening upper component. Similarly, either $b = k$, or $b \leq k - 2$. In the latter case, $b = b_0$, and hence x'_{b+2} is a lower component and x'_b is a closing upper component. Consequently, Y is a maximal upper sequence.

Because, by Lemma 4.29, maximal upper sequences are pairwise disjoint, the upper component x'_{i_0} does not occur in any other maximal upper sequence of X .

□

We now examine some specific cases of formal DNA molecules and their maximal upper sequences.

Lemma 4.31 *Let X be a nick free formal DNA molecule and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .*

1. *X contains a maximal upper sequence $Y = x'_i$ which equals a double component x'_i of X , if and only if X is double-complete.*

In that case, $Y = X$ is the only maximal upper sequence of X .

2. *X is a maximal upper sequence of itself, if and only if X does not contain any lower component.*

In that case, $Y = X$ is the only maximal upper sequence of X .

3. *X does not contain any maximal upper sequence, if and only if X does not contain any upper component and contains at least one lower component.*

Proof:

1. \implies A maximal upper sequence Y which equals a double component x'_i of X does not contain any upper component of X . If X contains such a maximal upper sequence, then by Lemma 4.30(1), X must be double-complete.

\Leftarrow If, on the other hand, $X = \binom{\alpha_1}{c(\alpha_1)}$ for an \mathcal{N} -word α_1 , then $k = 1$ and $x'_1 = \binom{\alpha_1}{c(\alpha_1)}$. As we observed before, $Y = x'_1 = \binom{\alpha_1}{c(\alpha_1)} = X$ is a maximal upper sequence. It is, in fact, a special case of the next claim.

Obviously, in this case, $Y = X$ is the *only* maximal upper sequence of X .

2. \implies If X is a maximal upper sequence of itself, then by definition, it only contains double components and upper components. Hence, it does not contain any lower component.

\Leftarrow If, on the other hand, X does not contain any lower component, then, because X is nick free, it only contains double components and upper components. Now, obviously $X = x'_1 \dots x'_k$ satisfies all conditions of a maximal upper sequence.

In this case, $Y = X$ is the only maximal upper sequence of X , because, by Lemma 4.29, maximal upper sequences are pairwise disjoint.

3. \implies Assume that X does not contain any maximal upper sequence.

By Claim 1, X cannot be double-complete, because otherwise X would be a maximal upper sequence itself. Hence, by Lemma 4.25, X contains at least one single-stranded component.

Because, by Lemma 4.30(2), each upper component of X occurs in a maximal upper sequence, X cannot contain any upper component. Consequently, X contains at least one lower component.

\Leftarrow Assume, on the other hand, that X does not contain any upper component and contains at least one lower component.

Because X contains at least one lower component, X is not double-complete. Then by Lemma 4.30(1), each maximal upper sequence of X contains at least one upper component. Now, because X does not contain any upper component, it certainly does not contain a maximal upper sequence. \square

There is a simple relation between the number of maximal upper sequences of a nick free formal DNA molecule X and $T_{\uparrow}(X)$. We only consider the case that X is double-complete separately, as $T_{\uparrow}(X) = 0$ for such a formal DNA molecule.

Lemma 4.32 *Let X be a nick free formal DNA molecule. Then the number of maximal upper sequences of X is equal to*

$$\begin{cases} 1 & \text{if } X \text{ is double-complete,} \\ T_{\uparrow}(X) & \text{otherwise.} \end{cases}$$

Proof: If X is double-complete, then by Lemma 4.31(1), $Y = X$ is the only maximal upper sequence of X .

Now, let us assume that X is not double-complete.

Let $Y = x'_a \dots x'_b$ be an arbitrary maximal upper sequence. By Lemma 4.30(1), Y contains at least one upper component. Further, by Lemma 4.28, the first upper component occurring in Y is an opening upper component of X and the last upper component occurring in Y is the corresponding closing upper component of X . Finally, by Lemma 4.6(1), Y does not contain any other opening upper component.

Hence, each maximal upper sequence contains exactly one opening upper component. On the other hand, by Lemma 4.30(2), each opening upper component occurs in a (exactly one) maximal upper sequence. This implies that the relation between opening upper components and maximal upper sequences that is induced by occurrence is bijective.

Consequently, the number of opening upper components ($T_{\uparrow}(X)$) is equal to the number of maximal upper sequences. \square

In addition to maximal upper sequences, we have separating lower sequences.

Definition 4.33 *Let X be a nick free formal DNA molecule and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .*

A separating lower sequence of X is an occurrence (Y_1, Y_2) of a non-empty substring X_1 of X such that $Y_1 = x'_1 \dots x'_b$ and $Y_2 = x'_a \dots x'_k$ for some b and a with $1 \leq b + 1 \leq a - 1 \leq k$ (hence $X_1 = x'_{b+1} \dots x'_{a-1}$), and

- *for $i = b + 1, \dots, a - 1$, x'_i is either a double component or a lower component, and there is at least one i with $b + 1 \leq i \leq a - 1$ for which x'_i is a lower component, and*
- *– either $b + 1 = 1$ (hence Y_1 is empty),*
– or $b + 1 \geq 2$ and x'_b is an upper component,
and
- *– either $a - 1 = k$ (hence Y_2 is empty),*
– or $a - 1 \leq k - 1$ and x'_a is an upper component.

For the formal DNA molecule from Figure 4.4, we have indicated the separating lower sequences in Figure 4.5. It is easy to see that in this example, each pair of consecutive maximal upper sequences are separated by a separating lower sequence. We will prove that this holds in general. This, of course, explains the adjective *separating* in the term *separating lower sequence*.

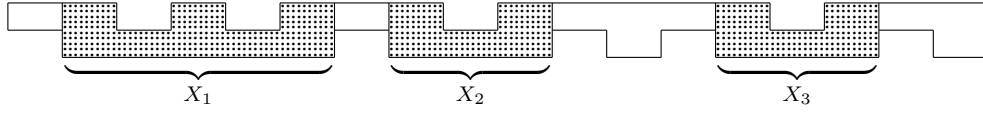


Figure 4.5: The separating lower sequences of the formal DNA molecule X from Figure 4.4.

Note that a separating lower sequence is not just the ‘lower analogue’ of a maximal upper sequence. A separating lower sequence of a nick free formal DNA molecule X has to contain at least one lower component of X , whereas a maximal upper sequence does not necessarily contain an upper component. Further, if the first component x'_{b+1} of a separating lower sequence is not the first component x'_1 of X , then it is a double component (because x'_b is an upper component). If, on the other hand, the first component of a maximal upper sequence is not the first component of X , then it is an (opening) upper component. There is an analogous difference between the last component of a separating lower sequence and the last component of a maximal upper sequence.

The question which type of components we find at the edges of a separating lower sequence is dealt with in the following result.

Lemma 4.34 *Let X be a nick free formal DNA molecule, let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X , and let $X_1 = x'_{b+1} \dots x'_{a-1}$ for some b and a with $1 \leq b+1 \leq a-1 \leq k$ be a separating lower sequence of X .*

1. *If $b+1 \geq 2$, then $b+1 < a-1 \leq k$, x'_b is a closing upper component, x'_{b+1} is a double component and x'_{b+2} is an opening lower component.*
2. *If $a-1 \leq k-1$, then $1 \leq b+1 < a-1$, x'_a is an opening upper component, x'_{a-1} is a double component and x'_{a-2} is a closing lower component.*

Proof:

1. Assume that $b+1 \geq 2$. Then, by definition, x'_b is an upper component, and by Corollary 2.5, x'_{b+1} is a double component. Because X_1 contains at least one lower component, we must have $b+1 < a-1$. Hence, x'_{b+2} is part of X_1 , and by the definition of a separating lower sequence, it is a double component or a lower component. By Corollary 2.5, it has to be a lower component. Indeed, x'_b is a closing upper component and x'_{b+2} is an opening lower component.
2. The proof of this claim is analogous to that of the previous claim.

□

The above result is the ‘separating lower sequence-version’ of Lemma 4.27. Such versions also exist for Lemma 4.28 – Lemma 4.32:

Lemma 4.35 (cf. Lemma 4.28) *Let X be a nick free formal DNA molecule, let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X , and let $X_1 = x'_{b+1} \dots x'_{a-1}$ for some b and a with $1 \leq b+1 \leq a-1 \leq k$ be a separating lower sequence of X .*

Then the first lower component occurring in X_1 is an opening lower component and the last lower component occurring in X_1 is the corresponding closing lower component.

Combining this with Lemma 4.34, we find that each separating lower sequence $X_1 = x'_{b+1} \dots x'_{a-1}$ with $b+1 \geq 2$ and $a-1 \leq k-1$ starts with a double component followed by an opening lower component, and ends with the *corresponding* closing lower component followed by another double component.

Proof: The proof of this result is similar to that of Lemma 4.28. However, for the case $b+1 \geq 2$ (and analogously, for the case $a-1 \leq k-1$), we use Lemma 4.34 rather than the definition of a separating lower sequence to argue that x'_{b+2} (x'_{a-2}) is an opening (closing) lower component. \square

Lemma 4.36 (cf. Lemma 4.29) *Let X be a nick free formal DNA molecule. Then the separating lower sequences of X are pairwise disjoint.*

Proof: Let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X , and let $X_1 = x'_{b_1+1} \dots x'_{a_1-1}$ and $X_2 = x'_{b_2+1} \dots x'_{a_2-1}$ be two different separating lower sequences. Without loss of generality, assume that $b_1 \neq b_2$, and in particular that $b_1 < b_2$.

Because $1 \leq b_1 + 1 < b_2 + 1$, we have $b_2 + 1 \geq 2$ and thus, by definition, x'_{b_2} is an upper component. Further, because $b_1 + 1 \leq b_2$, we must have $a_1 - 1 < b_2$, because otherwise (the separating lower sequence) X_1 would contain the upper component x'_{b_2} .

Consequently, $b_1 + 1 \leq a_1 - 1 < b_2 < b_2 + 1 \leq a_2 - 1$, and thus X_1 and X_2 are disjoint. \square

Lemma 4.37 (cf. Lemma 4.30) *Let X be a nick free formal DNA molecule. Then each lower component of X occurs in a (exactly one) separating lower sequence.*

Proof: The proof of this result is similar to that of Lemma 4.30(2). However, because at crucial positions the technical details are different, we give the complete proof.

Let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X , and let x'_{i_0} for some i_0 with $1 \leq i_0 \leq k$ be a lower component of X .

By Corollary 2.5, each component x'_i with $i \equiv i_0 \pmod{2}$ is an upper component or a lower component, and each component x'_i with $i \equiv i_0 + 1 \pmod{2}$ is a double component.

Now let b_0 be the smallest value such that each of $x'_{b_0+2}, x'_{b_0+4}, \dots, x'_{i_0-2}, x'_{i_0}$ is a lower component, and let a_0 be the largest value such that each of $x'_{i_0}, x'_{i_0+2}, \dots, x'_{a_0-4}, x'_{a_0-2}$ is a lower component. Because x'_{i_0} is a lower component, b_0 and a_0 are well defined, and $-1 \leq b_0 \leq i_0 - 2 < i_0 + 2 \leq a_0 \leq k + 2$.

We subsequently define indices b and a by

$$b = \begin{cases} 0 & \text{if } b_0 = -1 \\ b_0 & \text{otherwise} \end{cases} \quad \text{and} \quad a = \begin{cases} k + 1 & \text{if } a_0 = k + 2 \\ a_0 & \text{otherwise,} \end{cases}$$

and let $X_1 = x'_{b+1} \dots x'_{a-1}$.

Because $-1 \leq b_0$ and $a_0 \leq k+2$, we have $0 \leq b$ and $a \leq k+1$ and thus $1 \leq b+1$ and $a-1 \leq k$. Further, because $b_0 \leq i_0 - 2 < i_0 + 2 \leq a_0$, we have $b \leq i_0 - 1 < i_0 + 1 \leq a$ and thus $b+1 \leq i_0 \leq a-1$. Hence, X_1 is well defined and non-empty. It follows from the foregoing that each of $x'_{b+1}, \dots, x'_{a-1}$ is a double component or a lower component. At least one of them is a lower component, viz. x'_{i_0} .

Now, either $b+1 = 1$, or $b+1 \geq 2$. In the latter case, $b = b_0$ and $x'_b = x'_{b_0}$ is an upper component. Similarly, either $a-1 = k$, or $a-1 \leq k-1$. In the latter case,

$a = a_0$ and $x'_a = x'_{a_0}$ is an upper component. Because X_1 has all properties required in the definition, we conclude that it is a separating lower sequence.

Hence, x'_{i_0} occurs in the separating lower sequence X_1 . Because, by Lemma 4.36, separating lower sequences are pairwise disjoint, x'_{i_0} does not occur in any other separating lower sequence of X . \square

Lemma 4.38 (cf. Lemma 4.31) *Let X be a nick free formal DNA molecule.*

1. *X is a separating lower sequence of itself, if and only if X does not contain any upper component and contains at least one lower component.*

In that case, $X_1 = X$ is the only separating lower sequence of X .

2. *X does not contain any separating lower sequence, if and only if X does not contain any lower component.*

Proof: Let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .

1. The proof of this claim is similar to that of Lemma 4.31(2).

2. The proof of this claim is even more direct than that of Lemma 4.31(3), because we do not have to exclude the case that X is double-complete. The claim follows immediately from the definition and from Lemma 4.37. \square

Lemma 4.39 (cf. Lemma 4.32) *Let X be a nick free formal DNA molecule. Then the number of separating lower sequences of X is equal to $T_1(X)$.*

Proof: This proof is similar to that of Lemma 4.32. However, we do not have to consider the case that X is double-complete separately. Each separating lower sequence of X contains exactly one opening lower component and each opening lower component of X occurs in exactly one separating lower sequence. \square

Now, we have come to the relation between maximal upper sequences and separating lower sequences:

Lemma 4.40 *Let X be a nick free formal DNA molecule.*

1. *For each maximal upper sequence Y_1 of X and each separating lower sequence X_2 of X , Y_1 and X_2 are disjoint.*

2. *Each pair of consecutive maximal upper sequences are separated by a separating lower sequence.*

3. *Assume that X contains at least one maximal upper sequence. Let Y_1 be the first maximal upper sequence of X , with occurrence (X_1, X_2) , and let Y_2 be the last maximal upper sequence of X , with occurrence (Z_1, Z_2) .*

(a) *If $X_1 \neq \lambda$, then X_1 is a separating lower sequence.*

(b) *If $Z_2 \neq \lambda$, then Z_2 is a separating lower sequence.*

Proof: Let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .

1. Let $Y_1 = x'_{a_1} \dots x'_{b_1}$ be a maximal upper sequence, and let $X_2 = x'_{b_2+1} \dots x'_{a_2-1}$ be a separating lower sequence of X . Then Y_1 and X_2 must be different, because X_2 contains at least one lower component, whereas Y_1 does not contain any lower component at all. Without loss of generality, assume that $a_1 \neq b_2 + 1$.

If $a_1 < b_2 + 1$, then $b_2 + 1 \geq 2$ and thus, by Lemma 4.34(1), $b_2 + 1 < a_2 - 1 \leq k$, x'_{b_2} is an upper component, x'_{b_2+1} is a double component and x'_{b_2+2} is a lower component. Because Y_1 does not contain any lower component, we must have $b_1 < b_2 + 2$. By Lemma 4.27(2), x'_{b_1} is an upper component. Finally, because X_2 does not contain any upper component, b_1 cannot be equal to $b_2 + 1$. Hence, $b_1 < b_2 + 1$.

If, on the other hand, $b_2 + 1 < a_1$, then $a_1 \geq 2$ and thus, by Lemma 4.27(1), $a_1 \geq 3$, x'_{a_1-2} is a lower component and x'_{a_1} is an upper component. Because X_2 does not contain any upper component, we must have $a_2 - 1 < a_1$.

In both cases, we find that Y_1 and X_2 are disjoint.

2. Let $Y_1 = x'_{a_1} \dots x'_{b_1}$ and $Y_2 = x'_{a_2} \dots x'_{b_2}$ be two consecutive maximal upper sequences. Without loss of generality, assume that Y_1 precedes Y_2 . In the proof of Lemma 4.29, we have established that $b_1 < b_1 + 2 \leq a_2 - 2 < a_2$ (hence, $a_2 - b_1 \geq 4$). By Lemma 4.27, x'_{b_1+2} and x'_{a_2-2} are lower components, x'_{b_1+1} and x'_{a_2-1} are double components and x'_{b_1} and x'_{a_2} are upper components of X .

By Lemma 4.30(2), each upper component occurs in a maximal upper sequence of X . Thus, the existence of an upper component between Y_1 and Y_2 would contradict the assumption that they are consecutive. Hence, the sequence $x'_{b_1+1} \dots x'_{a_2-1}$ only consists of double components and lower components.

This implies that $x'_{b_1+1} \dots x'_{a_2-1}$ satisfies all conditions of a separating lower sequence.

3. The proof of this claim is similar to that of the previous claim.

For the first subclaim, we first establish that the last two components of X_1 are a lower component and a double component, respectively, and that the first component of Y_1 is an upper component. We subsequently prove that X_1 does not contain any upper component, because otherwise we could find a maximal upper sequence within X_1 , i.e., before the *first* maximal upper sequence. These properties together imply that X_1 is a separating lower sequence.

The proof of the second subclaim is completely analogous.

□

When we combine Lemma 4.31(3) and Lemma 4.38(1) with Lemma 4.29, Lemma 4.36 and Lemma 4.40, we obtain:

Theorem 4.41 *Each nick free formal DNA molecule is an alternating sequence of (all its) maximal upper sequences and (all its) separating lower sequences.*

When we take a subsequence X^s of the maximal upper sequences and separating lower sequences of a nick free formal DNA molecule X , we can, in turn, consider the maximal upper sequences and separating lower sequences of X^s :

Lemma 4.42 *Let X be a nick free formal DNA molecule, and consider X as an alternating sequence of maximal upper sequences and separating lower sequences (see Theorem 4.41). Let X^s be a non-empty subsequence of consecutive maximal upper sequences and separating lower sequences of X .*

Then the maximal upper sequences of X^s are exactly the maximal upper sequences of X occurring in the definition of X^s . The separating lower sequences of X^s are exactly the separating lower sequences of X occurring in the definition of X^s .

Proof: Because X is nick free and X^s is a non-empty substring of X , by Lemma 2.2, X^s is a nick free formal DNA molecule. This implies that indeed maximal upper sequences and separating lower sequences are defined for X^s .

First, we consider a special case. If X is double-complete, then by Lemma 4.31(1), it only consists of one maximal upper sequence, which is equal to X . Hence, X^s , which is a non-empty subsequence of the maximal upper sequences and separating lower sequences of X , must be equal to X . Then the claims are trivially valid.

Conversely, if X^s is double-complete, then it cannot contain any separating lower sequence of X , because by definition, a separating lower sequence contains at least one lower component. Hence, X^s must be equal to one maximal upper sequence of X . Apparently, X contains a maximal upper sequence that is double-complete. By Lemma 4.31(1), X must be double-complete itself, and X must be equal to this double-complete maximal upper sequence. Again, we conclude that $X^s = X$ and that the claims are trivially valid.

Now, let us assume that X is not double-complete and (thus) that X^s is not double-complete. Let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X . Because by definition maximal upper sequences and separating lower sequences of X are built up of components of X , so is X^s . Hence, there exist i_0 and j_0 with $1 \leq i_0 \leq j_0 \leq k$ such that $X^s = x'_{i_0} \dots x'_{j_0}$. Clearly, each component of X^s is also a component of X . The remainder of the proof consists of four steps.

1. Let $Y^s = x'_a \dots x'_b$ with $i_0 \leq a \leq b \leq j_0$ be a maximal upper sequence of X^s . By Lemma 4.30(1), Y^s contains at least one upper component. By definition, Y^s does not contain any lower component.

If $a = i_0$, then the first single-stranded component of X^s is an upper component. Because by definition, a separating lower sequence of X does not contain any upper component and contains at least one lower component, X^s (seen as a subsequence of maximal upper sequences and separating lower sequences of X) starts with a maximal upper sequence. In particular, $x'_a = x'_{i_0}$ is the first component of a maximal upper sequence Y_1 of X .

If $a \geq i_0 + 1$, then by Lemma 4.27(1), $a \geq i_0 + 2$, x'_{a-2} is a lower component, x'_{a-1} is a double component and x'_a is an upper component of X^s (and thus of X). By Lemma 4.30(2), x'_a occurs in a maximal upper sequence Y_1 of X . By definition, (the lower component) x'_{a-2} is not part of Y_1 . Hence, the first component of Y_1 must be either (the double component) x'_{a-1} or (the upper component) x'_a . By Lemma 4.27(1), it must be x'_a . Also in this case, x'_a is the first component of a maximal upper sequence Y_1 of X .

Analogously, we can prove that x'_b is the last component of a maximal upper sequence Y_2 of X .

Now we observe that (by Theorem 4.41) different maximal upper sequences of X are separated by at least one separating lower sequence, that each separating lower sequence of X contains at least one lower component, and that $Y^s = x'_a \dots x'_b$ does not contain any lower component. Consequently, Y_1 and Y_2 must be the same maximal upper sequence of X : $Y^s = Y_1 = Y_2$ is a maximal upper sequence of X .

2. In a similar way, we can prove that each separating lower sequence of X^s is also a separating lower sequence of X . References to results about maximal upper sequences must, of course, be replaced by references to the ‘separating lower sequence versions’ and vice versa.
3. Let Y be a maximal upper sequence of X which occurs in the definition of X^s (as sequence of consecutive maximal upper sequences and separating lower sequences of X).

Y cannot intersect with a separating lower sequence X_1^s of X^s , because, as we have established in step 2, X_1^s would also be a separating lower sequence of Y . Then X would have a maximal upper sequence Y and a separating lower sequence X_1^s that are not disjoint, which would contradict Lemma 4.40(1).

By Theorem 4.41, X^s is an alternating sequence of maximal upper sequences and separating lower sequences (of itself). Hence, Y must be contained in a maximal upper sequence Y^s of X^s . As we have proved in step 1, Y^s is also a maximal upper sequence of X .

If Y were not equal to Y^s , then X would have different maximal upper sequences Y and Y^s that are not disjoint, which would contradict Lemma 4.29. Consequently, Y is equal to Y^s .

4. Analogously, we can prove that each separating lower sequence of X occurring in the definition of X^s (as sequence of consecutive maximal upper sequences and separating lower sequences of X) is also a separating lower sequence of X^s .

□

The first and the last maximal upper sequence occurring in a nick free formal DNA molecule may be of special interest, in particular when they lie at the borders of the molecule.

Definition 4.43 *Let X be a nick free formal DNA molecule and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .*

- *The maximal upper prefix of X is*
 - *the occurrence (λ, X) of the empty string λ if*
 - * *either X does not contain any maximal upper sequence,*
 - * *or X contains at least one maximal upper sequence, and the first maximal upper sequence of X has occurrence (X_1, X_2) for a non-empty prefix X_1 and a suffix X_2 of X .*
 - *the first maximal upper sequence of X otherwise, i.e., if X contains at least one maximal upper sequence and the first maximal upper sequence of X has occurrence (λ, X_2) for a suffix X_2 of X .*

- The maximal upper suffix of X is
 - the occurrence (X, λ) of the empty string λ if
 - * either X does not contain any maximal upper sequence,
 - * or X contains at least one maximal upper sequence, and the last maximal upper sequence of X has occurrence (X_1, X_2) for a prefix X_1 and a non-empty suffix X_2 of X .
 - the last maximal upper sequence of X otherwise, i.e., if X contains at least one maximal upper sequence and the last maximal upper sequence of X has occurrence (X_1, λ) for a prefix X_1 of X .

Again, although formally the maximal upper prefix and the maximal upper suffix of a nick free formal DNA molecule X are defined as *occurrences* of substrings of X , we will often refer to them by the substrings themselves. Implicitly, however, we keep associating to them a position in X . For example, if both the maximal upper prefix and the maximal upper suffix of X are equal to λ , then they are *not* equal, because the occurrence of the maximal upper prefix is (λ, X) and the occurrence of the maximal upper suffix is (X, λ) .

In Figure 4.4, the maximal upper prefix and the maximal upper suffix of the formal DNA molecule depicted are also indicated.

Sometimes, the maximal upper prefix and the maximal upper suffix of a nick free formal DNA molecule *are* equal:

Lemma 4.44 *Let X be a nick free formal DNA molecule. Then the maximal upper prefix of X is equal to the maximal upper suffix of X , if and only if X does not contain any lower component.*

In that case, the maximal upper prefix and the maximal upper suffix are equal to X .

Proof: Let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .

If the maximal upper prefix and the maximal upper suffix are equal, then, as we have just mentioned, they are not empty. Hence, by definition, the maximal upper prefix is the first maximal upper sequence $Y_0 = x'_{a_0} \dots x'_{b_0}$ of X , with occurrence (λ, X_2) for a suffix X_2 of X . Clearly, $a_0 = 1$ for this maximal upper sequence, and $1 \leq b_0 \leq k$. Analogously, the maximal upper suffix is the last maximal upper sequence $Y_0 = x'_{a_1} \dots x'_{b_1}$ of X , with $1 \leq a_1 \leq k$ and $b_1 = k$. It follows that $Y_0 = Y_1 = x'_1 \dots x'_k = X$. In particular, X is a maximal upper sequence of itself.

If, on the other hand, X is a maximal upper sequence of itself, then it is both the first and the last maximal upper sequence of itself. In that case, by definition, both the maximal upper prefix and the maximal upper suffix are equal to X . In particular, they are equal to each other.

Now, the claim follows from Lemma 4.31(2). □

By definition, there are two cases in which the maximal upper prefix or the maximal upper suffix of a nick free formal DNA molecule are empty. To sharpen the intuition, we give two other characterizations of this situation.

Lemma 4.45 *Let X be a nick free formal DNA molecule, and consider X as an alternating sequence of maximal upper sequences and separating lower sequences (see Theorem 4.41).*

1. The following three statements are equivalent:

- (a) The maximal upper prefix of X is empty.
- (b) The alternating sequence starts with a separating lower sequence.
- (c) X contains at least one single-stranded component and the first single-stranded component of X is a lower component.

2. The following three statements are equivalent:

- (a) The maximal upper suffix of X is empty.
- (b) The alternating sequence ends with a separating lower sequence.
- (c) X contains at least one single-stranded component and the last single-stranded component of X is a lower component.

Proof:

1. (1a) \implies (1b) Assume that the maximal upper prefix of X is empty.

Then either X does not contain any maximal upper sequence, or X contains at least one maximal upper sequence but the first maximal upper sequence has occurrence (X_1, X_2) for a non-empty prefix X_1 and a suffix X_2 of X . In the former case, the alternating sequence only consists of a separating lower sequence. In the latter case, by Lemma 4.40(3a), X_1 is a separating lower sequence. In both cases, the alternating sequence starts with a separating lower sequence.

(1b) \implies (1c) This transition is clear, because by definition, a separating lower sequence does not contain any upper component and contains at least one lower component.

(1c) \implies (1a) Assume that X contains at least one single-stranded component and that the first single-stranded component is a lower component.

By Lemma 4.30(1), each maximal upper sequence of X contains at least one upper component. Further, by definition, a maximal upper sequence does not contain any lower component.

If X contains at least one maximal upper sequence, then the first maximal upper sequence cannot be $x'_1 \dots x'_b$ for an index b with $1 \leq b \leq k$. Otherwise, it would not only contain the required upper component, but also the first single-stranded component of X , which is a lower component. Hence, the maximal upper prefix of X is empty.

2. The proof of this claim is analogous to that of the previous claim.

□

Those maximal upper sequences of a nick free formal DNA molecule X which are not equal to the maximal upper prefix, nor to the maximal upper suffix of X are called *internal* maximal upper sequences. We denote the number of internal maximal upper sequences of X by $n_{\text{imus}}(X)$.

Lemma 4.46 *Let X be a nick free formal DNA molecule, and consider X as an alternating sequence of maximal upper sequences and separating lower sequences (see Theorem 4.41).*

1. A maximal upper sequence of X is internal if and only if it is both preceded and succeeded in X by a separating lower sequence.
2. (a) If X does not contain any lower component, then $n_{imus}(X) = 0$.
 (b) If X contains at least one lower component, then $n_{imus}(X) = T_{\downarrow}(X) - 1$.

Proof:

1. If the alternating sequence mentioned starts with a maximal upper sequence, then by definition, this is the maximal upper prefix of X . Analogously, if the alternating sequence ends with a maximal upper sequence, then this is the maximal upper suffix. Consequently, if a maximal upper sequence is internal, then it must be both preceded and succeeded by a separating lower sequence.

Conversely, if a maximal upper sequence is both preceded and succeeded by a separating lower sequence, then it certainly is not a prefix or a suffix of X , and thus it is internal.

2. (a) If X does not contain any lower component, then by Lemma 4.38(2), it does not contain any separating lower sequence. By Claim 1, there cannot be any internal maximal upper sequence, either.
- (b) Assume that X contains at least one lower component. Then because maximal upper sequences do not contain lower components, there is at least one separating lower sequence.

By Claim 1, a maximal upper sequence is internal if and only if it is both preceded and succeeded in X by a separating lower sequence. Hence, the number of internal maximal upper sequences is 1 less than the number of separating lower sequences. By Lemma 4.39, the latter number is $T_{\downarrow}(X)$.

□

We now define partitionings of a nick free formal DNA molecule which will appear to be useful to construct minimal DNA expressions.

Definition 4.47 *Let X be a nick free formal DNA molecule.*

A maximal upper partitioning of X is a sequence $Y_0, X_1, Y_1, X_2, Y_2, \dots, X_r, Y_r$ for some $r \geq 0$ such that

- $X = Y_0 X_1 Y_1 X_2 Y_2 \dots X_r Y_r$, and
- Y_0 is the maximal upper prefix of X and Y_r is the maximal upper suffix of X , and
- for $j = 1, \dots, r - 1$, Y_j is an internal maximal upper sequence of X .

If Y_1, Y_2, \dots, Y_{r-1} are precisely all internal maximal upper sequences of X , then the maximal upper partitioning is called complete.

For notational convenience, we will in general write $Y_0 X_1 Y_1 X_2 Y_2 \dots X_r Y_r$ instead of $Y_0, X_1, Y_1, X_2, Y_2, \dots, X_r, Y_r$ to describe a maximal upper partitioning. In some cases, we will use the symbol \mathcal{M} to refer to a particular maximal upper partitioning.

For each internal maximal upper sequence of a nick free formal DNA molecule X , we can independently decide whether or not to select it for a maximal upper partitioning. We therefore have the following result:

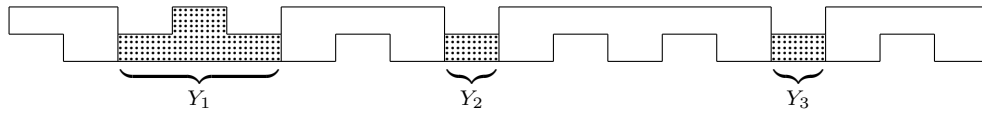


Figure 4.6: The maximal lower sequences of the formal DNA molecule X from Figure 4.5.

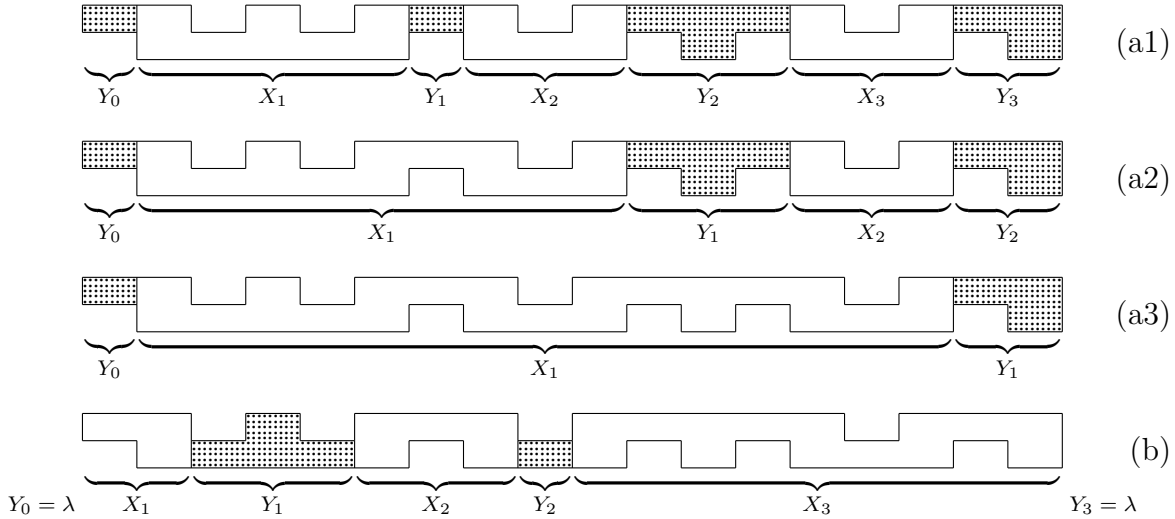


Figure 4.7: Different partitionings of the formal DNA molecule X from Figure 4.4, for which $T_{\uparrow}(X) = 4$ and $T_{\downarrow}(X) = 3$. (a1) The complete maximal upper partitioning of X ; (a2) another maximal upper partitioning of X ; (a3) yet another maximal upper partitioning of X : the one defined by zero internal maximal upper sequences; (b) a maximal lower partitioning of X , which is not complete.

Lemma 4.48 *Let X be a nick free formal DNA molecule. Then the number of different maximal upper partitionings of X is $2^{n_{imus}(X)}$.*

A maximal lower sequence, the maximal lower prefix, the maximal lower suffix, and a maximal lower partitioning of a nick free formal DNA molecule are defined analogously to the upper counterparts. Also, a separating upper sequence is defined analogously to a separating lower sequence. Only the terms ‘upper component’ and ‘lower component’ in the corresponding definitions have to be interchanged.

In Figure 4.6, we have indicated the maximal lower sequences of the formal DNA molecule from Figure 4.4 and Figure 4.5. It illustrates the difference between a separating lower sequence and a maximal lower sequence.

Figure 4.7 shows three maximal upper partitionings and one maximal lower partitioning of the same formal DNA molecule. In this DNA molecule, both the maximal lower prefix and the maximal lower suffix are empty. This is not surprising, due to the next result.

Lemma 4.49 *Let X be a nick free formal DNA molecule and let Y_0 be the maximal upper prefix of X , Y_1 be the maximal upper suffix of X , Y'_0 be the maximal lower prefix of X and Y'_1 be the maximal lower suffix of X .*

1. If X is double-complete, then $Y_0 = Y_1 = Y'_0 = Y'_1 = X$.
2. If X contains at least one single-stranded component, then
 - exactly one of the strings Y_0 and Y'_0 is empty, and
 - exactly one of the strings Y_1 and Y'_1 is empty.

Proof:

1. This claim follows immediately from Lemma 4.44.
2. Assume that X contains at least one single-stranded component.

Without loss of generality, assume that the first single-stranded component is a lower component. By Lemma 4.45(1), $Y_0 = \lambda$. By the analogue of this result for the maximal lower prefix, $Y'_0 \neq \lambda$.

In an analogous way, we prove that exactly one of the strings Y_1 and Y'_1 is empty. □

Sometimes, $\mathcal{M} = X$ is a maximal upper partitioning of X .

Lemma 4.50 *Let X be a nick free formal DNA molecule. Then $\mathcal{M} = X$ is a maximal upper partitioning of X , if and only if X does not contain any lower component.*

In that case $\mathcal{M} = X$ is the only maximal upper partitioning of X .

Proof: If $\mathcal{M} = X$ is a maximal upper partitioning $Y_0X_1Y_1X_2Y_2 \dots X_rY_r$ of X , then apparently $r = 0$ and both the maximal upper prefix and the maximal upper suffix of X are equal to X .

Conversely, if both the maximal upper prefix and the maximal upper suffix of X are equal to X , then the only partitioning of X that starts with the maximal upper prefix (and ends with the maximal upper suffix) is $\mathcal{M} = X$. This is indeed a maximal upper partitioning and there does not exist any other maximal upper partitioning of X .

Now, the claim follows from Lemma 4.44. □

Note that if X does not contain any upper component and contains at least one lower component, then by Lemma 4.38(1) the only separating lower sequence of X is $X_1 = X$, and by Lemma 4.45 both the maximal upper prefix Y_0 and the maximal upper suffix Y_1 of X are empty. Hence, the only maximal upper partitioning of X is $\mathcal{M} = Y_0X_1Y_1 = Y_0XY_1$. Even though $Y_0 = Y_1 = \lambda$, we cannot write $\mathcal{M} = X$ in this case, because formally \mathcal{M} is the sequence Y_0, X_1, Y_1 (with two commas).

It is important to realize what exactly the substrings X_j and Y_j occurring in the definition of a maximal upper sequence may be.



Lemma 4.51 *Let X be a nick free formal DNA molecule and let $\mathcal{M} = Y_0X_1Y_1X_2Y_2 \dots X_rY_r$ for some $r \geq 0$ be a maximal upper partitioning of X .*

1. For $j = 0, 1, \dots, r$, Y_j is either a maximal upper sequence of X , or the empty string λ . The latter case may occur only for $j = 0$ and $j = r$.
2. If \mathcal{M} is the complete maximal upper partitioning of X , then

- (a) each maximal upper sequence of X occurs in \mathcal{M} as Y_j for some (exactly one) j with $0 \leq j \leq r$;
- (b) the substrings X_1, X_2, \dots, X_r are exactly all separating lower sequences of X .

3. For $j = 1, \dots, r$,

- (a) X_j is an alternating sequence of separating lower sequences of X and maximal upper sequences of X , which starts and ends with a separating lower sequence;
- (b) X_j contains at least one single-stranded component, and both the first single-stranded component and the last single-stranded component of X_j are lower components.

Claims 1 and 3 provide us with the following intuitive understanding of a maximal upper partitioning. A maximal upper partitioning of a nick free formal DNA molecule X consists of subsequences $Y_0, X_1, Y_1, X_2, Y_2, \dots, X_r, Y_r$ of components of X , where each subsequence Y_j contains at least one upper component, but does not contain lower components, and each X_j starts with a ‘block’  and ends with a ‘block’ . Exceptions to this (may) occur if X is double-complete, or if the maximal upper prefix Y_0 or the maximal upper suffix Y_r of X is empty.

Proof:

1. This claim is immediate from the definitions of a maximal upper partitioning, a maximal upper sequence and the maximal upper prefix and suffix of a nick free formal DNA molecule.
2. (a) A maximal upper sequence is called internal, if it is not equal to the maximal upper prefix, nor to the maximal upper suffix.
By definition, in the complete maximal upper partitioning of X , the substrings Y_j occurring are the maximal upper prefix Y_0 , all internal maximal upper sequences and the maximal upper suffix Y_r of X , respectively. If the maximal upper prefix and the maximal upper suffix happen to coincide, then by Lemma 4.44, they are equal to X and thus r must be 0.
- (b) If we disregard empty substrings Y_j , which may occur only for $j = 0$ and $j = r$, then by Claims 1 and 2a, the complete maximal upper partitioning is an alternating sequence of all maximal upper sequences Y_j of X and other substrings X_j . By Theorem 4.41, the X_j ’s are exactly the separating lower sequences of X .
3. Consider X_j for an arbitrary j with $1 \leq j \leq r$.
 - (a) X_j is preceded in X by Y_{j-1} and succeeded in X by Y_j . Now, let $\mathcal{M}' = Y'_0 X'_1 Y'_1 X'_2 Y'_2 \dots X'_{r_0} Y'_{r_0}$ for some $r_0 \geq r$ be the complete maximal upper partitioning of X .

Because by definition,

$$\begin{aligned} Y_0 &= Y'_0 \text{ (the maximal upper prefix of } X\text{),} \\ \{Y_1, \dots, Y_{r-1}\} &\subseteq \{Y'_1, \dots, Y'_{r_0-1}\} \\ &\text{(internal maximal upper sequences of } X\text{), and} \\ Y_r &= Y'_{r_0} \text{ (the maximal upper suffix of } X\text{),} \end{aligned}$$

there exist j_0 and j_1 with $0 \leq j_0 < j_1 \leq r_0$ such that $Y_{j-1} = Y'_{j_0}$ and $Y_j = Y'_{j_1}$. Consequently, $X_j = X'_{j_0+1}Y'_{j_0+1} \dots Y'_{j_1-1}X'_{j_1}$, which is, by Claims 1 and 2b, an alternating sequence of separating lower sequences and maximal upper sequences of X , starting and ending with a separating lower sequence.

- (b) By definition, each separating lower sequence of X contains at least one lower component and does not contain any upper component of X . Hence, the first single-stranded component and the last single-stranded component of a separating lower sequence are well defined, and both of them are lower components. Now, the claim follows immediately from Claim 3a. □

By Lemma 4.51(3), we can easily determine the values of $T_{\downarrow}(X_j)$ and $T_{\uparrow}(X_j)$ for the substrings X_j occurring in a maximal upper partitioning.

Corollary 4.52 *Let X be a nick free formal DNA molecule and let $Y_0X_1Y_1X_2Y_2 \dots X_rY_r$ for some $r \geq 0$ be a maximal upper partitioning of X .*

1. For $j = 1, \dots, r$, consider X_j as an alternating sequence of separating lower sequences and maximal upper sequences of X (see Lemma 4.51(3a)). Then $T_{\downarrow}(X_j)$ is equal to the number of separating lower sequences of X occurring in X_j .
2. For $j = 1, \dots, r$, $T_{\uparrow}(X_j) = T_{\downarrow}(X_j) - 1$.

Proof:

1. Let $\mathcal{M}' = Y'_0X'_1Y'_1X'_2Y'_2 \dots X'_{r_0}Y'_{r_0}$ for some $r_0 \geq r$ be the complete maximal upper partitioning of X , and consider X_j with $1 \leq j \leq r$. Then, as in the proof of Lemma 4.51(3a), there exist j_0 and j_1 with $0 \leq j_0 < j_1 \leq r_0$ such that $X_j = X'_{j_0+1}Y'_{j_0+1} \dots Y'_{j_1-1}X'_{j_1}$. By Lemma 4.42, the separating lower sequences of X_j are (precisely) $X'_{j_0+1}, X'_{j_0+2}, \dots, X'_{j_1}$. Now the claim follows from Lemma 4.39, applied to X_j .
2. This claim is immediate from Lemma 4.51(3b) and Lemma 4.12(4). □

Recall that Corollary 4.19 gives lower bounds on the length $|E|$ of a DNA expression E denoting a certain formal DNA molecule X , in terms of $T_{\uparrow}(X)$, $T_{\downarrow}(X)$, $n_{\uparrow}(X)$ and $|\nu(X)|$. We are now ready to describe how to achieve these lower bounds for nick free formal DNA molecules with at least one single-stranded component.

Theorem 4.53 *Let X be a nick free formal DNA molecule which contains at least one single-stranded component and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .*

1. If $T_{\uparrow}(X) \geq T_{\downarrow}(X)$, then let $Y_0X_1Y_1X_2Y_2 \dots X_rY_r$ for some $r \geq 0$ be an arbitrary maximal upper partitioning of X , where for $j = 0, \dots, r$, $Y_j = x'_{a_j} \dots x'_{b_j}$ for some a_j and b_j ; further, let

$$E = \langle \uparrow \varepsilon_{a_0} \dots \varepsilon_{b_0} E_1 \varepsilon_{a_1} \dots \varepsilon_{b_1} E_2 \varepsilon_{a_2} \dots \varepsilon_{b_2} \dots E_r \varepsilon_{a_r} \dots \varepsilon_{b_r} \rangle,$$

where for all applicable i ,

$$\varepsilon_i = \begin{cases} \alpha_i & \text{if } x'_i = \binom{\alpha_i}{-} \text{ for an } \mathcal{N}\text{-word } \alpha_i \\ \langle \downarrow \alpha_i \rangle & \text{if } x'_i = \binom{\alpha_i}{c(\alpha_i)} \text{ for an } \mathcal{N}\text{-word } \alpha_i \end{cases} \quad (4.18)$$

and for $j = 1, \dots, r$, E_j is a minimal DNA expression denoting X_j .

Then E is a minimal DNA expression denoting X and

$$|E| = 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\downarrow}(X) + |\nu(X)|. \quad (4.19)$$

2. If $T_{\downarrow}(X) \geq T_{\uparrow}(X)$, then let $Y_0X_1Y_1X_2Y_2 \dots X_rY_r$ for some $r \geq 0$ be an arbitrary maximal lower partitioning of X , where for $j = 0, \dots, r$, $Y_j = x'_{a_j} \dots x'_{b_j}$ for some a_j and b_j ; further, let

$$E = \langle \downarrow \varepsilon_{a_0} \dots \varepsilon_{b_0} E_1 \varepsilon_{a_1} \dots \varepsilon_{b_1} E_2 \varepsilon_{a_2} \dots \varepsilon_{b_2} \dots E_r \varepsilon_{a_r} \dots \varepsilon_{b_r} \rangle,$$

where for all applicable i ,

$$\varepsilon_i = \begin{cases} \alpha_i & \text{if } x'_i = \binom{-}{\alpha_i} \text{ for an } \mathcal{N}\text{-word } \alpha_i \\ \langle \downarrow \alpha_i \rangle & \text{if } x'_i = \binom{\alpha_i}{c(\alpha_i)} \text{ for an } \mathcal{N}\text{-word } \alpha_i \end{cases}$$

and for $j = 1, \dots, r$, E_j is a minimal DNA expression denoting X_j .

Then E is a minimal DNA expression denoting X and

$$|E| = 3 + 3 \cdot T_{\uparrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|.$$

Note that, because X is nick free, $\nu(X) = X$, so that the term $|\nu(X)|$ in the formulae for $|E|$ may be replaced by $|X|$ – we leave it as it is, as $|\nu(X)|$ has become our standard way of denoting the number of \mathcal{N} -letters occurring in X .

Note further that the construction for minimal DNA expressions described in this result is recursive. The minimal DNA expression specified, which denotes the entire formal DNA molecule X , may have arguments E_j that are themselves minimal DNA expressions denoting formal DNA submolecules X_j of X .

This recursion is well defined. Without loss of generality, assume that $T_{\uparrow}(X) \geq T_{\downarrow}(X)$. As we will see in the proof below, if $T_{\downarrow}(X) = 0$, then there will be no arguments E_j at all. If, on the other hand, $T_{\uparrow}(X) \geq T_{\downarrow}(X) \geq 1$, then we will see that for each formal DNA submolecule X_j occurring in the maximal upper partitioning in Claim 1, $T_{\downarrow}(X) \geq T_{\downarrow}(X_j) > T_{\uparrow}(X_j)$. Hence, the minimum of $T_{\uparrow}(X_j)$ and $T_{\downarrow}(X_j)$ is smaller than the minimum of $T_{\uparrow}(X)$ and $T_{\downarrow}(X)$.

Note finally that if $T_{\uparrow}(X) = T_{\downarrow}(X)$ then both Claim 1 and Claim 2 are applicable, and we have both a minimal \uparrow -expression and a minimal \downarrow -expression denoting X .

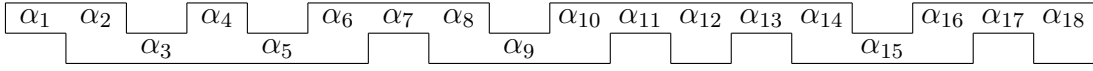


Figure 4.8: The formal DNA molecule from Figure 4.4 with occurring \mathcal{N} -words indicated.

In Figure 4.8, we have specified names for the components of the formal DNA molecule from (a.o.) Figure 4.4 and Figure 4.7. For this formal DNA molecule X , we have $T_{\uparrow}(X) = 4$ and $T_{\downarrow}(X) = 3$. Hence, by the above result, we can construct a minimal DNA expression denoting X from a maximal upper partitioning of X . Because X has two internal maximal upper sequences $(\binom{\alpha_7}{-})$ and $(\binom{\alpha_{11}}{-})\binom{\alpha_{12}}{c(\alpha_{12})}\binom{\alpha_{13}}{-}$, there are, by Lemma 4.48, four different maximal upper partitionings of X . We will consider two of them, the one depicted in Figure 4.7(a2) and the one depicted in Figure 4.7(a3).

For the former maximal upper partitioning, $r = 2$ and

$$\begin{aligned} Y_0 &= \binom{\alpha_1}{-}, \\ X_1 &= \binom{\alpha_2}{c(\alpha_2)}\binom{-}{\alpha_3}\binom{\alpha_4}{c(\alpha_4)}\binom{-}{\alpha_5}\binom{\alpha_6}{c(\alpha_6)}\binom{\alpha_7}{-}\binom{\alpha_8}{c(\alpha_8)}\binom{-}{\alpha_9}\binom{\alpha_{10}}{c(\alpha_{10})}, \\ Y_1 &= \binom{\alpha_{11}}{-}\binom{\alpha_{12}}{c(\alpha_{12})}\binom{\alpha_{13}}{-}, \\ X_2 &= \binom{\alpha_{14}}{c(\alpha_{14})}\binom{-}{\alpha_{15}}\binom{\alpha_{16}}{c(\alpha_{16})}, \\ Y_2 &= \binom{\alpha_{17}}{-}\binom{\alpha_{18}}{c(\alpha_{18})}. \end{aligned}$$

We have $T_{\downarrow}(X_1) = 2 > T_{\uparrow}(X_1) = 1$. When we apply Theorem 4.53(2) to X_1 and Theorem 4.53(1) to the separating upper sequence $(\binom{\alpha_6}{c(\alpha_6)})\binom{\alpha_7}{-}\binom{\alpha_8}{c(\alpha_8)}$ of X_1 , we find that a minimal DNA expression denoting X_1 is

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

Further, $T_{\downarrow}(X_2) = 1 > T_{\uparrow}(X_2) = 0$, and again by Theorem 4.53(2), a minimal DNA expression denoting X_2 is

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

Now, by Theorem 4.53(1), a minimal DNA expression denoting X is

$$E = \langle \uparrow \alpha_1 \langle \downarrow \langle \uparrow \alpha_2 \rangle \alpha_3 \langle \uparrow \alpha_4 \rangle \alpha_5 \langle \uparrow \langle \uparrow \alpha_6 \rangle \alpha_7 \langle \uparrow \alpha_8 \rangle \rangle \alpha_9 \langle \uparrow \alpha_{10} \rangle \rangle \alpha_{11} \langle \uparrow \alpha_{12} \rangle \alpha_{13} \langle \downarrow \langle \uparrow \alpha_{14} \rangle \alpha_{15} \langle \uparrow \alpha_{16} \rangle \rangle \alpha_{17} \langle \uparrow \alpha_{18} \rangle \rangle \quad (4.20)$$

According to the maximal upper partitioning depicted in Figure 4.7(a3), $r = 1$ and

$$\begin{aligned} Y_0 &= \binom{\alpha_1}{-}, \\ X_1 &= \binom{\alpha_2}{c(\alpha_2)}\binom{-}{\alpha_3}\binom{\alpha_4}{c(\alpha_4)}\binom{-}{\alpha_5}\binom{\alpha_6}{c(\alpha_6)}\binom{\alpha_7}{-}\binom{\alpha_8}{c(\alpha_8)}\binom{-}{\alpha_9} \\ &\quad \cdot \binom{\alpha_{10}}{c(\alpha_{10})}\binom{\alpha_{11}}{-}\binom{\alpha_{12}}{c(\alpha_{12})}\binom{\alpha_{13}}{-}\binom{\alpha_{14}}{c(\alpha_{14})}\binom{-}{\alpha_{15}}\binom{\alpha_{16}}{c(\alpha_{16})}, \\ Y_1 &= \binom{\alpha_{17}}{-}\binom{\alpha_{18}}{c(\alpha_{18})}. \end{aligned}$$

We now have $T_{\downarrow}(X_1) = 3$ and $T_{\uparrow}(X_1) = 2$. By Theorem 4.53(2), a minimal DNA expression denoting X_1 can be constructed from a maximal lower partitioning of X_1 .

Contrary to the previous case, X_1 contains an internal maximal lower sequence, $\binom{-}{\alpha_9}$, and thus there exist two different maximal lower partitionings of X_1 . We arbitrarily choose the maximal lower partitioning based on all maximal lower sequences of X_1 , hence including $\binom{-}{\alpha_9}$. For the separating upper sequences $\binom{\alpha_6}{c(\alpha_6)} \binom{\alpha_7}{-} \binom{\alpha_8}{c(\alpha_8)}$ and $\binom{\alpha_{10}}{c(\alpha_{10})} \binom{\alpha_{11}}{-} \binom{\alpha_{12}}{c(\alpha_{12})} \binom{\alpha_{13}}{-} \binom{\alpha_{14}}{c(\alpha_{14})}$ of X_1 , we find a minimal DNA-expression with Theorem 4.53(1). The resulting minimal DNA-expression for X_1 is

$$E_1 = \langle \downarrow \langle \downarrow \alpha_2 \rangle \alpha_3 \langle \downarrow \alpha_4 \rangle \alpha_5 \langle \uparrow \langle \downarrow \alpha_6 \rangle \alpha_7 \langle \downarrow \alpha_8 \rangle \rangle \alpha_9 \\ \langle \uparrow \langle \downarrow \alpha_{10} \rangle \alpha_{11} \langle \downarrow \alpha_{12} \rangle \alpha_{13} \langle \downarrow \alpha_{14} \rangle \rangle \alpha_{15} \langle \downarrow \alpha_{16} \rangle \rangle$$

and the corresponding minimal DNA-expression denoting X is

$$E = \langle \uparrow \alpha_1 \langle \downarrow \langle \downarrow \alpha_2 \rangle \alpha_3 \langle \downarrow \alpha_4 \rangle \alpha_5 \langle \uparrow \langle \downarrow \alpha_6 \rangle \alpha_7 \langle \downarrow \alpha_8 \rangle \rangle \alpha_9 \\ \langle \uparrow \langle \downarrow \alpha_{10} \rangle \alpha_{11} \langle \downarrow \alpha_{12} \rangle \alpha_{13} \langle \downarrow \alpha_{14} \rangle \rangle \alpha_{15} \langle \downarrow \alpha_{16} \rangle \rangle \alpha_{17} \langle \downarrow \alpha_{18} \rangle \rangle.$$

Indeed, both minimal DNA expressions for X have length

$$|E| = 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)| = 3 + 3 \cdot 3 + 3 \cdot 9 + |\nu(X)| = 39 + |\nu(X)|.$$

Proof of Theorem 4.53: First we prove for Claim 1, that the *string* E is well defined and that it is indeed a DNA expression denoting X . The proof for the other claim is entirely analogous.

- Because X is nick free, by Theorem 3.5 there exist DNA expressions denoting X , and thus it makes sense to talk about a minimal DNA expression denoting X . For the same reason, there exist maximal upper partitionings of X (and Lemma 4.48 specifies how many different partitionings there are).

By definition, the maximal upper prefix $Y_0 = x'_{a_0} \dots x'_{b_0}$, the internal maximal upper sequences $Y_j = x'_{a_j} \dots x'_{b_j}$ for $j = 1, \dots, r-1$, and the maximal upper suffix $Y_r = x'_{a_r} \dots x'_{b_r}$ consist of only double components $\binom{\alpha_i}{c(\alpha_i)}$ and upper components $\binom{\alpha_i}{-}$ for \mathcal{N} -words α_i . Hence, the arguments ε_i are well defined.

Obviously, for $j = 1, \dots, r$, the substring X_j of X is nick free. Hence, X_j is expressible and there exists a (at least one) minimal DNA expression E_j denoting X_j .

- Clearly, for each applicable i , $\mathcal{S}^+(\varepsilon_i) = x'_i$. Because x'_i is either a double component or an upper component of X , consecutive arguments ε_i and ε_{i+1} fit together by upper strands.

For $j = 1, \dots, r$, if $Y_{j-1} \neq \lambda$ (which is certainly the case if $j \geq 2$), then by Lemma 4.51(1), it is a maximal upper sequence. By Lemma 4.27(2), $R(\mathcal{S}^+(\varepsilon_{b_{j-1}})) = R(x'_{b_{j-1}}) = R(Y_{j-1}) \in \mathcal{A}_+$ and $L(\mathcal{S}^+(E_j)) = L(X_j) = L(x'_{b_{j-1}+1}) \in \mathcal{A}_{\pm}$. Hence, the argument $\varepsilon_{b_{j-1}}$ prefits E_j by upper strands. Analogously, for $j = 1, \dots, r$, if $Y_j \neq \lambda$ (which is certainly the case if $j \leq r-1$), then the argument E_j prefits ε_{a_j} by upper strands.

Consequently, E is indeed a DNA expression.

- For notational convenience, we assume that both Y_0 and Y_r are non-empty. Then by Lemma 3.6,

$$E \equiv E' = \langle \uparrow \langle \uparrow \varepsilon_{a_0} \dots \varepsilon_{b_0} \rangle E_1 \langle \uparrow \varepsilon_{a_1} \dots \varepsilon_{b_1} \rangle E_2 \langle \uparrow \varepsilon_{a_2} \dots \varepsilon_{b_2} \rangle \dots E_r \langle \uparrow \varepsilon_{a_r} \dots \varepsilon_{b_r} \rangle \rangle.$$

For an arbitrary j with $0 \leq j \leq r$, we consider the argument $\langle \uparrow \varepsilon_{a_j} \dots \varepsilon_{b_j} \rangle$ of E' . We established before that for $i = a_j, \dots, b_j$, $\mathcal{S}^+(\varepsilon_i) = x'_i$ is a double component or an upper component of X . By definition, such components are nick free. By Corollary 2.5, the double components and upper components occur in $x'_{a_j} \dots x'_{b_j}$, alternately. In particular, we do not have two consecutive double components. Hence, the operator \uparrow does not introduce (lower) nick letters between its arguments, and

$$\mathcal{S}(\langle \uparrow \varepsilon_{a_j} \dots \varepsilon_{b_j} \rangle) = \nu^+(x'_{a_j}) \dots \nu^+(x'_{b_j}) = x'_{a_j} \dots x'_{b_j} = Y_j.$$

We use this to determine $\mathcal{S}(E')$:

$$\begin{aligned} \mathcal{S}(E') &= \nu^+(Y_0)y_1\nu^+(X_1)y_2\nu^+(Y_1)y_3\nu^+(X_2)y_4\nu^+(Y_2) \dots \nu^+(X_r)y_{2r}\nu^+(Y_r) \quad (4.21) \\ &= Y_0y_1X_1y_2Y_1y_3X_2y_4Y_2 \dots X_ry_{2r}Y_r, \end{aligned}$$

where the y_i 's are defined as in (2.15). The second equality in (4.21) holds because each maximal upper sequence Y_j and each substring X_j of X is nick free.

By Lemma 4.27, for $j = 1, \dots, r$, $R(Y_{j-1}) \in \mathcal{A}_+$, $L(X_j), R(X_j) \in \mathcal{A}_\pm$ and $L(Y_j) \in \mathcal{A}_+$. Consequently, all y_i 's are empty, and thus

$$\mathcal{S}(E) = \mathcal{S}(E') = Y_0X_1Y_1X_2Y_2 \dots X_rY_r = X.$$

We subsequently prove that E has the specified length, and (thus) is minimal. We do this simultaneously for Claim 1 and Claim 2, by induction on the lower of $T_\uparrow(X)$ and $T_\downarrow(X)$.

- If $T_\uparrow(X) \geq T_\downarrow(X) = 0$, then by Lemma 4.10(2), X does not contain any lower component, and by Lemma 4.50 the only maximal upper partitioning of X is $\mathcal{M} = Y_0 = X$. Hence,

$$E = \langle \uparrow \varepsilon_1 \dots \varepsilon_k \rangle,$$

where for $i = 1, \dots, k$, ε_i is defined by (4.18). Then obviously,

$$|E| = 3 + 3 \cdot n_\uparrow(X) + |X| = 3 + 3 \cdot T_\downarrow(X) + 3 \cdot n_\uparrow(X) + |\nu(X)|.$$

By Corollary 4.19(1), this is the minimal length possible for a \uparrow -expression denoting X . In order to conclude that E is a minimal DNA expression for X , we have to verify that there does not exist a shorter \downarrow -expression or \updownarrow -expression for X .

By assumption, X contains at least one single-stranded component, which must be an upper component. By definition, the semantics of a \downarrow -expression does not contain any single-stranded component. Consequently, there does not exist any \downarrow -expression denoting X , let alone a \updownarrow -expression shorter than E .

By Lemma 4.10(1), $T_\uparrow(X)$ must be positive, and hence larger than $T_\downarrow(X)$. By Lemma 4.11(1), $T_\uparrow(X) = 1$. Now, let E' be an arbitrary \downarrow -expression denoting X . Then by Corollary 4.19(2), E' is longer than E :

$$|E'| \geq 3 + 3 \cdot T_\uparrow(X) + 3 \cdot n_\uparrow(X) + |\nu(X)| = 6 + 3 \cdot n_\uparrow(X) + |X| > |E|. \quad (4.22)$$

Hence, only \uparrow -expressions denoting X may be minimal, and in particular, E is minimal.

- The proof for the case that $T_{\downarrow}(X) \geq T_{\uparrow}(X) = 0$ is, of course, analogous.
- Let $p \geq 0$, and suppose that for each nick free formal DNA molecule X with either $T_{\downarrow}(X) \leq p$, or $T_{\uparrow}(X) \leq p$, a DNA expression E as specified in Claim 1 or Claim 2 (depending on which one is applicable) is a minimal DNA expression denoting X , with the prescribed length (induction hypothesis).

Now consider a nick free formal DNA molecule X for which $T_{\uparrow}(X) \geq T_{\downarrow}(X) = p+1$. Let $\mathcal{M} = Y_0 X_1 Y_1 X_2 Y_2 \dots X_r Y_r$ be an arbitrary maximal upper partitioning of X , where for $j = 0, \dots, r$, $Y_j = x'_{a_j} \dots x'_{b_j}$ for some a_j and b_j . Let

$$E = \langle \uparrow \varepsilon_{a_0} \dots \varepsilon_{b_0} E_1 \varepsilon_{a_1} \dots \varepsilon_{b_1} E_2 \varepsilon_{a_2} \dots \varepsilon_{b_2} \dots E_r \varepsilon_{a_r} \dots \varepsilon_{b_r} \rangle,$$

be the DNA expression denoting X which corresponds to \mathcal{M} , as described in Claim 1.

By definition, for each applicable i , $\mathcal{S}^+(\varepsilon_i) = x'_i$ is either an upper component or a double component of X , and for $j = 1, \dots, r$, $\mathcal{S}^+(E_j) = \mathcal{S}(E_j) = X_j$. For each argument ε of E , $\mathcal{S}^+(\varepsilon)$ is nick free, and in particular $\#\nabla(\mathcal{S}^+(\varepsilon)) = 0$. Now, by equations (4.2), (4.3) and (4.4) from Lemma 4.16(1),

$$T_{\downarrow}(X) = \sum_{\substack{\text{arguments} \\ \varepsilon \text{ of } E}} T_{\downarrow}(\mathcal{S}^+(\varepsilon)) = \sum_{j=0}^r \sum_{i=a_j}^{b_j} T_{\downarrow}(x'_i) + \sum_{j=1}^r T_{\downarrow}(X_j) \quad \text{and (4.23)}$$

$$n_{\downarrow}(X) = \sum_{\substack{\text{arguments} \\ \varepsilon \text{ of } E}} n_{\downarrow}(\mathcal{S}^+(\varepsilon)) = \sum_{j=0}^r \sum_{i=a_j}^{b_j} n_{\downarrow}(x'_i) + \sum_{j=1}^r n_{\downarrow}(X_j). \quad (4.24)$$

We consider Y_j with $0 \leq j \leq r$. $Y_j = x'_{a_j} \dots x'_{b_j}$ is an alternating sequence of upper components and double components x'_i of X . Obviously, $T_{\downarrow}(x'_i) = 0$ for each of these components x'_i , and hence, $\sum_{i=a_j}^{b_j} T_{\downarrow}(x'_i) = 0$. Because of the alternation of the upper components and the double components x'_i in Y_j , we also have $\sum_{i=a_j}^{b_j} n_{\downarrow}(x'_i) = n_{\downarrow}(Y_j)$. We can therefore rewrite (4.23) and (4.24) into

$$T_{\downarrow}(X) = \sum_{j=1}^r T_{\downarrow}(X_j) \quad \text{and} \quad (4.25)$$

$$n_{\downarrow}(X) = \sum_{j=0}^r n_{\downarrow}(Y_j) + \sum_{j=1}^r n_{\downarrow}(X_j). \quad (4.26)$$

As $T_{\downarrow}(X) = p+1 \geq 1$, (4.25) implies in particular that $r \geq 1$.

For an arbitrary j with $1 \leq j \leq r$, we now concentrate on the substring X_j . Obviously $T_{\downarrow}(X_j) \leq p+1$. Further, by Corollary 4.52(2), $T_{\uparrow}(X_j) = T_{\downarrow}(X_j) - 1 \leq p$. Now, by the induction hypothesis, we can construct a minimal \downarrow -expression E'_j denoting X_j for which

$$|E'_j| = 3+3 \cdot T_{\uparrow}(X_j) + 3 \cdot n_{\downarrow}(X_j) + |\nu(X_j)| = 3 \cdot T_{\downarrow}(X_j) + 3 \cdot n_{\downarrow}(X_j) + |\nu(X_j)|. \quad (4.27)$$

Because all minimal DNA expressions denoting X_j have the same length, this is also the length of the DNA expression E_j occurring in E .

When we combine (4.27) with (4.25) and (4.26), we can establish the length of E :

$$\begin{aligned}
|E| &= 3 + \sum_{j=0}^r \sum_{i=a_j}^{b_j} |\varepsilon_i| + \sum_{j=1}^r |E_j| \\
&= 3 + \sum_{j=0}^r (3 \cdot n_{\uparrow}(Y_j) + |Y_j|) + \sum_{j=1}^r (3 \cdot T_{\downarrow}(X_j) + 3 \cdot n_{\uparrow}(X_j) + |\nu(X_j)|) \\
&= 3 + 3 \cdot \sum_{j=1}^r T_{\downarrow}(X_j) + 3 \cdot \left(\sum_{j=0}^r n_{\uparrow}(Y_j) + \sum_{j=1}^r n_{\uparrow}(X_j) \right) \\
&\quad + \sum_{j=0}^r |\nu(Y_j)| + \sum_{j=1}^r |\nu(X_j)| \\
&= 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|.
\end{aligned}$$

By Corollary 4.19(1), this is the minimal length for a \uparrow -expression denoting X .

Because X contains at least one single-stranded component (in fact, at least $p + 1$ upper components and $p + 1$ lower components), there does not exist any \uparrow -expression denoting X . If E' is a \downarrow -expression denoting X , then by Corollary 4.19(2), E' cannot be shorter than E :

$$\begin{aligned}
|E'| &\geq 3 + 3 \cdot T_{\uparrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)| \\
&\geq 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)| = |E|.
\end{aligned}$$

We conclude that E is a minimal DNA expression for X with the prescribed length.

- The proof for the case that $T_{\downarrow}(X) \geq T_{\uparrow}(X) = p + 1$ is, of course, analogous.

□

Two intermediate results in the above proof are worth to be generalized.

Corollary 4.54 *Let X be a nick free formal DNA molecule and let $Y_0 X_1 Y_1 X_2 Y_2 \dots X_r Y_r$ for some $r \geq 0$ be a maximal upper partitioning of X .*

1. $T_{\downarrow}(X_1) + \dots + T_{\downarrow}(X_r) = T_{\downarrow}(X)$.
2. $n_{\uparrow}(Y_0) + n_{\uparrow}(X_1) + n_{\uparrow}(Y_1) + n_{\uparrow}(X_2) + n_{\uparrow}(Y_2) + \dots + n_{\uparrow}(X_r) + n_{\uparrow}(Y_r) = n_{\uparrow}(X)$.

Note that if $T_{\downarrow}(X) = 0$, then by Lemma 4.10(2), X does not contain any lower component, and by Lemma 4.50, the only maximal upper partitioning of X is $\mathcal{M} = X$. In that case, $r = 0$ and the claims are immediate.

Proof: The claims appeared as equalities (4.25) and (4.26) in the proof of Theorem 4.53 for the case that $T_{\uparrow}(X) \geq T_{\downarrow}(X) \geq 1$. Their validity followed after the construction

of a (minimal) \uparrow -expression E denoting X which was based on the maximal upper partitioning.

If $T_{\downarrow}(X) = 0$ or $T_{\downarrow}(X) > T_{\uparrow}(X)$, then we can also construct a \uparrow -expression E denoting X according to the description in Theorem 4.53(1). Although this \uparrow -expression is not necessarily minimal, the claims can be verified in the same way as for the case that $T_{\uparrow}(X) \geq T_{\downarrow}(X) \geq 1$. \square

By Theorem 4.23 and Theorem 4.53, we do not only know how to construct a minimal DNA expression E for a given nick free formal DNA molecule. We also know the length $|E|$ of this minimal DNA expression without having to explicitly construct the DNA expression itself. The length is simply a function of some elementary structural properties of the formal DNA molecule.

We can combine the two values for $|E|$ from Theorem 4.53(1) and (2).

Corollary 4.55 *Let X be a nick free formal DNA molecule which contains at least one single-stranded component, and let E be a minimal DNA expression denoting X . Then*

$$|E| = 3 + 3 \cdot p + 3 \cdot n_{\uparrow}(X) + |\nu(X)|,$$

where p is the minimum of $T_{\downarrow}(X)$ and $T_{\uparrow}(X)$.

Proof: Let p be the minimum of $T_{\downarrow}(X)$ and $T_{\uparrow}(X)$.

If $p = T_{\downarrow}(X)$ (hence $T_{\uparrow}(X) \geq T_{\downarrow}(X)$), then Theorem 4.53(1) specifies the length of a minimal DNA expression denoting X . In particular, it specifies the length of E :

$$|E| = 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)| = 3 + 3 \cdot p + 3 \cdot n_{\uparrow}(X) + |\nu(X)|.$$

If, on the other hand, $p = T_{\uparrow}(X)$ (hence $T_{\downarrow}(X) \geq T_{\uparrow}(X)$), then by Theorem 4.53(2),

$$|E| = 3 + 3 \cdot T_{\uparrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)| = 3 + 3 \cdot p + 3 \cdot n_{\uparrow}(X) + |\nu(X)|.$$

\square

We will see later that sometimes it is sufficient to have an upper bound on the length of a minimal DNA expression for a nick free formal DNA molecule. We give two such upper bounds here.

Corollary 4.56 *Let X be a nick free formal DNA molecule and let E be a minimal DNA expression denoting X .*

1. $|E| \leq 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|.$ (4.28)
2. $|E| \leq 3 + 3 \cdot T_{\uparrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|.$

Proof:

1. If X does not contain any single-stranded component, hence $X = \binom{\alpha_1}{c(\alpha_1)}$ for an \mathcal{N} -word α_1 , then by definition $T_{\downarrow}(X) = 0$, $n_{\uparrow}(X) = 1$ and $|\nu(X)| = |X| = |\alpha_1|$. By Theorem 4.23, $E = \langle \uparrow \alpha_1 \rangle$ is the unique minimal DNA expression denoting X . Then the left hand side of (4.28) evaluates to $3 + |\alpha_1|$ and the right hand side evaluates to $6 + |\alpha_1|$. Indeed, the inequality holds.

If X contains at least one single-stranded component, then the claim follows from Corollary 4.55, because obviously the minimum p of $T_{\downarrow}(X)$ and $T_{\uparrow}(X)$ satisfies $p \leq T_{\downarrow}(X)$.

2. The proof of this claim is analogous to that of the previous claim.

□

For future reference, we prove two properties of the arguments of the minimal DNA expressions we construct by Theorem 4.53.

Lemma 4.57 *Let X be a nick free formal DNA molecule which contains at least one single-stranded component, and let E be a minimal DNA expression denoting X as described in Theorem 4.53.*

1. *Each \mathcal{N} -word-argument of E is a maximal \mathcal{N} -word occurrence in E .*
2. *The arguments of E are \mathcal{N} -words and DNA expressions, alternately.*

Proof: Without loss of generality, assume that E has been constructed according to Theorem 4.53(1). Hence, $T_{\uparrow}(X) \geq T_{\downarrow}(X)$ and E is a \uparrow -expression.

Let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X . Further, let $Y_0 X_1 Y_1 X_2 Y_2 \dots X_r Y_r$ for some $r \geq 0$ be the maximal upper partitioning of X that E is based on, where for $j = 0, \dots, r$, $Y_j = x'_{a_j} \dots x'_{b_j}$ for some a_j and b_j . Finally, let

$$E = \langle \uparrow \varepsilon_{a_0} \dots \varepsilon_{b_0} E_1 \varepsilon_{a_1} \dots \varepsilon_{b_1} E_2 \varepsilon_{a_2} \dots \varepsilon_{b_2} \dots E_r \varepsilon_{a_r} \dots \varepsilon_{b_r} \rangle$$

as specified in Theorem 4.53(1).

1. Consider an arbitrary \mathcal{N} -word-argument of E .

For $j = 1, \dots, r$, E_j is a minimal DNA expression denoting X_j , and in particular E_j is not an \mathcal{N} -word. Hence, the \mathcal{N} -word we consider must be equal to $\varepsilon_i = \alpha_i$ with $a_j \leq i \leq b_j$ for some j with $0 \leq j \leq r$. By construction, $x'_i = \mathcal{S}^+(\varepsilon_i) = \binom{\alpha_i}{-}$ is an upper component of X , which is part of the maximal upper sequence $Y_j = x'_{a_j} \dots x'_{b_j}$.

If $a_j \leq i - 1$, then by Corollary 2.5, x'_{i-1} is a double component $\binom{\alpha_{i-1}}{c(\alpha_{i-1})}$ of X for an \mathcal{N} -word α_{i-1} , which is also part of Y_j . Hence, the argument preceding $\varepsilon_i = \alpha_i$ in E is the \downarrow -expression $\varepsilon_{i-1} = \langle \downarrow \alpha_{i-1} \rangle$.

If $a_j = i$ and $j = 0$, then $a_j = i = 1$ and ε_i is the first argument of E . If $a_j = i$ and $j \geq 1$, then ε_i is preceded by the (minimal) DNA expression E_j .

In none of the cases, ε_i is preceded in E by an \mathcal{N} -word-argument. Analogously, we can prove that ε_i is not succeeded in E by an \mathcal{N} -word-argument. Consequently, the \mathcal{N} -word $\varepsilon_i = \alpha_i$ is a maximal \mathcal{N} -word occurrence.

2. Consider an argument ε of E that is not the last argument.

If ε is an \mathcal{N} -word, then by Claim 1, it is a maximal \mathcal{N} -word occurrence and it is succeeded by an argument that is a DNA expression.

If, on the other hand, ε is a DNA expression, then by construction, either it is E_j for some j with $1 \leq j \leq r$, or it is a \downarrow -expression $\varepsilon_i = \langle \downarrow \alpha_i \rangle$ for an \mathcal{N} -word α_i with $a_j \leq i \leq b_j$ for some j with $0 \leq j \leq r$.

In the former case, $\varepsilon = E_j$ denotes the substring X_j of X , which is succeeded in X by the maximal upper sequence $Y_j = x'_{a_j} \dots x'_{b_j}$. By Lemma 4.27(1), the first

component x'_{a_j} of Y_j is an (opening) upper component $\binom{\alpha_{a_j}}{-}$ for an \mathcal{N} -word α_{a_j} . Hence, the argument succeeding $\varepsilon = E_j$ in E is (the \mathcal{N} -word) $\varepsilon_{a_j} = \alpha_{a_j}$.

In the latter case, $\varepsilon = \varepsilon_i = \langle \uparrow \alpha_i \rangle$ denotes the double component $x'_i = \binom{\alpha_i}{c(\alpha_i)}$ of X , which is part of the maximal upper sequence Y_j . Suppose that x'_i is the last component of Y_j , hence $i = b_j$. Then, because ε is not the last argument of E , $j \leq r - 1$ and Y_j is succeeded in X by the substring X_{j+1} . This, however, gives a contradiction, because by Lemma 4.27(2), x'_{b_j} is a (closing) upper component. Hence, $i \leq b_j - 1$ and x'_i is succeeded in Y_j by the upper component $x'_{i+1} = \binom{\alpha_{i+1}}{-}$ for an \mathcal{N} -word α_{i+1} . This upper component corresponds to an \mathcal{N} -word-argument $\varepsilon_{i+1} = \alpha_{i+1}$, which succeeds $\varepsilon_i = \langle \uparrow \alpha_i \rangle$ in E .

In both cases, the DNA expression ε is succeeded in E by an \mathcal{N} -word-argument. □

As we mentioned after the statement of Theorem 4.53, if X is a nick free formal DNA molecule containing at least one single-stranded component and $T_\uparrow(X) = T_\downarrow(X)$, then there both exist a minimal \uparrow -expression and a minimal \downarrow -expression denoting X .

We now show that if X is nick free and $T_\uparrow(X) \neq T_\downarrow(X)$, then all minimal DNA expressions are of the same type: they are either \uparrow -expressions or \downarrow -expressions, depending on which of $T_\uparrow(X)$ and $T_\downarrow(X)$ is higher.

Lemma 4.58 *Let X be a nick free formal DNA molecule and let E be a minimal DNA expression denoting X .*

1. *If $T_\uparrow(X) > T_\downarrow(X)$, then E is a \uparrow -expression.*
2. *If $T_\downarrow(X) > T_\uparrow(X)$, then E is a \downarrow -expression.*

Proof:

1. Assume that $T_\uparrow(X) > T_\downarrow(X)$. Then X contains at least one (opening) upper component. Because the semantics of a \uparrow -expression does not contain single-stranded components, E cannot be a \downarrow -expression.

By Corollary 4.55,

$$|E| = 3 + 3 \cdot T_\downarrow(X) + 3 \cdot n_\uparrow(X) + |\nu(X)|.$$

By Corollary 4.19(2), each \downarrow -expression E' denoting X satisfies

$$|E'| \geq 3 + 3 \cdot T_\uparrow(X) + 3 \cdot n_\uparrow(X) + |\nu(X)| > |E|,$$

because $T_\uparrow(X) > T_\downarrow(X)$. Consequently, a \downarrow -expression E' denoting X cannot be minimal and E has to be a \uparrow -expression.

2. The proof of this claim is analogous to that of the previous claim. □

When we combine this result with Corollary 4.52(2), we immediately obtain

Corollary 4.59 *Let X be a nick free formal DNA molecule.*

1. *Let $Y_0X_1Y_1X_2Y_2 \dots X_rY_r$ for some $r \geq 0$ be an arbitrary maximal upper partitioning of X . Then for $j = 1, \dots, r$, each minimal DNA expression E_j denoting X_j (in particular, the one occurring in Theorem 4.53(1)) is a \downarrow -expression.*
2. *Let $Y_0X_1Y_1X_2Y_2 \dots X_rY_r$ for some $r \geq 0$ be an arbitrary maximal lower partitioning of X . Then for $j = 1, \dots, r$, each minimal DNA expression E_j denoting X_j (in particular, the one occurring in Theorem 4.53(2)) is a \uparrow -expression.*

4.2.2 Minimal DNA expressions for a formal DNA molecule with nick letters

For expressible formal DNA molecules that contain nick letters, it is easy to say what type of DNA expressions (\uparrow -expressions, \downarrow -expressions or \updownarrow -expressions) can be minimal.

Lemma 4.60 *Let X be an expressible formal DNA molecule.*

1. *If X contains at least one lower nick letter \triangle , then each minimal DNA expression denoting X is a \uparrow expression.*
2. *If X contains at least one upper nick letter ∇ , then each minimal DNA expression denoting X is a \downarrow expression.*

Proof:

1. Assume that X contains at least one lower nick letter. By Corollary 4.24, a minimal DNA expression denoting X is either a \uparrow -expression or a \downarrow -expression. However, by Lemma 3.2(1), there does not exist any \downarrow -expression denoting X , let alone a minimal \downarrow -expression denoting X . Consequently, a minimal DNA expression denoting X must be a \uparrow -expression.
2. The proof of this claim is analogous to that of the previous claim.

□

Given an expressible formal DNA molecule with nick letters, it is, of course, not sufficient to know the *type* of the minimal DNA expression(s) denoting it. We want to construct the minimal DNA expression(s) themselves. For this, we decompose a formal DNA molecule into nick free pieces and nick letters, as follows:

Definition 4.61 *Let X be a formal DNA molecule. The nick free decomposition of X is the sequence $Z_1, y_1, Z_2, y_2, \dots, y_{m-1}, Z_m$ for some $m \geq 1$ such that*

- $X = Z_1y_1Z_2y_2 \dots y_{m-1}Z_m$, and
- for $h = 1, \dots, m$, Z_h is nick free, and
- for $h = 1, \dots, m - 1$, $y_h \in \{\nabla, \triangle\}$.

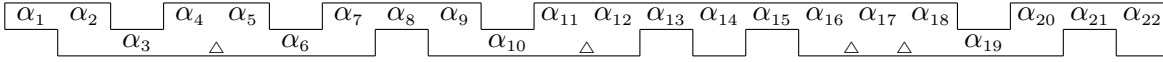


Figure 4.9: A (pictorial representation of a) formal DNA molecule containing lower nick letters.

To simplify the notation, we will in general write $Z_1y_1Z_2y_2\dots y_{m-1}Z_m$ instead of $Z_1, y_1, Z_2, y_2, \dots, y_{m-1}, Z_m$ to denote the nick free decomposition of a formal DNA molecule X .

Obviously, because the substrings Z_h in the definition are nick free and the y_i 's are exactly the nick letters occurring in X , the nick free decomposition of a formal DNA molecule is well defined.

Consider, for example, the formal DNA molecule X depicted in Figure 4.9. This molecule contains four lower nick letters and no upper nick letters. The nick free decomposition of X is $Z_{1\triangle}Z_{2\triangle}Z_{3\triangle}Z_{4\triangle}Z_5$, where

$$\begin{aligned}
 Z_1 &= \begin{pmatrix} \alpha_1 \\ - \end{pmatrix} \begin{pmatrix} \alpha_2 \\ c(\alpha_2) \end{pmatrix} \begin{pmatrix} - \\ \alpha_3 \end{pmatrix} \begin{pmatrix} \alpha_4 \\ c(\alpha_4) \end{pmatrix}, \\
 Z_2 &= \begin{pmatrix} \alpha_5 \\ c(\alpha_5) \end{pmatrix} \begin{pmatrix} - \\ \alpha_6 \end{pmatrix} \begin{pmatrix} \alpha_7 \\ c(\alpha_7) \end{pmatrix} \begin{pmatrix} \alpha_8 \\ - \end{pmatrix} \begin{pmatrix} \alpha_9 \\ c(\alpha_9) \end{pmatrix} \begin{pmatrix} - \\ \alpha_{10} \end{pmatrix} \begin{pmatrix} \alpha_{11} \\ c(\alpha_{11}) \end{pmatrix}, \\
 Z_3 &= \begin{pmatrix} \alpha_{12} \\ c(\alpha_{12}) \end{pmatrix} \begin{pmatrix} \alpha_{13} \\ - \end{pmatrix} \begin{pmatrix} \alpha_{14} \\ c(\alpha_{14}) \end{pmatrix} \begin{pmatrix} \alpha_{15} \\ - \end{pmatrix} \begin{pmatrix} \alpha_{16} \\ c(\alpha_{16}) \end{pmatrix}, \\
 Z_4 &= \begin{pmatrix} \alpha_{17} \\ c(\alpha_{17}) \end{pmatrix}, \\
 Z_5 &= \begin{pmatrix} \alpha_{18} \\ c(\alpha_{18}) \end{pmatrix} \begin{pmatrix} - \\ \alpha_{19} \end{pmatrix} \begin{pmatrix} \alpha_{20} \\ c(\alpha_{20}) \end{pmatrix} \begin{pmatrix} \alpha_{21} \\ - \end{pmatrix} \begin{pmatrix} \alpha_{22} \\ c(\alpha_{22}) \end{pmatrix}.
 \end{aligned} \tag{4.29}$$

We give two properties of the substrings Z_h in a nick free decomposition:

Lemma 4.62 *Let X be a formal DNA molecule, let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X , and let $Z_1y_1Z_2y_2\dots y_{m-1}Z_m$ for some $m \geq 1$ be the nick free decomposition of X .*

Then for $h = 1, \dots, m$,

1. Z_h is a formal DNA submolecule of X and in particular $Z_h \neq \lambda$.
2. there exist a_h and b_h with $1 \leq a_h \leq b_h \leq k$ such that $Z_1y_1\dots Z_{h-1}y_{h-1} = x'_1 \dots x'_{a_h-1}$ and $y_hZ_{h+1}\dots y_{m-1}Z_m = x'_{b_h+1} \dots x'_k$ (hence, $Z_h = x'_{a_h} \dots x'_{b_h}$).

Proof: Consider Z_h with $1 \leq h \leq m$.

1. By the definition of a formal DNA molecule, $X \neq \lambda$, $L(X), R(X) \in \mathcal{A}$ and nick letters do not occur in consecutive positions in X . This implies that $Z_h \neq \lambda$. Because Z_h is nick free, in particular $L(Z_h), R(Z_h) \in \mathcal{A}$.

Now the claim follows from Lemma 2.2.

2. If $h = 1$, then $Z_1y_1\dots Z_{h-1}y_{h-1} = \lambda$ and the value $a_h = 1$ suffices for the first part of the claim.

If $h \geq 2$, then $R(Z_1y_1\dots Z_{h-1}y_{h-1}) = y_{h-1} \in \{\nabla, \triangle\}$. Because each nick letter occurring in X is by definition a component of X , there exists an $a_h \geq 2$ such that $y_{h-1} = x'_{a_h-1}$ and $Z_1y_1\dots Z_{h-1}y_{h-1} = x'_1 \dots x'_{a_h-1}$.

In an analogous way, we find a $b_h \leq k$ such that $y_hZ_{h+1}\dots y_{m-1}Z_m = x'_{b_h+1} \dots x'_k$.

By Claim 1, $Z_h = x'_{a_h} \dots x'_{b_h}$ is non-empty. Hence, $a_h \leq b_h$. \square

In order to describe minimal DNA expressions denoting an expressible formal DNA molecule containing nick letters (e.g., minimal \uparrow -expressions denoting a formal DNA molecule with lower nick letters), we need the concept of an operator-minimal DNA expression.

Definition 4.63 *A DNA expression E is operator-minimal if for every DNA expression E' with the same outermost operator as E and with $\mathcal{S}(E') = \mathcal{S}(E)$, $|E'| \geq |E|$.*

For example, a \uparrow -expression E denoting a formal DNA molecule X is operator-minimal if there does not exist a shorter \uparrow -expression denoting X . Obviously, each minimal DNA expression is also operator-minimal.

To illustrate the notion of operator-minimality, we return to the formal DNA molecule X depicted in Figure 4.9. The second formal DNA submolecule occurring in the nick free decomposition of X is

$$Z_2 = \binom{\alpha_5}{c(\alpha_5)} \binom{-}{\alpha_6} \binom{\alpha_7}{c(\alpha_7)} \binom{\alpha_8}{-} \binom{\alpha_9}{c(\alpha_9)} \binom{-}{\alpha_{10}} \binom{\alpha_{11}}{c(\alpha_{11})} \quad (4.30)$$

(see (4.29)). We have $T_\uparrow(Z_2) = 1$ and $T_\downarrow(Z_2) = 2$.

By Lemma 4.58(2), each minimal DNA expression E_2 denoting Z_2 is a \downarrow -expression. When we apply Theorem 4.53(2) to Z_2 , we obtain

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

for which

$$|E_2| = 3 + 3 \cdot T_\uparrow(Z_2) + 3 \cdot n_\uparrow(Z_2) + |\nu(Z_2)| = 3 + 3 \cdot 1 + 3 \cdot 4 + |\nu(Z_2)| = 18 + |\nu(Z_2)|.$$

Now let E'_2 be a \uparrow -expression denoting Z_2 . Then by Corollary 4.19(1),

$$|E'_2| \geq 3 + 3 \cdot T_\downarrow(Z_2) + 3 \cdot n_\downarrow(Z_2) + |\nu(Z_2)| = 3 + 3 \cdot 2 + 3 \cdot 4 + |\nu(Z_2)| = 21 + |\nu(Z_2)|.$$

Indeed, a \uparrow -expression E'_2 denoting Z_2 will never be minimal. If, however, $E'_2 = 21 + |\nu(Z_2)|$, then E'_2 is operator-minimal. It is not difficult to construct an operator-minimal \uparrow -expression denoting Z_2 . We can simply take

$$E'_2 = \langle \uparrow E_2 \rangle = \langle \uparrow \langle \downarrow \langle \uparrow \alpha_5 \rangle \alpha_6 \langle \uparrow \langle \downarrow \alpha_7 \rangle \alpha_8 \langle \downarrow \alpha_9 \rangle \rangle \alpha_{10} \langle \downarrow \alpha_{11} \rangle \rangle \rangle, \quad (4.31)$$

because $\mathcal{S}(E'_2) = \nu^+(\mathcal{S}(E_2)) = \mathcal{S}(E_2) = Z_2$. Another operator-minimal \uparrow -expression denoting Z_2 , which is less directly related to E_2 , is

$$E''_2 = \langle \uparrow \langle \downarrow \langle \uparrow \alpha_5 \rangle \alpha_6 \langle \uparrow \alpha_7 \rangle \rangle \alpha_8 \langle \downarrow \langle \uparrow \alpha_9 \rangle \alpha_{10} \langle \uparrow \alpha_{11} \rangle \rangle \rangle. \quad (4.32)$$

Lemma 4.22 directly relates the minimality of a DNA expression E to the minimality of the DNA subexpressions of E . For operator-minimal DNA expressions, we have a weaker result. Its proof is similar to the second part of the proof of Lemma 4.22:

Lemma 4.64 *If a DNA expression E is operator-minimal, then each proper DNA subexpression of E is minimal.*

This result cannot be reversed. It is not sufficient for a DNA expression to be operator-minimal that all its proper DNA subexpressions are minimal. For example, the DNA expression $E = \langle \uparrow \langle \uparrow \alpha_1 \rangle \rangle$ has only one proper DNA subexpression: $E^s = \langle \uparrow \alpha_1 \rangle$. It follows from Theorem 4.53(1) that E^s is minimal, whereas E is clearly not operator-minimal.

The following result specifies how to construct an operator-minimal \uparrow -expression or \downarrow -expression for a nick free formal DNA molecule:

Theorem 4.65 *Let X be a nick free formal DNA molecule and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X .*

1. *Let $Y_0 X_1 Y_1 X_2 Y_2 \dots X_r Y_r$ for some $r \geq 0$ be an arbitrary maximal upper partitioning of X , where for $j = 0, \dots, r$, $Y_j = x'_{a_j} \dots x'_{b_j}$ for some a_j and b_j . Further, let*

$$E = \langle \uparrow \varepsilon_{a_0} \dots \varepsilon_{b_0} E_1 \varepsilon_{a_1} \dots \varepsilon_{b_1} E_2 \varepsilon_{a_2} \dots \varepsilon_{b_2} \dots E_r \varepsilon_{a_r} \dots \varepsilon_{b_r} \rangle,$$

where for all applicable i ,

$$\varepsilon_i = \begin{cases} \alpha_i & \text{if } x'_i = \binom{\alpha_i}{-} \text{ for an } \mathcal{N}\text{-word } \alpha_i \\ \langle \downarrow \alpha_i \rangle & \text{if } x'_i = \binom{\alpha_i}{c(\alpha_i)} \text{ for an } \mathcal{N}\text{-word } \alpha_i \end{cases} \quad (4.33)$$

and for $j = 1, \dots, r$, E_j is a minimal DNA expression denoting X_j . Then E is an operator-minimal \uparrow -expression denoting X and

$$|E| = 3 + 3 \cdot T_{\uparrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|. \quad (4.34)$$

2. *Let $Y_0 X_1 Y_1 X_2 Y_2 \dots X_r Y_r$ for some $r \geq 0$ be an arbitrary maximal lower partitioning of X , where for $j = 0, \dots, r$, $Y_j = x'_{a_j} \dots x'_{b_j}$ for some a_j and b_j . Further, let*

$$E = \langle \downarrow \varepsilon_{a_0} \dots \varepsilon_{b_0} E_1 \varepsilon_{a_1} \dots \varepsilon_{b_1} E_2 \varepsilon_{a_2} \dots \varepsilon_{b_2} \dots E_r \varepsilon_{a_r} \dots \varepsilon_{b_r} \rangle,$$

where for all applicable i ,

$$\varepsilon_i = \begin{cases} \alpha_i & \text{if } x'_i = \binom{-}{\alpha_i} \text{ for an } \mathcal{N}\text{-word } \alpha_i \\ \langle \downarrow \alpha_i \rangle & \text{if } x'_i = \binom{\alpha_i}{c(\alpha_i)} \text{ for an } \mathcal{N}\text{-word } \alpha_i \end{cases}$$

and for $j = 1, \dots, r$, E_j is a minimal DNA expression denoting X_j . Then E is an operator-minimal \downarrow -expression denoting X and

$$|E| = 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\downarrow}(X) + |\nu(X)|.$$

By Theorem 4.53, we know how to construct minimal DNA expressions E_j denoting the formal DNA submolecules X_j occurring in both claims. Hence, the specification above is complete.

Note that there are two subtle differences between Theorem 4.65 and Theorem 4.53.

First, we do not demand the formal DNA molecule X to contain at least one single-stranded component. There also exist an operator-minimal \uparrow -expression and an operator-minimal \downarrow -expression denoting $X = \binom{\alpha_1}{c(\alpha_1)}$ for an \mathcal{N} -word α_1 . For this formal DNA molecule, there exists exactly one maximal upper partitioning and exactly one maximal lower partitioning, namely $\mathcal{M} = X$. The corresponding operator-minimal \uparrow -expression is $E = \langle \uparrow \langle \downarrow \alpha_1 \rangle \rangle$ and the corresponding operator-minimal \downarrow -expression is $E = \langle \downarrow \langle \uparrow \alpha_1 \rangle \rangle$.

Second, when we describe an operator-minimal \uparrow -expression (in Claim 1), we do not restrict ourselves to formal DNA molecules X with $T_{\uparrow}(X) \geq T_{\downarrow}(X)$. There exist operator-minimal \uparrow -expressions for every nick free formal DNA molecule. Indeed, we have given operator-minimal \uparrow -expressions for the formal DNA molecule Z_2 from (4.30), for which $T_{\uparrow}(Z_2) < T_{\downarrow}(Z_2)$. Analogously, there exist operator-minimal \downarrow -expressions for every nick free formal DNA molecule (see Claim 2).

Note finally that the operator-minimal \uparrow -expression and the operator-minimal \downarrow -expression described in Theorem 4.65 achieve the lower bounds from Corollary 4.19(1) and (2), respectively.

Indeed, the two operator-minimal \uparrow -expressions we have given in (4.31) and (4.32), which denote the formal DNA molecule Z_2 from (4.30), can be constructed according to the description in Claim 1. Z_2 has one internal maximal upper sequence, viz. $\binom{\alpha_8}{-}$, and hence, by Lemma 4.48, there are two maximal upper partitionings of Z_2 . The first one is $\mathcal{M}' = X'_1 = Z_2$ and the second one is $\mathcal{M}'' = X''_1 Y''_1 X''_2$ with

$$\begin{aligned} X''_1 &= \binom{\alpha_5}{c(\alpha_5)} \binom{-}{\alpha_6} \binom{\alpha_7}{c(\alpha_7)} \\ Y''_1 &= \binom{\alpha_8}{-} \\ X''_2 &= \binom{\alpha_9}{c(\alpha_9)} \binom{-}{\alpha_{10}} \binom{\alpha_{11}}{c(\alpha_{11})} \end{aligned}$$

The DNA expression E'_2 from (4.31) corresponds to \mathcal{M}' and the DNA expression E''_2 from (4.32) corresponds to \mathcal{M}'' .

Proof of Theorem 4.65:

1. First, we can prove that the string E is well defined and that it is a DNA expression denoting X . For this, we refer to the corresponding part of the proof of Theorem 4.53, because that carries over entirely.

We subsequently prove that E has the specified length and (thus) is operator-minimal. Contrary to the proof of Theorem 4.53, we do not need induction for this. Nevertheless, we will see that the main ingredients of the proof are the same.

By Corollary 4.54(1),

$$T_{\downarrow}(X_1) + \cdots + T_{\downarrow}(X_r) = T_{\downarrow}(X).$$

Further, by Corollary 4.52(2), for $j = 1, \dots, r$, $T_{\uparrow}(X_j) = T_{\downarrow}(X_j) - 1$.

When we apply Theorem 4.53(2), we find that a minimal DNA expression E_j denoting a substring X_j has length

$$|E_j| = 3 + 3 \cdot T_{\uparrow}(X_j) + 3 \cdot n_{\uparrow}(X_j) + |\nu(X_j)| = 3 \cdot T_{\downarrow}(X_j) + 3 \cdot n_{\uparrow}(X_j) + |\nu(X_j)|$$

We can then calculate the length of E , in the same way as we did in the proof of Theorem 4.53:

$$|E| = \dots = 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|.$$

Because, by Corollary 4.19(1), this equals the lower bound for the length of a \uparrow -expression denoting X , E is operator-minimal.

2. The proof of this claim is analogous to that of the previous claim.

□

By a proof identical to that of Lemma 4.57, we find

Lemma 4.66 *Let X be a nick free formal DNA molecule and let E be an operator-minimal DNA expression denoting X as described in Theorem 4.65.*

1. *Each \mathcal{N} -word-argument of E is a maximal \mathcal{N} -word occurrence in E .*
2. *The arguments of E are \mathcal{N} -words and DNA expressions, alternately.*

We use the operator-minimal DNA expressions from Theorem 4.65 to obtain minimal DNA expressions for expressible formal DNA molecules containing nick letters.

Theorem 4.67 *Let X be an expressible formal DNA molecule which contains at least one lower nick letter \triangle , and let $Z_{1\triangle}Z_{2\triangle}\dots_{\triangle}Z_m$ for some $m \geq 2$ be the nick free decomposition of X .*

For $h = 1, \dots, m$, 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 (hence, if $E_h = \langle \uparrow \varepsilon_{h,1} \dots \varepsilon_{h,n_h} \rangle$ for some $n_h \geq 1$ and \mathcal{N} -words or DNA expressions $\varepsilon_{h,1}, \dots, \varepsilon_{h,n_h}$, then $\widehat{E}_h = \varepsilon_{h,1} \dots \varepsilon_{h,n_h}$).

Then $E = \langle \uparrow \widehat{E}_1 \dots \widehat{E}_m \rangle$ is a minimal DNA expression denoting X and

$$|E| = 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|. \quad (4.35)$$

Also in this case, the (minimal) \uparrow -expression described achieves the lower bound for \uparrow -expressions from Corollary 4.19(1).

Of course, there is an analogous result for expressible formal DNA molecules containing upper nick letters.

We have established before that the nick free decomposition for the formal DNA molecule from Figure 4.9 is $Z_{1\triangle}Z_{2\triangle}Z_{3\triangle}Z_{4\triangle}Z_5$, where Z_1, \dots, Z_5 are given in (4.29). Because none of Z_1, Z_3, Z_4, Z_5 has an internal maximal upper sequence, there exists exactly one maximal upper partitioning for each of them. Hence, Theorem 4.65 specifies one operator-minimal \uparrow -expression for each of them:

$$\begin{aligned} E_1 &= \langle \uparrow \alpha_1 \langle \downarrow \langle \downarrow \alpha_2 \rangle \alpha_3 \langle \downarrow \alpha_4 \rangle \rangle \rangle, \\ E_3 &= \langle \uparrow \langle \downarrow \alpha_{12} \rangle \alpha_{13} \langle \downarrow \alpha_{14} \rangle \alpha_{15} \langle \downarrow \alpha_{16} \rangle \rangle, \\ E_4 &= \langle \uparrow \langle \downarrow \alpha_{17} \rangle \rangle, \\ E_5 &= \langle \uparrow \langle \downarrow \langle \downarrow \alpha_{18} \rangle \alpha_{19} \langle \downarrow \alpha_{20} \rangle \rangle \alpha_{21} \langle \downarrow \alpha_{22} \rangle \rangle. \end{aligned}$$

The formal DNA submolecule Z_2 has one internal maximal upper sequence, giving rise to two different maximal upper partitionings. As we observed before, the DNA expressions E'_2 and E''_2 from (4.31) and (4.32) are the operator-minimal \uparrow -expressions corresponding to these maximal upper partitionings.

To construct a minimal DNA expression denoting the entire formal DNA molecule X , we may arbitrarily choose either of E'_2 and E''_2 . When we choose E''_2 , we obtain

$$\begin{aligned} E = \langle \uparrow & \alpha_1 \langle \downarrow \langle \downarrow \alpha_2 \rangle \alpha_3 \langle \downarrow \alpha_4 \rangle \rangle \langle \downarrow \langle \downarrow \alpha_5 \rangle \alpha_6 \langle \downarrow \alpha_7 \rangle \rangle \alpha_8 \langle \downarrow \langle \downarrow \alpha_9 \rangle \alpha_{10} \langle \downarrow \alpha_{11} \rangle \rangle \\ & \langle \downarrow \langle \downarrow \alpha_{12} \rangle \alpha_{13} \langle \downarrow \alpha_{14} \rangle \alpha_{15} \langle \downarrow \alpha_{16} \rangle \langle \downarrow \alpha_{17} \rangle \\ & \langle \downarrow \langle \downarrow \alpha_{18} \rangle \alpha_{19} \langle \downarrow \alpha_{20} \rangle \rangle \alpha_{21} \langle \downarrow \alpha_{22} \rangle \rangle. \end{aligned} \quad (4.36)$$

Indeed,

$$|E| = 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)| = 3 + 3 \cdot 4 + 3 \cdot 13 + |\nu(X)| = 54 + |\nu(X)|.$$

Note that both $T_{\uparrow}(Z_1) = T_{\downarrow}(Z_1) = 1$ and $T_{\uparrow}(Z_5) = T_{\downarrow}(Z_5) = 1$. Hence, by Theorem 4.53(2), E_1 and E_5 are not only operator-minimal, but also ('absolutely') minimal, and there also exist minimal \downarrow -expressions denoting Z_1 and Z_5 . In the current context, however, we must choose the (operator-)minimal \uparrow -expressions.

Proof of Theorem 4.67: By Theorem 3.4, because X is expressible and it contains at least one lower nick letter \triangle , it does not contain any upper nick letter. Hence, each nick letter y_h in the definition of the nick free decomposition is a lower nick letter. Indeed, the nick free decomposition of X is $Z_{1\triangle}Z_{2\triangle}\dots_{\triangle}Z_m$ with $m \geq 2$.

By definition, each Z_h is nick free. Hence, by Theorem 4.65, there indeed exists an operator-minimal \uparrow -expression E_h denoting Z_h .

By the definition of a formal DNA molecule, a nick letter may occur only between two elements from \mathcal{A}_{\pm} . Hence, for $h = 1, \dots, m-1$, $R(Z_h), L(Z_{h+1}) \in \mathcal{A}_{\pm}$. Consequently, the DNA expressions E_1, \dots, E_m fit together by upper strands (so that $\langle \uparrow E_1 \dots E_m \rangle$ is a DNA expression), and

$$\mathcal{S}(\langle \uparrow E_1 E_2 \dots E_m \rangle) = \nu^+(Z_1)_{\triangle} \nu^+(Z_2)_{\triangle} \dots_{\triangle} \nu^+(Z_m).$$

Because Z_1, \dots, Z_m are nick free, this is equal to X . By Lemma 3.6, also $E = \langle \uparrow \hat{E}_1 \dots \hat{E}_m \rangle$ is a DNA expression denoting X .

By Theorem 4.65, for $h = 1, \dots, m$,

$$|E_h| = 3 + 3 \cdot T_{\downarrow}(Z_h) + 3 \cdot n_{\uparrow}(Z_h) + |\nu(Z_h)|,$$

and thus

$$|\hat{E}_h| = 3 \cdot T_{\downarrow}(Z_h) + 3 \cdot n_{\uparrow}(Z_h) + |\nu(Z_h)|. \quad (4.37)$$

Since each Z_h is nick free, we certainly have $\#_{\nabla}(Z_h) = 0$. Now when we apply equations (4.2), (4.3) and (4.4) from Lemma 4.16(1) to $\langle \uparrow E_1 E_2 \dots E_m \rangle$, we find

$$T_{\downarrow}(Z_1) + \dots + T_{\downarrow}(Z_m) = T_{\downarrow}(X) \quad \text{and} \quad (4.38)$$

$$n_{\uparrow}(Z_1) + \dots + n_{\uparrow}(Z_m) = n_{\uparrow}(X) \quad (4.39)$$

We use (4.37), (4.38) and (4.39) to calculate $|E|$:

$$\begin{aligned} |E| &= 3 + \sum_{h=1}^m |\hat{E}_h| \\ &= 3 + \sum_{h=1}^m (3 \cdot T_{\downarrow}(Z_h) + 3 \cdot n_{\uparrow}(Z_h) + |\nu(Z_h)|) \\ &= 3 + 3 \cdot \sum_{h=1}^m T_{\downarrow}(Z_h) + 3 \cdot \sum_{h=1}^m n_{\uparrow}(Z_h) + \sum_{h=1}^m |\nu(Z_h)| \\ &= 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|. \end{aligned}$$

By Corollary 4.19(1), this is the minimal length of a \uparrow -expression denoting X . Then by Lemma 4.60(1), E is a minimal DNA expression for X . \square

One may wonder if we really need the concept of operator-minimality in the construction of minimal DNA expressions for formal DNA molecules containing nick letters.

Let X be a formal DNA molecule, with nick free decomposition $Z_{1\Delta}Z_{2\Delta}\dots_{\Delta}Z_m$ for some $m \geq 2$. Then a minimal DNA expression denoting X might also be constructed by (1) determining minimal DNA expressions E_1, \dots, E_m denoting the nick free formal DNA submolecules Z_1, \dots, Z_m , respectively, (2) defining the \uparrow -expression $\langle \uparrow E_1 \dots E_m \rangle$ with these minimal DNA expressions as arguments, and (3) removing redundant operators \uparrow according to Lemma 3.6, i.e. replacing those DNA expressions E_h that are themselves \uparrow -expressions by their respective arguments. In order to make step (3) as effective as possible, we should in step (1) choose for \uparrow -expressions E_h whenever we can. In particular, if, for some h with $1 \leq h \leq m$, Z_h contains at least one single-stranded component and $T_{\uparrow}(Z_h) = T_{\downarrow}(Z_h)$, then E_h should be a \uparrow -expression as specified by Theorem 4.53(1) (and not a \downarrow -expression as specified by Theorem 4.53(2)).

Let us apply this alternative method to the formal DNA molecule X from Figure 4.9. The nick free decomposition of X is $Z_{1\Delta}Z_{2\Delta}Z_{3\Delta}Z_{4\Delta}Z_5$, where Z_1, \dots, Z_5 are given in (4.29).

As we observed before, each minimal DNA expression denoting $Z_2 = \binom{\alpha_5}{c(\alpha_5)} \binom{-}{\alpha_6} \cdot \binom{\alpha_7}{c(\alpha_7)} \binom{\alpha_8}{-} \binom{\alpha_9}{c(\alpha_9)} \binom{-}{\alpha_{10}} \binom{\alpha_{11}}{c(\alpha_{11})}$ is a \downarrow -expression. According to the alternative method, such a \downarrow -expression appears unchanged in a minimal DNA expression. However, in the minimal DNA expression E denoting X from (4.36), the arguments corresponding to Z_2 are $\langle \downarrow \langle \uparrow \alpha_5 \rangle \alpha_6 \langle \uparrow \alpha_7 \rangle \rangle$, α_8 and $\langle \downarrow \langle \uparrow \alpha_9 \rangle \alpha_{10} \langle \uparrow \alpha_{11} \rangle \rangle$, which is not just one \downarrow -expression.

Hence, E cannot be obtained by the alternative method, whereas it can be obtained by the construction from Theorem 4.67, which is based on operator-minimal \uparrow -expressions.

There does not exist an analogue of Corollary 4.56 for the case with nicks. For example, it is not true in general that a minimal DNA expression E denoting a formal DNA molecule X with at least one nick letter satisfies

$$|E| \leq 3 + 3 \cdot T_{\uparrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|. \tag{4.40}$$

Consider, for example, the formal DNA molecule

$$X = \binom{\alpha_1}{c(\alpha_1)} \binom{-}{\alpha_2} \binom{\alpha_3}{c(\alpha_3)} \triangle \binom{\alpha_4}{c(\alpha_4)} \binom{-}{\alpha_5} \binom{\alpha_6}{c(\alpha_6)},$$

where $\alpha_1, \dots, \alpha_6$ are arbitrary \mathcal{N} -words. When we count the components of X , we find that $T_{\uparrow}(X) = 1$, $T_{\downarrow}(X) = 2$ and $n_{\uparrow}(X) = 4$. By Theorem 4.67,

$$E = \langle \uparrow \langle \downarrow \langle \uparrow \alpha_1 \rangle \alpha_2 \langle \uparrow \alpha_3 \rangle \rangle \langle \downarrow \langle \uparrow \alpha_4 \rangle \alpha_5 \langle \uparrow \alpha_6 \rangle \rangle \rangle$$

is a minimal DNA expression denoting X , and

$$\begin{aligned} |E| &= 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)| = 21 + |\nu(X)| \\ &> 3 + 3 \cdot T_{\uparrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)| = 18 + |\nu(X)|. \end{aligned}$$

The violation of (4.40) is due to the lower nick letter occurring in X . The fact that $T_{\downarrow}(X) > T_{\uparrow}(X)$ suggests that a minimal DNA expression for X should be a \downarrow -expression. This is, however, impossible, because by Lemma 3.2(1), a formal DNA molecule containing a lower nick letter cannot be denoted by a \downarrow -expression.

We finally consider the arguments of a minimal DNA expression denoting a formal DNA molecule containing (lower) nick letters. In one respect, they resemble the arguments of (operator-)minimal DNA expressions denoting nick free formal DNA molecules, which we considered in Lemma 4.57 and Lemma 4.66, in one respect they do not.

Lemma 4.68 *Let X be an expressible formal DNA molecule which contains at least one lower nick letter Δ , and let $Z_{1\Delta}Z_{2\Delta}\dots_{\Delta}Z_m$ for some $m \geq 2$ be the nick free decomposition of X . For $h = 1, \dots, m$, let E_h be an operator-minimal \uparrow -expression denoting Z_h , and let E be the minimal DNA expression denoting X which is based on E_1, \dots, E_h as described in Theorem 4.67.*

1. *For $h = 1, \dots, m$, each argument of E_h that is a maximal \mathcal{N} -word occurrence in E_h is also a maximal \mathcal{N} -word occurrence in E . Hence, if for $h = 1, \dots, m$, each \mathcal{N} -word-argument of E_h is a maximal \mathcal{N} -word occurrence in E_h , then also each \mathcal{N} -word-argument of E is a maximal \mathcal{N} -word occurrence in E .*
2. *There exist (at least) two consecutive arguments of E which are DNA expressions. Hence, the arguments of E are not \mathcal{N} -words and DNA expressions, alternately.*

Proof: Consider any h with $1 \leq h \leq m - 1$. By definition, the lower nick letter between Z_h and Z_{h+1} is both preceded and succeeded in X by a double \mathcal{A} -letter: $R(Z_h), L(Z_{h+1}) \in \mathcal{A}_\pm$. This implies that both the last component of Z_h and the first component of Z_{h+1} are double components.

If the last argument of E_h (which denotes Z_h) were an \mathcal{N} -word, then the last component of Z_h would be an upper component. Hence, the last argument of E_h is a DNA expression. Analogously, the first argument of E_{h+1} is a DNA expression.

1. For some h with $1 \leq h \leq m$, consider an argument α of E_h that is a maximal \mathcal{N} -word occurrence in E_h . By the construction from Theorem 4.67, the arguments of E are exactly the arguments of E_1, \dots, E_m . In particular, α is also an argument of E .

If α is the first argument of E_h , then h must be equal to 1 and α is also the first argument of E , which is preceded by the outermost operator \uparrow of E . If α is not the first argument of E_h , then the argument preceding it in E_h is a DNA expression. Obviously, this argument also precedes α in E .

In neither case, α is preceded in E by another \mathcal{N} -word. Analogously, we can prove that α is not succeeded in E by another \mathcal{N} -word. Hence, α is a maximal \mathcal{N} -word occurrence in E .

Now, assume that for $h = 1, \dots, m$, each \mathcal{N} -word-argument of E_h is a maximal \mathcal{N} -word occurrence in E_h . Consider any \mathcal{N} -word-argument α of E . By construction, α is an argument of E_h for some h with $1 \leq h \leq m$. By assumption, it is a maximal \mathcal{N} -word occurrence in E_h . As we have just proved, α is also a maximal \mathcal{N} -word occurrence in E .

2. Consider any h with $1 \leq h \leq m - 1$. By the construction from Theorem 4.67, the last argument of E_h and the first argument of E_{h+1} are consecutive arguments of E . As we have observed at the beginning of the proof, both arguments are DNA expressions.

□

4.2.3 All minimal DNA expressions for a formal DNA molecule

By Theorem 4.23, the only minimal DNA expression denoting the formal DNA molecule $\binom{\alpha_1}{c(\alpha_1)}$ for an \mathcal{N} -word α_1 is $\langle \uparrow \alpha_1 \rangle$. For such a formal DNA molecule, we thus have a (trivial) characterization of its minimal DNA expression. We also want to characterize the minimal DNA expressions for other types of expressible formal DNA molecules, hence expressible formal DNA molecules containing single-stranded components and/or nick letters.

Let X be such a formal DNA molecule. Theorem 4.53 or Theorem 4.67 (depending on whether or not X contains nick letters) provides a way to construct a minimal DNA expression denoting X . There may, however, be other minimal DNA expression for the same molecule, DNA expressions which do not fit the description given in Theorem 4.53 or Theorem 4.67

By Corollary 4.24, we know that each minimal DNA expression denoting X is a \uparrow -expression or a \downarrow -expression. In many cases, the outermost operator of the minimal DNA expression(s) is completely determined. If X is nick free and $T_\uparrow(X) \neq T_\downarrow(X)$, then the outermost operator is determined by Lemma 4.58. If X contains nick letters, then the outermost operator is determined by Lemma 4.60. This is, however, not by far a *characterization* of the minimal DNA expressions denoting X .

To achieve a characterization, we will investigate the structure of arbitrary minimal \uparrow -expressions and arbitrary minimal \downarrow -expressions. Because results on \uparrow -expressions and results on \downarrow -expressions are completely analogous and can be proved in a completely analogous way, we will only give the results for the \uparrow -expressions.

Initially, we consider *operator-minimal* \uparrow -expressions rather than minimal \uparrow -expressions. However, because minimal DNA expressions are in particular operator-minimal, the results we achieve for operator-minimal \uparrow -expressions are certainly valid for minimal \uparrow -expressions.

We start with a simple result:

Lemma 4.69 *Let E be an operator-minimal \uparrow -expression denoting a certain formal DNA molecule X (which may contain nick letters).*

Then no argument of E is a \uparrow -expression.

Proof: Assume that $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$ for some $n \geq 1$ and \mathcal{N} -words or DNA expressions $\varepsilon_1, \dots, \varepsilon_n$. Suppose that for some i , the argument ε_i is a \uparrow -expression $E_i = \langle \uparrow \varepsilon_{i,1} \dots \varepsilon_{i,n_i} \rangle$ for some $n_i \geq 1$ and arguments $\varepsilon_{i,1}, \dots, \varepsilon_{i,n_i}$. Then by Lemma 3.6, the DNA expression

$$E' = \langle \uparrow \varepsilon_1 \dots \varepsilon_{i-1} \varepsilon_{i,1} \dots \varepsilon_{i,n_i} \varepsilon_{i+1} \dots \varepsilon_n \rangle,$$

which is 3 letters shorter than E and has the same outermost operator, is equivalent to E . This contradicts the operator-minimality of E . \square

If an argument of an operator-minimal DNA expression is itself a DNA expression, then it must be minimal. Hence, when we combine Theorem 4.23 and Lemma 4.69, we find

Corollary 4.70 *Let E be an operator-minimal \uparrow -expression denoting a certain formal DNA molecule X (which may contain nick letters).*

Then each argument of E is either an \mathcal{N} -word α , or a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α , or a \downarrow -expression.

Whether or not the formal DNA molecule denoted by a \uparrow -expression E contains nick letters, there may be nick letters in its argument(s). In that case, however, E is not operator-minimal, let alone minimal, as follows from the next result.

Lemma 4.71 *Let $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$, where $n \geq 1$ and $\varepsilon_1, \dots, \varepsilon_n$ are \mathcal{N} -words or DNA expressions, be an operator-minimal DNA expression denoting a certain formal DNA molecule X (which may contain nick letters).*

Then for $i = 1, \dots, n$, $X_i = \mathcal{S}^+(\varepsilon_i)$ is nick free.

Proof: For $i = 1, \dots, n$, let $X_i = \mathcal{S}^+(\varepsilon_i)$. Then

$$X = \nu^+(X_1)y_1\nu^+(X_2)y_2\dots y_{n-1}\nu^+(X_n), \quad (4.41)$$

where for $i = 1, \dots, n-1$, y_i is either equal to λ or to Δ , depending on $R(X_i)$ and $L(X_{i+1})$ (see Definition 2.8).

Consider an argument ε_i for some i with $1 \leq i \leq n$. By Corollary 4.70, ε_i is either an \mathcal{N} -word α , or a \uparrow -expression $\langle \uparrow \alpha \rangle$ for an \mathcal{N} -word α , or a \downarrow -expression.

If ε_i is an \mathcal{N} -word α , then $X_i = \mathcal{S}^+(\varepsilon_i) = \binom{\alpha}{-}$, which is indeed nick free.

If ε_i is a \uparrow -expression $\langle \uparrow \alpha \rangle$ for an \mathcal{N} -word α , then $X_i = \mathcal{S}(\varepsilon_i) = \binom{\alpha}{c(\alpha)}$, which is also nick free.

Now, assume that ε_i is a \downarrow -expression E_i . By Lemma 3.2(1), $X_i = \mathcal{S}(E_i)$ does not contain lower nick letters. Consequently, $\nu^+(X_i)$ does not contain any nick letters.

By Corollary 4.19(2),

$$|E_i| \geq 3 + 3 \cdot T_{\uparrow}(X_i) + 3 \cdot n_{\uparrow}(X_i) + |\nu(X_i)|, \quad (4.42)$$

and by Lemma 4.13(1),

$$T_{\uparrow}(X_i) \geq T_{\uparrow}(\nu^+(X_i)). \quad (4.43)$$

Suppose that X_i is not nick free. Then it must contain at least one upper nick letter: $\#_{\nabla}(X_i) \geq 1$. Hence, by Lemma 4.13(4),

$$n_{\uparrow}(X_i) > n_{\uparrow}(\nu^+(X_i)). \quad (4.44)$$

Substituting (4.43) and (4.44) into (4.42), we obtain

$$|E_i| > 3 + 3 \cdot T_{\uparrow}(\nu^+(X_i)) + 3 \cdot n_{\uparrow}(\nu^+(X_i)) + |\nu(X_i)|.$$

On the other hand, because $\nu^+(X_i)$ is nick free, we know by Corollary 4.56(2) that a minimal DNA-expression E'_i denoting $\nu^+(X_i)$ satisfies

$$|E'_i| \leq 3 + 3 \cdot T_{\uparrow}(\nu^+(X_i)) + 3 \cdot n_{\uparrow}(\nu^+(X_i)) + |\nu(X_i)|$$

(obviously, $\nu(X_i) = \nu(\nu^+(X_i))$).

Now, if we replace the argument E_i of E by the shorter DNA expression E'_i , which satisfies $E_i \nabla \equiv E'_i$, then by Lemma 3.7, the resulting string E' is a DNA expression satisfying $E \nabla \equiv E'$. Because $\nu^+(\mathcal{S}^+(E'_i)) = \nu^+(\nu^+(X_i)) = \nu^+(X_i)$ and by Lemma 2.7, $L(\nu^+(X_i)) = L(X_i)$ and $R(\nu^+(X_i)) = R(X_i)$, we even have $\mathcal{S}(E') = X$ (see (4.41)). In other words, $E \equiv E'$.

Because E' is shorter than E and has the same outermost operator as E , E would not be operator-minimal, which contradicts our assumption. Hence, also in the case that E_i is a \downarrow -expression, must X_i be nick free. \square

Lemma 4.71 immediately implies:

Corollary 4.72 *Let E be an operator-minimal \uparrow -expression denoting a certain formal DNA molecule X (which may contain nick letters). Then each nick letter occurring in X has been introduced by the outermost operator \uparrow of E .*

Since the arguments of an operator-minimal \uparrow -expression are nick free, its semantics as given by Definition 2.8 can be simplified.

Corollary 4.73 *Let $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$, where $n \geq 1$ and $\varepsilon_1, \dots, \varepsilon_n$ are \mathcal{N} -words or DNA expressions, be an operator-minimal DNA expression denoting a certain formal DNA molecule X . For $i = 1, \dots, n$, let $X_i = \mathcal{S}^+(\varepsilon_i)$.*

1.
$$X = X_1 y_1 X_2 y_2 \dots y_{n-1} X_n,$$

where for $i = 1, \dots, n-1$, $y_i = \triangle$ if $R(X_i), L(X_{i+1}) \in \mathcal{A}_\pm$, and $y_i = \lambda$ otherwise.

2. If X is nick free, then

$$X = X_1 X_2 \dots X_n.$$

Proof:

1. By the definition of the semantics of a \uparrow -expression,

$$X = \nu^+(X_1) y_1 \nu^+(X_2) y_2 \dots y_{n-1} \nu^+(X_n), \quad (4.45)$$

where for $i = 1, \dots, n-1$, $y_i = \triangle$ or $y_i = \lambda$ as specified in the claim.

By Lemma 4.71, for $i = 1, \dots, n$, X_i is nick free, and hence $\nu^+(X_i) = X_i$. Now, the claim follows immediately.

2. If X is nick free, then all y_i 's in Claim 1 are equal to λ .

□

In Corollary 4.52 and Corollary 4.54(1), we considered the value of the function T_\downarrow for the formal DNA submolecules X_j occurring in a maximal upper partitioning. We now do the same for the formal DNA submolecules corresponding to the arguments of an operator-minimal \uparrow -expression.

Lemma 4.74 *Let $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$, where $n \geq 1$ and $\varepsilon_1, \dots, \varepsilon_n$ are \mathcal{N} -words or DNA expressions, be an operator-minimal DNA expression denoting a certain formal DNA molecule X (which may contain nick letters). For $i = 1, \dots, n$, let $X_i = \mathcal{S}^+(\varepsilon_i)$.*

1. For $i = 1, \dots, n$, if ε_i is a \downarrow -expression E_i , then $X_i = \mathcal{S}(E_i)$ and $T_\uparrow(X_i) = T_\downarrow(X_i) - 1$. Hence, X_i contains at least one single-stranded component and both the first single-stranded component and the last single-stranded component of X_i are lower components.

2.
$$\sum_{\downarrow\text{-expr. } \varepsilon_i} T_\downarrow(X_i) = T_\downarrow(X_1) + \dots + T_\downarrow(X_n) = T_\downarrow(X).$$

Proof:

1. Consider an argument ε_i that is a \downarrow -expression E_i . By definition, $X_i = \mathcal{S}(E_i)$. By Lemma 4.71, X_i is nick free. Because E is operator-minimal, E_i is minimal.

Suppose that $T_{\uparrow}(X_i) \geq T_{\downarrow}(X_i)$. Then by Corollary 4.19(2),

$$\begin{aligned} |E_i| &\geq 3 + 3 \cdot T_{\uparrow}(X_i) + 3 \cdot n_{\uparrow}(X_i) + |\nu(X_i)| \\ &\geq 3 + 3 \cdot T_{\downarrow}(X_i) + 3 \cdot n_{\uparrow}(X_i) + |\nu(X_i)|. \end{aligned}$$

- If $T_{\uparrow}(X_i) = 0$, then also $T_{\downarrow}(X_i) = 0$, and by Lemma 4.10, X_i does not contain any single-stranded component. Hence, by Lemma 4.25, $X_i = \binom{\alpha}{c(\alpha)}$ for an \mathcal{N} -word α . This, however, leads to a contradiction, because by Theorem 4.23, the unique minimal DNA expression denoting $X_i = \binom{\alpha}{c(\alpha)}$ is $E'_i = \langle \downarrow \alpha \rangle$.
- If $T_{\uparrow}(X_i) \geq 1$, then X_i contains at least one upper component. By Theorem 4.53(1), there exists a minimal \uparrow -expression E'_i denoting X_i for which

$$|E'_i| = 3 + 3 \cdot T_{\downarrow}(X_i) + 3 \cdot n_{\uparrow}(X_i) + |\nu(X_i)|.$$

Hence, E'_i is at most as long as E_i . Because $E_i \equiv E'_i$, we can replace E_i in E by E'_i . By Lemma 3.7, the resulting overall string E' is a DNA expression satisfying $E \equiv E'$. Obviously, E' is at most as long as E and has the same outermost operator, which implies that E' is operator-minimal, just like E .

This, however, contradicts Lemma 4.69, because E' is a \uparrow -expression and one of its arguments, E'_i , is also a \uparrow -expression.

Both if $T_{\uparrow}(X_i) = 0$ and if $T_{\uparrow}(X_i) \geq 1$, we end up in a contradiction. This implies that $T_{\uparrow}(X_i) < T_{\downarrow}(X_i)$. By Lemma 4.11(2), $T_{\uparrow}(X_i) = T_{\downarrow}(X_i) - 1$. Hence, X_i contains at least one lower component and by Lemma 4.12, both the first single-stranded component and the last single-stranded component of X_i are lower components.

2. Consider an argument ε_i for some i with $1 \leq i \leq n$.

By Corollary 4.70, ε_i is either an \mathcal{N} -word α , or a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α , or a \uparrow -expression. If ε_i is an \mathcal{N} -word α (in which case $X_i = \binom{\alpha}{-}$) or a \uparrow -expression $\langle \uparrow \alpha \rangle$ ($X_i = \binom{\alpha}{c(\alpha)}$), then obviously $T_{\downarrow}(X_i) = 0$. This gives us the first equality in the claim.

By Lemma 4.71, $X_i = \mathcal{S}^+(\varepsilon_i)$ is nick free. In particular, $\#_{\nabla}(X_i) = 0$. But then the second equality in the claim follows immediately from inequalities (4.2) and (4.3) from Lemma 4.16(1).

□

If a \uparrow -expression or a \downarrow -expression has two or more consecutive \mathcal{N} -word-arguments, then these arguments may be substituted by one argument being the concatenation of the \mathcal{N} -words. This substitution does not change the semantics of the DNA expression. In fact, the original DNA expression and the new DNA expression cannot even be

distinguished from each other, unless one explicitly indicates what the arguments of each of them is. Hence, we do not lose generality when we assume that the arguments of a \uparrow -expression or a \downarrow -expression are either maximal \mathcal{N} -word occurrences or DNA expressions.

Indeed, by Lemma 4.57(1) and Lemma 4.66(1), the \mathcal{N} -words α_i that we specified as arguments for the (operator-)minimal DNA expressions in Theorem 4.53 and Theorem 4.65 are maximal \mathcal{N} -word occurrences.

In Theorem 4.67, we use operator-minimal DNA expressions to construct a minimal DNA expression E for a formal DNA molecule containing nick letters. By Lemma 4.68(1), the arguments of any operator-minimal DNA expression that are maximal \mathcal{N} -word occurrences remain maximal \mathcal{N} -word occurrences in E .

We now examine the relation between components of the formal DNA molecule X denoted by an operator-minimal \uparrow -expression E and the components of the arguments of E .

Lemma 4.75 *Let $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$, where $n \geq 1$ and $\varepsilon_1, \dots, \varepsilon_n$ are maximal \mathcal{N} -word occurrences or DNA expressions, be an operator-minimal DNA expression denoting a certain formal DNA molecule X (which may contain nick letters). For $i = 1, \dots, n$, let $X_i = \mathcal{S}^+(\varepsilon_i)$, and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X . Then for $i = 1, \dots, n$,*

1. *each component of X_i is also a component of X ;*
2. *there exist a_i and b_i with $1 \leq a_i \leq b_i \leq k$ such that $X_i = x'_{a_i} \dots x'_{b_i}$.*

Proof:

1. By Corollary 4.73(1),

$$X = X_1 y_1 X_2 y_2 \dots y_{n-1} X_n, \quad (4.46)$$

where for $i = 1, \dots, n-1$, $y_i = \Delta$ if both $R(X_i)$ and $L(X_{i+1})$ are double \mathcal{A} -letters, and $y_i = \lambda$ otherwise.

Obviously, no component of any X_i is split up over different components of X . To complete the proof, we have to demonstrate that no component of any X_i merges into a larger component of X .

Because X satisfies (4.46), it is clear that different components of the same X_i do not merge into the same component of X . By definition, if for some i with $1 \leq i \leq n-1$, $y_i = \Delta$, then it is a component of X by itself. Hence, only if $y_i = \lambda$ and the last component of X_i and the first component of X_{i+1} are of the same type (upper component, lower component or double component) then these two components merge into one component of X . No other component of any X_i merges into a larger component of X .

Let us consider X_i and X_{i+1} with $1 \leq i \leq n-1$ such that $y_i = \lambda$. Because the arguments ε_i and ε_{i+1} have to fit together by upper strands, both the last component of X_i and the first component of X_{i+1} cannot be lower components. Because $y_i = \lambda$, the two components cannot both be double components, either. At least one of them has to be an upper component.

Without loss of generality, assume that the last component of X_i is an upper component. By Corollary 4.70 and Lemma 4.74(1), ε_i is a maximal \mathcal{N} -word occurrence. By definition, ε_{i+1} is not an \mathcal{N} -word, and thus is either a \uparrow -expression $\langle \uparrow \alpha \rangle$ for an \mathcal{N} -word α , or a \downarrow -expression. In neither case, the first component of X_{i+1} is an upper component. Consequently, the last component of X_i and the first component of X_{i+1} do not merge into one component.

2. This claim follows immediately from the previous claim. □

We temporarily focus on operator-minimal \uparrow -expressions for *nick free* formal DNA molecules.

Lemma 4.76 *Let $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$, where $n \geq 1$ and $\varepsilon_1, \dots, \varepsilon_n$ are maximal \mathcal{N} -word occurrences or DNA expressions, be an operator-minimal DNA expression denoting a certain nick free formal DNA molecule X . For $i = 1, \dots, n$, let $X_i = \mathcal{S}^+(\varepsilon_i)$ and let $x'_{i,1} \dots x'_{i,k_i}$ for some $k_i \geq 1$ be the decomposition of X_i .*

1. *For $i = 1, \dots, n-1$, if ε_i is a \downarrow -expression, then $k_i \geq 2$, x'_{i,k_i-1} is a closing lower component of X , x'_{i,k_i} is a double component of X , ε_{i+1} is a maximal \mathcal{N} -word occurrence α and $X_{i+1} = \binom{\alpha}{-}$ is an opening upper component of X .*
2. *For $i = 2, \dots, n$, if ε_i is a \downarrow -expression, then $k_i \geq 2$, $x'_{i,2}$ is an opening lower component of X , $x'_{i,1}$ is a double component of X , ε_{i-1} is a maximal \mathcal{N} -word occurrence α and $X_{i-1} = \binom{\alpha}{-}$ is a closing upper component of X .*
3. *The arguments $\varepsilon_1, \dots, \varepsilon_n$ of E are maximal \mathcal{N} -word occurrences and DNA expressions, alternately.*

Proof: Claims 1 and 2 are completely analogous and so are their proofs. We give the proof of Claim 2. We first observe, however, that by Corollary 4.73(2),

$$X = X_1 X_2 \dots X_n.$$

2. Assume that ε_i with $2 \leq i \leq n$ is a \downarrow -expression. By Lemma 4.71, X_i is nick free. By Lemma 4.74(1), X_i contains at least one single-stranded component and the first single-stranded component of X_i is a lower component. Because ε_{i-1} has to prefit ε_i by upper strands, $x'_{i,1}$ cannot be a lower component. Then by Corollary 2.5, it is a double component of X_i , $k_i \geq 2$ and $x'_{i,2}$ is a lower component of X_i . By Lemma 4.75(1), $x'_{i,1}$ and $x'_{i,2}$ are also a double component of X and a lower component of X , respectively.

By Corollary 4.70, ε_{i-1} is either a maximal \mathcal{N} -word occurrence α , or a \uparrow -expression $\langle \uparrow \alpha \rangle$ for an \mathcal{N} -word α , or a \downarrow -expression.

If it is a \downarrow -expression, then we can prove that the last component of X_{i-1} is a double component, in the same way that we proved that $x'_{i,1}$ is a double component of X_i . Then, however, the outermost operator \uparrow of E would introduce a lower nick letter between X_{i-1} and X_i , which would contradict the fact that X is nick free. This would also happen if ε_{i-1} were a \uparrow -expression $\langle \uparrow \alpha \rangle$.

Hence, ε_{i-1} is a maximal \mathcal{N} -word occurrence α and $X_{i-1} = \binom{\alpha}{-}$ is an upper \mathcal{A} -word, and thus an upper component of X . Because it immediately precedes $x'_{i,1}x'_{i,2}$ in X , it is a closing upper component of X and $x'_{i,2}$ is an opening lower component of X .

3. Consider an argument ε_i with $1 \leq i \leq n-1$. By Corollary 4.70, ε_i is either a maximal \mathcal{N} -word occurrence α_i , or a \uparrow -expression $\langle \uparrow \alpha_i \rangle$ for an \mathcal{N} -word α_i , or a \downarrow -expression.

If ε_i is a maximal \mathcal{N} -word occurrence in E , then by the definition of a maximal \mathcal{N} -word occurrence, ε_{i+1} is a DNA expression.

If ε_i is a \uparrow -expression $\langle \uparrow \alpha_i \rangle$ for an \mathcal{N} -word α_i , then by Claim 2, ε_{i+1} cannot be a \downarrow -expression. If ε_{i+1} were a \uparrow -expression $\langle \uparrow \alpha_{i+1} \rangle$ for an \mathcal{N} -word α_{i+1} , then the outermost operator \uparrow of E would introduce a nick letter between $X_i = \binom{\alpha_i}{c(\alpha_i)}$ and $X_{i+1} = \binom{\alpha_{i+1}}{c(\alpha_{i+1})}$. Because X is nick free, this is not possible. Hence, ε_{i+1} is a maximal \mathcal{N} -word occurrence.

Finally, if ε_i is a \downarrow -expression, then by Claim 1, ε_{i+1} is a maximal \mathcal{N} -word occurrence.

□

Now we are ready to characterize the minimal \uparrow -expressions denoting a nick free formal DNA molecule.

Theorem 4.77 *Let $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$, where $n \geq 1$ and $\varepsilon_1, \dots, \varepsilon_n$ are maximal \mathcal{N} -word occurrences or DNA expressions, be a minimal DNA expression denoting a certain nick free formal DNA molecule X . For $i = 1, \dots, n$, let $X_i = \mathcal{S}^+(\varepsilon_i)$.*

Let $\varepsilon_{i_1}, \varepsilon_{i_2}, \dots, \varepsilon_{i_r}$, with $0 \leq r \leq n$ and $i_1 < i_2 < \dots < i_r$, be all \downarrow -arguments of E . Finally, let Y_0, Y_1, \dots, Y_r be defined by

$$\begin{aligned} Y_0 &= \begin{cases} X_1 \dots X_n & \text{if } r = 0 \\ X_1 \dots X_{i_1-1} & \text{if } r \geq 1 \end{cases} \\ Y_j &= X_{i_j+1} \dots X_{i_{j+1}-1} \quad (j = 1, \dots, r-1) \\ Y_r &= \begin{cases} X_1 \dots X_n & \text{if } r = 0 \\ X_{i_r+1} \dots X_n & \text{if } r \geq 1 \end{cases} \end{aligned}$$

1. $Y_0 X_{i_1} Y_1 X_{i_2} Y_2 \dots X_{i_r} Y_r$ is a maximal upper partitioning of X .
2. E satisfies the description of a minimal DNA expression denoting X given in Theorem 4.53(1).

Proof: Because a minimal DNA expression is in particular operator-minimal, all earlier results in this subsection are also valid for E .

By Corollary 4.73(2),

$$X = X_1 \dots X_n.$$

By Corollary 4.70, each argument ε_i of E which is not a \downarrow -expression, is either a maximal \mathcal{N} -word occurrence α , or a \uparrow -expression $\langle \uparrow \alpha \rangle$ for an \mathcal{N} -word α . The corresponding formal DNA molecule $X_i = \mathcal{S}^+(\varepsilon_i)$ is $\binom{\alpha}{-}$ or $\binom{\alpha}{c(\alpha)}$, respectively. By Lemma 4.75(1), this is a component of X ; in particular, it is an upper component or a double component of X .

1. We distinguish two cases: $r = 0$ and $r \geq 1$.

- If $r = 0$, hence if no argument ε_i of E is a \downarrow -expression, then for $i = 1, \dots, n$, X_i is an upper component or a double component of X . By Lemma 4.50, the only maximal upper partitioning of X is $\mathcal{M} = X$. Indeed, $Y_0 = X_1 \dots X_n = X$ in this case.

- If $r \geq 1$, then Y_0 is the concatenation of X_i 's preceding the first \downarrow -expression ε_{i_1} , Y_r is the concatenation of X_i 's following the last \downarrow -expression ε_{i_r} and for $j = 1, \dots, r-1$, Y_j is the concatenation of X_i 's between the \downarrow -expressions ε_{i_j} and $\varepsilon_{i_{j+1}}$. Indeed, $Y_0 X_{i_1} Y_1 X_{i_2} Y_2 \dots X_{i_r} Y_r = X_1 \dots X_n = X$. Further, each Y_j is the concatenation of a number of upper components and double components of X .

Let us consider any j with $1 \leq j \leq r-1$. By Lemma 4.76(3), the two \downarrow -expressions ε_{i_j} and $\varepsilon_{i_{j+1}}$ are not consecutive arguments of E . Hence, $i_j + 1 \leq i_{j+1} - 1$ and Y_j is not empty. By Lemma 4.76(1) and (2), Y_j starts with an opening upper component $X_{i_{j+1}}$ of X and ends with a closing upper component $X_{i_{j+1}-1}$ of X . Also, both X_{i_j} and $X_{i_{j+1}}$ contain at least two components of X . Then by definition, $Y_j = X_{i_{j+1}} \dots X_{i_{j+1}-1}$ is a maximal upper sequence of X . In particular, because it is preceded in X by X_{i_j} and followed in X by $X_{i_{j+1}}$, it is an *internal* maximal upper sequence.

If $Y_0 = \lambda$, then $i_1 = 1$, i.e., ε_1 is a \downarrow -expression. By Lemma 4.74(1), X_1 (and thus X) contains at least one single-stranded component and the first single-stranded component of X_1 (and thus of X) is a lower component. But then, by Lemma 4.45(1), the maximal upper prefix of X is empty. Hence, Y_0 is equal to the maximal upper prefix.

If $Y_0 \neq \lambda$, then $i_1 \geq 2$. By Lemma 4.76(2), Y_0 ends with a closing upper component X_{i_1-1} of X and X_{i_1} contains at least two components. This implies that $Y_0 = X_1 \dots X_{i_1-1}$ is a maximal upper sequence of X . It obviously is the first maximal upper sequence. Because it has occurrence $(\lambda, X_{i_1} \dots X_n)$ in X , it is the maximal upper prefix of X .

We have thus proved that Y_0 is equal to the maximal upper prefix of X , both if it is empty and if it is not. Analogously we can prove that Y_r is equal to the maximal upper suffix of X .

Consequently, $Y_0 X_{i_1} Y_1 X_{i_2} Y_2 \dots X_{i_r} Y_r$ is a maximal upper partitioning of X .

2. We first establish that Theorem 4.53(1) is applicable to X .

By assumption, X is nick free. By Theorem 4.23, X is not equal to $\binom{\alpha}{c(\alpha)}$ for an \mathcal{N} -word α . because the only minimal DNA expression denoting such a formal DNA molecule is $\langle \uparrow \alpha \rangle$, whereas E is a \uparrow -expression. Hence, by Lemma 4.25, X contains at least one single-stranded component.

Finally, by Lemma 4.58(2), if $T_{\downarrow}(X) > T_{\uparrow}(X)$, then each minimal DNA expression denoting X would be a \downarrow -expression. Because E is a \uparrow -expression, we must have $T_{\uparrow}(X) \geq T_{\downarrow}(X)$. Indeed, Theorem 4.53(1) applies to X .

By Claim 1, $Y_0 X_{i_1} Y_1 X_{i_2} Y_2 \dots X_{i_r} Y_r$ is a maximal upper partitioning of X . When we define indices a_j and b_j for $j = 0, \dots, r$ by

$$a_0 = 1,$$

$$\begin{aligned}
a_j &= i_j + 1 & (j = 1, \dots, r), \\
b_j &= i_{j+1} - 1 & (j = 0, \dots, r-1), \text{ and} \\
b_r &= n,
\end{aligned}$$

it is easy to verify that for $j = 0, \dots, r$, $Y_j = X_{a_j} \dots X_{b_j}$.

Now consider any argument ε_i of E such that $a_j \leq i \leq b_j$ for some j with $0 \leq j \leq r$, in other words: any ε_i that is not a \downarrow -expression. As we observed before, $X_i = \mathcal{S}^+(\varepsilon_i)$ is an upper component or a double component of X . Then because ε_i is either a maximal \mathcal{N} -word occurrence α , or a \uparrow -expression $\langle \uparrow \alpha \rangle$ for an \mathcal{N} -word α , it satisfies (4.18) from Theorem 4.53(1).

Finally, consider any argument ε_{i_j} with $1 \leq j \leq r$. By definition, ε_{i_j} is a \downarrow -expression denoting X_{i_j} . In particular, because E is minimal, ε_{i_j} is a *minimal* \downarrow -expression denoting X_{i_j} .

□

As all auxiliary results in this subsection deal with operator-minimal \uparrow -expressions, it is only a small step from Theorem 4.77 to a characterization of the operator-minimal \uparrow -expressions denoting a nick free formal DNA molecule.

Theorem 4.78 *Let $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$, where $n \geq 1$ and $\varepsilon_1, \dots, \varepsilon_n$ are maximal \mathcal{N} -word occurrences or DNA expressions, be an operator-minimal DNA expression denoting a certain nick free formal DNA molecule X . For $i = 1, \dots, n$, let $X_i = \mathcal{S}^+(\varepsilon_i)$.*

Let $\varepsilon_{i_1}, \varepsilon_{i_2}, \dots, \varepsilon_{i_r}$, with $0 \leq r \leq n$ and $i_1 < i_2 < \dots < i_r$, be all \downarrow -arguments of E . Finally, let Y_0, Y_1, \dots, Y_r be defined by

$$\begin{aligned}
Y_0 &= \begin{cases} X_1 \dots X_n & \text{if } r = 0 \\ X_1 \dots X_{i_1-1} & \text{if } r \geq 1 \end{cases} \\
Y_j &= X_{i_{j+1}} \dots X_{i_{j+1}-1} & (j = 1, \dots, r-1) \\
Y_r &= \begin{cases} X_1 \dots X_n & \text{if } r = 0 \\ X_{i_r+1} \dots X_n & \text{if } r \geq 1 \end{cases}
\end{aligned}$$

1. $Y_0 X_{i_1} Y_1 X_{i_2} Y_2 \dots X_{i_r} Y_r$ is a maximal upper partitioning of X .
2. E satisfies the description of an operator-minimal \uparrow -expression denoting X given in Theorem 4.65(1).

Proof: The proof of Claim 1 is identical to that of Theorem 4.77(1).

The proof of Claim 2 is a bit shorter than that of Theorem 4.77(2) In order to demonstrate that Theorem 4.65(1) is applicable to X , we do not have to elaborate on single-stranded components occurring in X , nor on $T_\uparrow(X)$ and $T_\downarrow(X)$. We only have to observe that X is nick free. The rest of the proof is identical. □

We finally characterize the minimal \uparrow -expressions for a formal DNA molecule X containing (lower) nick letters. The characterization in fact reduces to a characterization of operator-minimal \uparrow -expressions denoting nick free formal DNA submolecules of X , which is provided by Theorem 4.78.

Theorem 4.79 Let $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$, where $n \geq 1$ and $\varepsilon_1, \dots, \varepsilon_n$ are maximal \mathcal{N} -word occurrences or DNA expressions, be a minimal DNA expression denoting a certain formal DNA molecule X which contains at least one nick letter. Let $Z_{1\Delta}Z_{2\Delta} \dots \Delta Z_m$ for some $m \geq 2$ be the nick free decomposition of X .

Then E satisfies the description of a minimal DNA expression denoting X given in Theorem 4.67. Hence, there exist indices i_0, i_1, \dots, i_m , such that

- $i_0 = 0 < i_1 < i_2 < \dots < i_{m-1} < i_m = n$, and
- for $h = 1, \dots, m$, $\langle \uparrow \varepsilon_{i_{h-1}+1} \dots \varepsilon_{i_h} \rangle$ is an operator-minimal \uparrow -expression denoting Z_h .

Note that the indices i_0, i_1, \dots, i_m are unique. Suppose that j_0, j_1, \dots, j_m is another sequence of indices that satisfies the conditions, and that h_0 is the smallest value for which $i_{h_0} \neq j_{h_0}$. As $i_0 = j_0 = 0$, $h_0 \geq 1$. Then, however, both $\langle \uparrow \varepsilon_{i_{h_0-1}+1} \dots \varepsilon_{i_{h_0}} \rangle$ and $\langle \uparrow \varepsilon_{j_{h_0-1}+1} \dots \varepsilon_{j_{h_0}} \rangle = \langle \uparrow \varepsilon_{i_{h_0-1}+1} \dots \varepsilon_{j_{h_0}} \rangle$ would denote Z_{h_0} . This is impossible, because by definition each argument of a DNA expression contributes at least an \mathcal{A} -letter to the semantics.

Proof: By assumption, X contains at least one nick letter, and by Lemma 3.1(1), X does not contain upper nick letters. Hence, indeed $m \geq 2$ and each nick letter occurring in the nick free decomposition of X is a lower nick letter.

For $i = 1, \dots, n$, let $X_i = \mathcal{S}^+(\varepsilon_i)$. By Corollary 4.73(1),

$$X = X_1 y'_1 X_2 y'_2 \dots y'_{n-1} X_n, \quad (4.47)$$

where for $i = 1, \dots, n-1$, y'_i is either equal to λ or to Δ , depending on $R(X_i)$ and $L(X_{i+1})$.

On the other hand, we have

$$X = Z_{1\Delta}Z_{2\Delta} \dots \Delta Z_m.$$

Because by Lemma 4.71, the X_i 's themselves are nick free, the lower nick letters occurring in the nick free decomposition of X do not occur in them. Each of the lower nick letters must correspond to a y'_i in (4.47). More formally, there exist indices i_1, i_2, \dots, i_{m-1} such that

- $1 \leq i_1 < i_2 < \dots < i_{m-1} \leq n-1$,
- for $h = 1, \dots, m-1$, $y'_{i_h} = \Delta$ and the occurrence

$$(X_1 y'_1 \dots y'_{i_h-1} X_{i_h}, X_{i_h+1} y'_{i_h+1} \dots y'_{n-1} X_n) \quad (4.48)$$

of y'_{i_h} in X is equal to the occurrence

$$(Z_{1\Delta} \dots \Delta Z_h, Z_{h+1\Delta} \dots \Delta Z_m) \quad (4.49)$$

of Δ in X , and

- for $i \in \{1, \dots, n\} \setminus \{i_1, \dots, i_{m-1}\}$, $y'_i = \lambda$.

Let us define $i_0 = 0$ and $i_m = n$, and consider any h with $1 \leq h \leq m$. It is clear from (4.48) and (4.49) that

$$Z_h = X_{i_{h-1}+1} y'_{i_{h-1}+1} \cdots y'_{i_h-1} X_{i_h},$$

which is equal to $X_{i_{h-1}+1} X_{i_{h-1}+2} \cdots X_{i_h}$, because every y'_i with $i_{h-1} + 1 \leq i \leq i_h - 1$ is equal to λ . Note that by Lemma 4.62(1) $Z_h \neq \lambda$. Indeed, because $i_{h-1} < i_h$, $X_{i_{h-1}+1} X_{i_{h-1}+2} \cdots X_{i_h} \neq \lambda$.

Now, consider the \uparrow -expression $E_h = \langle \uparrow \varepsilon_{i_{h-1}+1} \cdots \varepsilon_{i_h} \rangle$. By definition,

$$\mathcal{S}(E_h) = X_{i_{h-1}+1} y'_{i_{h-1}+1} \cdots y'_{i_h-1} X_{i_h} = Z_h.$$

Hence, E_h is a \uparrow -expression denoting Z_h . Suppose that E_h is not operator-minimal, i.e., that there exists a \uparrow -expression $E'_h = \langle \uparrow \varepsilon_{h,1} \cdots \varepsilon_{h,n_h} \rangle$ for some $n_h \geq 1$ and \mathcal{N} -words or DNA expressions $\varepsilon_{h,1}, \dots, \varepsilon_{h,n_h}$, such that $\mathcal{S}(E'_h) = \mathcal{S}(E_h) = Z_h$ and $|E'_h| < |E_h|$. Then apparently,

$$|\varepsilon_{h,1} \cdots \varepsilon_{h,n_h}| < |\varepsilon_{i_{h-1}+1} \cdots \varepsilon_{i_h}| \quad (4.50)$$

and

$$\begin{aligned} E &= \langle \uparrow \varepsilon_1 \cdots \varepsilon_n \rangle \\ &\equiv \langle \uparrow \varepsilon_1 \cdots \varepsilon_{i_{h-1}} \langle \uparrow \varepsilon_{i_{h-1}+1} \cdots \varepsilon_{i_h} \rangle \varepsilon_{i_h+1} \cdots \varepsilon_n \rangle \\ &= \langle \uparrow \varepsilon_1 \cdots \varepsilon_{i_{h-1}} E_h \varepsilon_{i_h+1} \cdots \varepsilon_n \rangle \\ &\equiv \langle \uparrow \varepsilon_1 \cdots \varepsilon_{i_{h-1}} E'_h \varepsilon_{i_h+1} \cdots \varepsilon_n \rangle \\ &= \langle \uparrow \varepsilon_1 \cdots \varepsilon_{i_{h-1}} \langle \uparrow \varepsilon_{h,1} \cdots \varepsilon_{h,n_h} \rangle \varepsilon_{i_h+1} \cdots \varepsilon_n \rangle \\ &\equiv \langle \uparrow \varepsilon_1 \cdots \varepsilon_{i_{h-1}} \varepsilon_{h,1} \cdots \varepsilon_{h,n_h} \varepsilon_{i_h+1} \cdots \varepsilon_n \rangle. \end{aligned}$$

The second equivalence in this derivation is valid by Lemma 3.7, the other two by Lemma 3.6. Because of (4.50), the resulting \uparrow -expression $\langle \uparrow \varepsilon_1 \cdots \varepsilon_{i_{h-1}} \varepsilon_{h,1} \cdots \varepsilon_{h,n_h} \varepsilon_{i_h+1} \cdots \varepsilon_n \rangle$ is shorter than E . This, however, contradicts the fact that E is minimal. Consequently, E_h must be operator-minimal. \square

For each type of expressible formal DNA molecule X , we have described how to construct a minimal DNA expression denoting X and what the length of this DNA expression is. We have also demonstrated that there do not exist other minimal DNA expressions for X than the ones satisfying the description. We give a short overview of the results:

Corollary 4.80 *Let X be an expressible 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 $E = \langle \uparrow \alpha_1 \rangle$ (see Theorem 4.23).

The length of this minimal DNA expression is

$$|E| = 3 \cdot n_{\uparrow}(X) + |\nu(X)| = 3 + |X|.$$

2. If X is nick free, contains at least one single-stranded component and $T_{\uparrow}(X) = T_{\downarrow}(X)$, then the only minimal DNA expressions denoting X are \uparrow -expressions based on a maximal upper partitioning of X as described in Theorem 4.53(1), and \downarrow -expressions based on a maximal lower partitioning of X as described in Theorem 4.53(2) (see also Theorem 4.77).

The length of a minimal DNA expression E is

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

3. If X is nick free and $T_{\uparrow}(X) > T_{\downarrow}(X)$, then the only minimal DNA expressions denoting X are \uparrow -expressions based on a maximal upper partitioning of X , as described in Theorem 4.53(1) (see also Theorem 4.77).

The length of a minimal DNA expression E is

$$|E| = 3 + 3 \cdot T_{\uparrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|.$$

4. If X is nick free and $T_{\downarrow}(X) > T_{\uparrow}(X)$, then the only minimal DNA expressions denoting X are \downarrow -expressions based on a maximal lower partitioning of X , as described in Theorem 4.53(2) (see also Theorem 4.77).

The length of a minimal DNA expression E is

$$|E| = 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\downarrow}(X) + |\nu(X)|.$$

5. If X contains at least one lower nick letter, then the only minimal DNA expressions denoting X are \uparrow -expressions based on operator-minimal \uparrow -expressions for the formal DNA submolecules Z_1, Z_2, \dots, Z_m occurring in the nick free decomposition $Z_{1\Delta}Z_{2\Delta}\dots_{\Delta}Z_m$ of X , as described in Theorem 4.67 (see also Theorem 4.79).

The operator-minimal \uparrow -expressions denoting a (nick free) formal DNA submolecule Z_h are in turn based on a maximal upper partitioning of Z_h , as described in Theorem 4.65(1) (see also Theorem 4.78).

The length of a minimal DNA expression E denoting X is

$$|E| = 3 + 3 \cdot T_{\uparrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|.$$

6. If X contains at least one upper nick letter, then the only minimal DNA expressions denoting X are \downarrow -expressions based on operator-minimal \downarrow -expressions for the formal DNA submolecules Z_1, Z_2, \dots, Z_m occurring in the nick free decomposition $Z_{1\nabla}Z_{2\nabla}\dots_{\nabla}Z_m$ of X , analogous to the description in Theorem 4.67 (see also Theorem 4.79).

The operator-minimal \downarrow -expressions denoting a (nick free) formal DNA submolecule Z_h are in turn based on a maximal lower partitioning of Z_h , as described in Theorem 4.65(2) (see also Theorem 4.78).

The length of a minimal DNA expression E denoting X is

$$|E| = 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\downarrow}(X) + |\nu(X)|.$$

In each of the cases, a minimal DNA expression achieves the applicable lower bound on its length from Corollary 4.19.

Now, we can also say more about the numbers of operators occurring in a minimal or operator-minimal DNA expression.

A minimal \downarrow -expression contains only one occurrence of an operator, the operator \downarrow . We observe once more that minimal \uparrow -expressions and \downarrow -expressions achieve the lower bounds on their lengths from Corollary 4.19.

Next, consider an operator-minimal \uparrow -expression E denoting a formal DNA molecule X . If X is nick free, then by Theorem 4.65(1), each operator-minimal \uparrow -expression denoting X , and in particular E , achieves the lower bound from Corollary 4.19(1). If X contains (lower) nick letters, then E is minimal and thus also achieves the lower bound. Analogously, each operator-minimal \downarrow -expression achieves the lower bound from Corollary 4.19(2).

Because the lower bounds in Corollary 4.19(1) and (2) followed immediately from Theorem 4.18, we have

Corollary 4.81 *Let E be a DNA expression, and let $X = \mathcal{S}(E)$.*

1. *If E is a minimal \downarrow -expression, then*

$$\begin{aligned}\#_{\uparrow,\downarrow}(E) &= 0 \quad \text{and} \\ \#_{\downarrow}(E) &= 1.\end{aligned}$$

2. *If E is a minimal or operator-minimal \uparrow -expression, then*

$$\begin{aligned}\#_{\uparrow,\downarrow}(E) &= 1 + T_{\downarrow}(X) \quad \text{and} \\ \#_{\uparrow}(E) &= n_{\uparrow}(X).\end{aligned}$$

3. *If E is a minimal or operator-minimal \downarrow -expression, then*

$$\begin{aligned}\#_{\uparrow,\downarrow}(E) &= 1 + T_{\uparrow}(X) \quad \text{and} \\ \#_{\downarrow}(E) &= n_{\downarrow}(X).\end{aligned}$$

Because a minimal DNA expression is in particular operator-minimal, we would not have to mention minimal \uparrow -expressions and \downarrow -expressions explicitly in Claims 2 and 3. We do mention these minimal DNA expressions, because we want to make clear that the results also hold for them.

We conclude this subsection with two examples of types of formal DNA molecules for which the minimal DNA expressions are unique.

- Let X be a nick free formal DNA molecule which does not contain any lower component. By Lemma 4.10(2), $T_{\downarrow}(X) = 0$.

If X does not contain any upper component, then by Lemma 4.25, $X = \binom{\alpha_1}{c(\alpha_1)}$ for an \mathcal{N} -word α_1 . In that case, the only minimal DNA expression denoting X is $E = \langle \uparrow \alpha_1 \rangle$.

If, on the other hand, X contains at least one upper component, then by Lemma 4.10(1), $T_{\uparrow}(X) > 0$. A minimal DNA expression denoting X is based on a maximal upper partitioning \mathcal{M} of X . By Lemma 4.50, there is only one maximal upper partitioning of X : $\mathcal{M} = Y_0 = X$. In that case, Theorem 4.53(1) specifies exactly one minimal DNA expression.

- Let X be an expressible formal DNA molecule which does not contain any single-stranded component. Then by Corollary 2.6 and Theorem 3.4, $X = \binom{\alpha_1}{c(\alpha_1)} y_1 \binom{\alpha_2}{c(\alpha_2)} y_2 \dots y_{m-1} \binom{\alpha_m}{c(\alpha_m)}$ for some $m \geq 1$ and \mathcal{N} -words $\alpha_1, \alpha_2, \dots, \alpha_m$, where either $y_1 = y_2 = \dots = y_{m-1} = \triangle$, or $y_1 = y_2 = \dots = y_{m-1} = \nabla$. Without loss of generality, assume that $y_1 = y_2 = \dots = y_{m-1} = \triangle$.

If $m = 1$, then $X = \binom{\alpha_1}{c(\alpha_1)}$. Again the only minimal DNA expression denoting X is $E = \langle \uparrow \alpha_1 \rangle$.

If $m \geq 2$, then X contains at least one lower nick letter. A minimal DNA expression denoting X is based on operator-minimal \uparrow -expressions for $Z_1 = \binom{\alpha_1}{c(\alpha_1)}, \dots, Z_m = \binom{\alpha_m}{c(\alpha_m)}$. For $h = 1, \dots, m$, an operator minimal \uparrow -expression denoting $Z_h = \binom{\alpha_h}{c(\alpha_h)}$ is based on a maximal upper partitioning \mathcal{M}_h of Z_h . Because Z_h does not contain any lower component, the only maximal upper partitioning of Z_h is $\mathcal{M}_h = Z_h = \binom{\alpha_h}{c(\alpha_h)}$. Hence, the only operator-minimal \uparrow -expression denoting Z_h is $\langle \uparrow \langle \uparrow \alpha_h \rangle \rangle$. Now, Theorem 4.67 specifies one minimal DNA expression denoting X :

$$E = \langle \uparrow \langle \uparrow \alpha_1 \rangle \langle \uparrow \alpha_2 \rangle \dots \langle \uparrow \alpha_m \rangle \rangle.$$

Note that (regardless of the value of m) $T_{\downarrow}(X) = 0$ and $n_{\uparrow}(X) = m$. Indeed, the minimal \uparrow -expression E for the case that $m \geq 2$ has length

$$|E| = 3 + 3 \cdot m + |\alpha_1 \alpha_2 \dots \alpha_m| = 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|.$$

4.2.4 The number of minimal DNA expressions

In the preceding sections, we have seen examples of formal DNA molecules for which the minimal DNA expression is unique and examples of formal DNA molecules for which there exist more than one minimal DNA expression. Now, we will calculate the *number* of minimal DNA expression denoting an *arbitrary* expressible formal DNA molecule X .

We first introduce some notation.

Definition 4.82 *Let X be an expressible formal DNA molecule. Then*

- $n_{\min\uparrow}(X)$ is the number of minimal \uparrow -expressions denoting X ,
- $n_{\min\downarrow}(X)$ is the number of minimal \downarrow -expressions denoting X ,
- $n_{\min\updownarrow}(X)$ is the number of minimal \updownarrow -expressions denoting X ,

- $n_{\min}(X)$ is the number of minimal DNA expressions denoting X ,
- $n_{\text{opermin}\uparrow}(X)$ is the number of operator-minimal \uparrow -expressions denoting X ,
- $n_{\text{opermin}\downarrow}(X)$ is the number of operator-minimal \downarrow -expressions denoting X ,
- $n_{\text{opermin}\updownarrow}(X)$ is the number of operator-minimal \updownarrow -expressions denoting X .

Obviously, $n_{\min}(X) = n_{\min\uparrow}(X) + n_{\min\downarrow}(X) + n_{\min\updownarrow}(X)$. For each type of expressible formal DNA molecule, we can easily obtain values for (some of) the seven functions and relations between the functions:

Lemma 4.83 *Let X be an expressible formal DNA molecule.*

1. *If X is double-complete, then*

$$\begin{aligned} n_{\min\uparrow}(X) &= 0, \\ n_{\text{opermin}\uparrow}(X) &= 1, \\ n_{\min\downarrow}(X) &= 0, \\ n_{\text{opermin}\downarrow}(X) &= 1, \\ n_{\min\updownarrow}(X) = n_{\text{opermin}\updownarrow}(X) &= 1 \text{ and} \\ n_{\min}(X) &= 1. \end{aligned}$$

2. *If X is nick free, contains at least one single-stranded component and $T_{\uparrow}(X) = T_{\downarrow}(X)$, then*

$$\begin{aligned} n_{\min\uparrow}(X) = n_{\text{opermin}\uparrow}(X) &\geq 1, \\ n_{\min\downarrow}(X) = n_{\text{opermin}\downarrow}(X) &\geq 1, \\ n_{\min\updownarrow}(X) = n_{\text{opermin}\updownarrow}(X) &= 0 \text{ and} \\ n_{\min}(X) &= n_{\min\uparrow}(X) + n_{\min\downarrow}(X). \end{aligned}$$

3. *If X is nick free and $T_{\uparrow}(X) > T_{\downarrow}(X)$, then*

$$\begin{aligned} n_{\min\uparrow}(X) = n_{\text{opermin}\uparrow}(X) &\geq 1, \\ n_{\min\downarrow}(X) &= 0, \\ n_{\text{opermin}\downarrow}(X) &\geq 1, \\ n_{\min\updownarrow}(X) = n_{\text{opermin}\updownarrow}(X) &= 0 \text{ and} \\ n_{\min}(X) &= n_{\min\uparrow}(X). \end{aligned}$$

4. *If X is nick free and $T_{\downarrow}(X) > T_{\uparrow}(X)$, then*

$$\begin{aligned} n_{\min\uparrow}(X) &= 0, \\ n_{\text{opermin}\uparrow}(X) &\geq 1, \\ n_{\min\downarrow}(X) = n_{\text{opermin}\downarrow}(X) &\geq 1, \\ n_{\min\updownarrow}(X) = n_{\text{opermin}\updownarrow}(X) &= 0 \text{ and} \\ n_{\min}(X) &= n_{\min\downarrow}(X). \end{aligned}$$

5. If X contains at least one lower nick letter, then let $Z_1 \triangle Z_2 \triangle \dots \triangle Z_m$ for some $m \geq 2$ be the nick free decomposition of X .

$$\begin{aligned} n_{\min \uparrow}(X) = n_{\text{opermin} \uparrow}(X) &= n_{\text{opermin} \uparrow}(Z_1) \times \dots \times n_{\text{opermin} \uparrow}(Z_m) \geq 1, \\ n_{\min \downarrow}(X) = n_{\text{opermin} \downarrow}(X) &= 0, \\ n_{\min \uparrow}(X) &= 0 \text{ and} \\ n_{\min}(X) &= n_{\min \uparrow}(X). \end{aligned}$$

- (a) If X does not contain any single-stranded component, then

$$n_{\text{opermin} \uparrow}(X) = |Z_1| \times |Z_m|.$$

- (b) If X contains at least one single-stranded component, then

$$n_{\text{opermin} \uparrow}(X) = 0.$$

6. If X contains at least one upper nick letter, then let $Z_1 \nabla Z_2 \nabla \dots \nabla Z_m$ for some $m \geq 2$ be the nick free decomposition of X .

$$\begin{aligned} n_{\min \uparrow}(X) = n_{\text{opermin} \uparrow}(X) &= 0, \\ n_{\min \downarrow}(X) = n_{\text{opermin} \downarrow}(X) &= n_{\text{opermin} \downarrow}(Z_1) \times \dots \times n_{\text{opermin} \downarrow}(Z_m) \geq 1, \\ n_{\min \uparrow}(X) &= 0 \text{ and} \\ n_{\min}(X) &= n_{\min \downarrow}(X). \end{aligned}$$

- (a) If X does not contain any single-stranded component, then

$$n_{\text{opermin} \uparrow}(X) = |Z_1| \times |Z_m|.$$

- (b) If X contains at least one single-stranded component, then

$$n_{\text{opermin} \uparrow}(X) = 0.$$

Proof: We only prove the correctness of some of the equations.

It follows immediately from the semantics of a DNA expression that for certain formal DNA molecules X , there does not exist any \uparrow -expression, \downarrow -expression or \updownarrow -expression denoting X . Hence, the corresponding number of operator-minimal DNA expressions is also 0.

The correctness of the remaining equations follows directly from Corollary 4.80.

1. Assume that $X = \binom{\alpha_1}{c(\alpha_1)}$ for an \mathcal{N} -word α_1 .

By Theorem 4.65(1) and Theorem 4.78, we can construct an operator-minimal \uparrow -expression denoting X from an arbitrary maximal upper partitioning of X and there is no other way to construct an operator-minimal \uparrow -expression. X obviously does not contain any lower component, By Lemma 4.50, there exists exactly one maximal upper partitioning of X : $\mathcal{M} = Y_0 = X$. Hence, there is also exactly one operator-minimal \uparrow -expression denoting X : $E = \langle \uparrow \langle \downarrow \alpha_1 \rangle \rangle$, and $n_{\text{opermin} \uparrow}(X) = 1$.

The proof of the equality $n_{\text{opermin} \downarrow}(X) = 1$ is completely analogous.

3. Assume that X is nick free.

By Theorem 4.65(2), an operator-minimal \downarrow -expression denoting X can be constructed from an arbitrary minimal lower partitioning of X . By (the ‘lower analogue’ of) Lemma 4.48, the number of maximal lower partitionings of any nick free formal DNA molecule is positive. Hence, $n_{\text{opermin}\downarrow}(X) \geq 1$, regardless of the values of $T_{\uparrow}(X)$ and $T_{\downarrow}(X)$.

4. The proof of this claim is analogous to that of the previous claim.

5. Assume that X contains at least one lower nick letter, and let $Z_{1\Delta}Z_{2\Delta}\dots_{\Delta}Z_m$ for some $m \geq 2$ be the nick free decomposition of X .

By Theorem 4.67, Lemma 4.60(1) and Theorem 4.79, we can construct a minimal DNA expression denoting X from arbitrary operator-minimal \uparrow -expressions E_1, \dots, E_m denoting Z_1, \dots, Z_m , respectively, and there is no other way to obtain a minimal DNA expression.

For $h = 1, \dots, m$, there are $n_{\text{opermin}\uparrow}(Z_h) \geq 1$ operator-minimal \uparrow -expressions E_h denoting Z_h . Different choices for an E_h yield different minimal DNA expressions, because we simply copy the arguments of E_h into the minimal DNA expression. Finally, let $1 \leq h_1 \neq h_2 \leq m$. Then the choice of an operator-minimal \uparrow -expression for Z_{h_1} is independent of the choice for Z_{h_2} . Hence,

$$n_{\min}(X) = n_{\text{opermin}\uparrow}(Z_1) \times n_{\text{opermin}\uparrow}(Z_2) \times \dots \times n_{\text{opermin}\uparrow}(Z_m) \geq 1.$$

(a) Assume that X does not contain any single-stranded component. Then by Corollary 2.6, $X = \binom{\alpha_1}{c(\alpha_1)}_{\Delta} \binom{\alpha_2}{c(\alpha_2)}_{\Delta} \dots_{\Delta} \binom{\alpha_m}{c(\alpha_m)}$ for \mathcal{N} -words $\alpha_1, \alpha_2, \dots, \alpha_m$. Apparently, for $h = 1, \dots, m$, $Z_h = \binom{\alpha_h}{c(\alpha_h)}$.

Let $E = \langle \uparrow \varepsilon_1 \rangle$ be an arbitrary operator-minimal \downarrow -expression denoting X .

If ε_1 were an \mathcal{N} -word α_1 , then $X = \mathcal{S}(E) = \binom{\alpha_1}{c(\alpha_1)}$ would be nick free, which would contradict our prior assumption. Hence, ε_1 is a DNA expression E_1 : $E = \langle \uparrow E_1 \rangle$.

Let $X_1 = \mathcal{S}(E_1)$. By definition, $X = \kappa(X_1)$. Because the function κ does not introduce, nor remove nick letters, X_1 contains $m - 1$ lower nick letters, just like X . As E is operator-minimal, its argument E_1 must be minimal. Hence, by Lemma 4.60(1), E_1 is a \uparrow -expression, and by Theorem 4.67,

$$|E_1| = 3 + 3 \cdot T_{\downarrow}(X_1) + 3 \cdot n_{\uparrow}(X_1) + |\nu(X_1)|. \quad (4.51)$$

Let $Z_{1,1\Delta}Z_{1,2\Delta}\dots_{\Delta}Z_{1,m}$ be the nick free decomposition of X_1 . The relation $X = \kappa(X_1)$ implies that for $h = 1, \dots, m$, $Z_h = \binom{\alpha_h}{c(\alpha_h)} = \kappa(Z_{1,h})$.

By definition, each lower nick letter occurring in (the formal DNA molecule) X_1 is both preceded and succeeded by a double \mathcal{A} -letter. Hence, for $h = 1, \dots, m$, $Z_{1,h}$ contains at least one double component: $n_{\uparrow}(Z_{1,h}) \geq 1$. This implies

$$n_{\uparrow}(X_1) = n_{\uparrow}(Z_{1,1}) + n_{\uparrow}(Z_{1,2}) + \dots + n_{\uparrow}(Z_{1,m}) \geq m. \quad (4.52)$$

When we substitute this into (4.51), we obtain

$$|E_1| \geq 3 + 3 \cdot T_{\downarrow}(X_1) + 3 \cdot m + |\nu(X_1)| \geq 3 + 3 \cdot m + |\nu(X_1)|, \quad (4.53)$$

and thus

$$|E| = 3 + |E_1| \geq 6 + 3 \cdot m + |\nu(X_1)| = 6 + 3 \cdot m + |\nu(X)|. \quad (4.54)$$

Note that this lower bound on the length of E is higher than the lower bounds from Corollary 4.19(3) and (5). Apparently, the latter lower bounds are not tight for \downarrow -expressions denoting formal DNA molecules with (lower) nick letters.

Now consider the \downarrow -expression

$$E' = \langle \downarrow \langle \uparrow \langle \downarrow \alpha_1 \rangle \langle \downarrow \alpha_2 \rangle \dots \langle \downarrow \alpha_m \rangle \rangle \rangle.$$

It is easy to verify that $\mathcal{S}(E') = \binom{\alpha_1}{c(\alpha_1)}_{\Delta} \binom{\alpha_2}{c(\alpha_2)}_{\Delta} \dots \binom{\alpha_m}{c(\alpha_m)}_{\Delta} = X$, and that $|E'| = 6 + 3 \cdot m + |\nu(X)|$. As E' achieves the lower bound for \downarrow -expressions denoting X which is given in (4.54), E' is operator-minimal. Because also E is operator-minimal, it has the same length as E' . We thus have equality in (4.53) and (4.54):

$$|E_1| = 3 + 3 \cdot m + |\nu(X_1)|$$

and

$$|E| = 6 + 3 \cdot m + |\nu(X)|.$$

Consequently, $T_{\downarrow}(X_1) = 0$, $n_{\downarrow}(X_1) = m$ and in particular, for $h = 1, \dots, m$, $n_{\downarrow}(Z_{1,h}) = 1$. It is not hard to see that if for some h with $1 \leq h \leq m$, $Z_{1,h}$ contained an opening \downarrow -component, then this would also be an opening \downarrow -component of X_1 . This would, however, contradict $T_{\downarrow}(X_1) = 0$. Hence, for $h = 1, \dots, m$, $T_{\downarrow}(Z_{1,h}) = 0$.

By Theorem 4.79, there exist operator-minimal \uparrow -expressions $E_{1,1}, E_{1,2}, \dots, E_{1,m}$ denoting $Z_{1,1}, Z_{1,2}, \dots, Z_{1,m}$, respectively, such that $E_1 = \langle \uparrow \hat{E}_{1,1} \hat{E}_{1,2} \dots \hat{E}_{1,m} \rangle$, where for $h = 1, \dots, m$, $\hat{E}_{1,h}$ is the sequence of the arguments of $E_{1,h}$. By Theorem 4.65(1), for $h = 1, \dots, m$,

$$\begin{aligned} |E_{1,h}| &= 3 + 3 \cdot T_{\downarrow}(Z_{1,h}) + 3 \cdot n_{\downarrow}(Z_{1,h}) + |\nu(Z_{1,h})| \\ &= 3 + 3 \cdot 0 + 3 \cdot 1 + |\nu(Z_{1,h})| \\ &= 6 + |\alpha_h|. \end{aligned}$$

Consider any h with $1 \leq h \leq m$.

- If $2 \leq h \leq m-1$, then $Z_{1,h}$ both starts and ends with a double component. Because $n_{\downarrow}(Z_{1,h}) = 1$, these have to be the same double component: $Z_{1,h} = \binom{\alpha_h}{c(\alpha_h)}$, as $\kappa(Z_{1,h}) = \binom{\alpha_h}{c(\alpha_h)}$. By Claim 1, there is exactly one operator-minimal \uparrow -expression $E_{1,h}$ denoting $Z_{1,h}$ (and $E_{1,h} = \langle \uparrow \langle \downarrow \alpha_h \rangle \rangle$).

- If $h = 1$, then $Z_{1,h} = Z_{1,1}$ ends with a double component $\binom{\alpha''}{c(\alpha'')}$ for an \mathcal{N} -word α'' . Because $n_{\downarrow}(Z_{1,1}) = 1$, $Z_{1,1}$ does not contain any other double component. Hence, by Corollary 2.5, $\binom{\alpha''}{c(\alpha'')}$ is preceded in $Z_{1,1}$ by at most one other component, which then is a single-stranded component.

If $\binom{\alpha''}{c(\alpha'')}$ is not preceded in $Z_{1,1}$ by any other component, then $Z_{1,1} = \binom{\alpha''}{c(\alpha'')} = \binom{\alpha_1}{c(\alpha_1)}$.

If $\binom{\alpha''}{c(\alpha'')}$ is preceded in $Z_{1,1}$ by a single-stranded component, then this cannot be a lower component. Otherwise, it would be an *opening* lower component of $Z_{1,1}$, which would contradict $T_{\downarrow}(Z_{1,1}) = 0$. Consequently, $\binom{\alpha''}{c(\alpha'')}$ is preceded in $Z_{1,1}$ by an upper component $\binom{\alpha'}{-}$ for an \mathcal{N} -word α' : $Z_{1,1} = \binom{\alpha'}{-} \binom{\alpha''}{c(\alpha'')}$. Because $\kappa(Z_{1,1}) = \binom{\alpha'}{c(\alpha')}$ $\binom{\alpha''}{c(\alpha'')}$ $= \binom{\alpha'\alpha''}{c(\alpha'\alpha'')} = \binom{\alpha_1}{c(\alpha_1)}$, $\alpha'\alpha'' = \alpha_1$. Both α' and α'' must contain at least one \mathcal{N} -letter, which implies that there are $|\alpha_1| - 1$ ways to partition α_1 into α' and α'' .

We thus have $1 + (|\alpha_1| - 1) = |\alpha_1|$ possibilities for $Z_{1,1}$. By Lemma 4.50, both if $Z_{1,1} = \binom{\alpha_1}{c(\alpha_1)}$ and if $Z_{1,1} = \binom{\alpha'}{-} \binom{\alpha''}{c(\alpha'')}$ for a partitioning (α', α'') of α_1 , there is exactly one maximal upper partitioning of $Z_{1,1}$. Hence, by Theorem 4.65(1) and Theorem 4.78, there is exactly one operator-minimal \uparrow -expression $E_{1,1}$ denoting $Z_{1,1}$: either $E_{1,1} = \langle \uparrow \langle \downarrow \alpha_1 \rangle \rangle$ (if $Z_{1,1} = \binom{\alpha_1}{c(\alpha_1)}$), or $E_{1,1} = \langle \uparrow \alpha' \langle \downarrow \alpha'' \rangle \rangle$ (if $Z_{1,1} = \binom{\alpha'}{-} \binom{\alpha''}{c(\alpha'')}$ for a partitioning (α', α'') of α_1). We thus have $|\alpha_1|$ possibilities for $E_{1,1}$ and indeed, for each of the possibilities, $|E_{1,1}| = 6 + |\alpha_1|$.

- The case $h = m$ is dealt with analogously to the case $h = 1$. After having established that $Z_{1,m}$ begins with a double component, which may be succeeded by an upper component, we find that there are $|\alpha_m|$ possibilities for $Z_{1,m}$ and that each of these possibilities yields one DNA expression $E_{1,m}$.

We conclude that the operator-minimal \uparrow -expressions $E_{1,2}, \dots, E_{1,m-1}$ are fixed, that there are $|\alpha_1| = |Z_1|$ possibilities for $E_{1,1}$ (each one corresponding to a possibility for $Z_{1,1}$) and that there are $|\alpha_m| = |Z_m|$ possibilities for $E_{1,m}$ (each one corresponding to a possibility for $Z_{1,m}$). Obviously the choice for $E_{1,1}$ is independent of the choice for $E_{1,m}$, which implies that there are $|\alpha_1| \times |\alpha_m| = |Z_1| \times |Z_m|$ possibilities for E_1 altogether.

It is easily verified that indeed, for each of these possibilities, $E = \langle \downarrow E_1 \rangle$ denotes $X = \binom{\alpha_1}{c(\alpha_1)} \triangle \binom{\alpha_2}{c(\alpha_2)} \triangle \dots \triangle \binom{\alpha_m}{c(\alpha_m)}$, and that $|E| = 6 + 3 \cdot m + |\nu(X)|$.

6. The proof of this claim is analogous to that of the previous claim.

□

The only thing we do not know yet, are the numbers of operator-minimal \uparrow -expressions and \downarrow -expressions denoting a nick free formal DNA molecule which contains at least one single-stranded component. To achieve these numbers, we first concentrate on maximal upper partitionings and maximal lower partitionings, which, by Theorem 4.65 and Theorem 4.78, form the basis for the construction of these operator minimal DNA expressions.

We define functions $f_{T_{\downarrow}}$ and $f_{T_{\uparrow}}$ as follows.

Definition 4.84 *Let X be a nick free formal DNA molecule.*

For a maximal upper partitioning $\mathcal{M} = Y_0 X_1 Y_1 X_2 Y_2 \dots X_r Y_r$ with $r \geq 0$,

$$f_{T_{\downarrow}}(\mathcal{M}) = (T_{\downarrow}(X_1), T_{\downarrow}(X_2), \dots, T_{\downarrow}(X_r)).$$

For a maximal lower partitioning $\mathcal{M} = Y_0 X_1 Y_1 X_2 Y_2 \dots X_r Y_r$ with $r \geq 0$,

$$f_{T_{\uparrow}}(\mathcal{M}) = (T_{\uparrow}(X_1), T_{\uparrow}(X_2), \dots, T_{\uparrow}(X_r)).$$

N	Ordered partitions of N
0	()
1	(1)
2	(2), (1, 1)
3	(3), (1, 2), (2, 1), (1, 1, 1)
4	(4), (1, 3), (2, 2), (3, 1), (1, 1, 2), (1, 2, 1), (2, 1, 1), (1, 1, 1, 1)

Table 4.1: All ordered partitions for $N = 0, 1, 2, 3, 4$.

Hence, $f_{T_{\downarrow}}$ maps a maximal upper partitioning onto a sequence of numbers, and $f_{T_{\uparrow}}$ maps a maximal lower partitioning onto a sequence of numbers. Note that if $T_{\downarrow}(X) = 0$ for a nick free formal DNA molecule X , then by Lemma 4.10(2) and Lemma 4.50, the only maximal upper partitioning of X is $\mathcal{M} = X$ (with $r = 0$), for which $f_{T_{\downarrow}}(\mathcal{M}) = ()$. Analogously, if $T_{\uparrow}(X) = 0$, then the only maximal lower partitioning of X is $\mathcal{M} = X$, for which $f_{T_{\uparrow}}(\mathcal{M}) = ()$.

When we apply $f_{T_{\downarrow}}$ and $f_{T_{\uparrow}}$ to the three maximal upper partitionings and the maximal lower partitioning from Figure 4.7, we obtain

$$\begin{aligned} f_{T_{\downarrow}}(\mathcal{M}_{a1}) &= (1, 1, 1), \\ f_{T_{\downarrow}}(\mathcal{M}_{a2}) &= (2, 1), \\ f_{T_{\downarrow}}(\mathcal{M}_{a3}) &= (3) \text{ and} \\ f_{T_{\uparrow}}(\mathcal{M}_b) &= (1, 1, 2). \end{aligned}$$

Here, the subscript of \mathcal{M} refers to the subfigure of Figure 4.7.

We will examine the relation between maximal upper (or lower) partitionings on the one hand, and sequences of integers on the other hand, induced by the function $f_{T_{\downarrow}}$ ($f_{T_{\uparrow}}$, respectively).

Definition 4.85 *Let $N \geq 0$ be an integer. An ordered partition of N is a sequence of positive integers (t_1, \dots, t_r) for some $r \geq 0$ such that $t_1 + \dots + t_r = N$.*

As an illustration of this definition, Table 4.1 contains all ordered partitions for $N = 0, 1, 2, 3, 4$.

Lemma 4.86 *Let X be a nick free formal DNA molecule.*

1. *The function $f_{T_{\downarrow}}$ is a bijection from the maximal upper partitionings of X onto the ordered partitions of $T_{\downarrow}(X)$.*
2. *The function $f_{T_{\uparrow}}$ is a bijection from the maximal lower partitionings of X onto the ordered partitions of $T_{\uparrow}(X)$.*

As can be read from Table 4.1, if $T_{\downarrow}(X) = 0$, then the the only ordered partition of $T_{\downarrow}(X)$ is the empty sequence $()$ (with $r = 0$). Indeed, this is the image under $f_{T_{\downarrow}}$ of the only maximal upper partitioning in this case.

We want to emphasize that the function $f_{T_{\downarrow}}$ (or $f_{T_{\uparrow}}$) is a bijection from maximal upper (lower, respectively) partitionings onto ordered partitions, *only* when the formal DNA molecule X is given. For example, there are infinitely many maximal upper partitionings that are mapped by $f_{T_{\downarrow}}$ to the empty sequence $()$, namely the maximal

upper partitionings of all nick free formal DNA molecules X with $T_{\downarrow}(X) = 0$.

Proof:

1. Let $\mathcal{M} = Y_0X_1Y_1X_2Y_2 \dots X_rY_r$ for some $r \geq 0$ be a maximal upper partitioning of X . By Corollary 4.52(2), for $j = 1, \dots, r$, $T_{\downarrow}(X_j) = T_{\uparrow}(X_j) + 1 \geq 1$, and by Corollary 4.54(1), $T_{\downarrow}(X_1) + T_{\downarrow}(X_2) + \dots + T_{\downarrow}(X_r) = T_{\downarrow}(X)$. Indeed, $f_{T_{\downarrow}}(\mathcal{M}) = (T_{\downarrow}(X_1), T_{\downarrow}(X_2), \dots, T_{\downarrow}(X_r))$ is an ordered partition of $T_{\downarrow}(X)$. Hence, $f_{T_{\downarrow}}$ is a mapping from the maximal upper partitionings of X into the set of these ordered partitions.

We will now prove that $f_{T_{\downarrow}}$ is surjective and injective, and thus is a bijection. Let $\mathcal{M}' = Y'_0X'_1Y'_1X'_2Y'_2 \dots X'_{r_0}Y'_{r_0}$ for some $r_0 \geq 0$ be the complete maximal upper partitioning of X . By Lemma 4.51(2b), $X'_1, X'_2, \dots, X'_{r_0}$ are exactly all separating lower sequences of X , Hence, by Lemma 4.39, $r_0 = T_{\downarrow}(X)$.

Let (t_1, t_2, \dots, t_r) for some $r \geq 0$ be an arbitrary ordered partition of $T_{\downarrow}(X)$. Then consider the following maximal upper partitioning:

$$\mathcal{M} = Y'_0X_1Y'_{t_1}X_2Y'_{t_1+t_2} \dots X_{t_1+t_2+\dots+t_r}Y'_{t_1+t_2+\dots+t_r}. \quad (4.55)$$

Hence, for $j = 1, \dots, r$, the substring X_j is equal to the subsequence

$$X'_{t_1+\dots+t_{j-1}+1}Y'_{t_1+\dots+t_{j-1}+1} \dots Y'_{t_1+\dots+t_{j-1}}X'_{t_1+\dots+t_j}$$

of separating lower sequences and maximal upper sequences of X . Since all t_j 's are positive and $t_1+t_2+\dots+t_r = T_{\downarrow}(X) = r_0$, the sequences $Y'_{t_1}, Y'_{t_1+t_2}, \dots, Y'_{t_1+t_2+\dots+t_{r-1}}$ occurring in (4.55) are different internal maximal upper sequences and $Y'_{t_1+t_2+\dots+t_r}$ is the maximal upper suffix Y'_{r_0} of X . This implies that, indeed, \mathcal{M} is a maximal upper partitioning of X .

By Corollary 4.52(1), for $j = 1, \dots, r$,

$$T_{\downarrow}(X_j) = (t_1 + \dots + t_j) - (t_1 + \dots + t_{j-1} + 1) + 1 = t_j.$$

Consequently,

$$f_{T_{\downarrow}}(\mathcal{M}) = (T_{\downarrow}(X_1), T_{\downarrow}(X_2), \dots, T_{\downarrow}(X_r)) = (t_1, t_2, \dots, t_r).$$

We have thus created a maximal upper partitioning of X which is mapped by $f_{T_{\downarrow}}$ onto the ordered partition (t_1, t_2, \dots, t_r) . Because this was an arbitrary ordered partition of $T_{\downarrow}(X)$, $f_{T_{\downarrow}}$ is surjective.

Let

$$\begin{aligned} \mathcal{M}_1 &= Y_{1,0}X_{1,1}Y_{1,1}X_{1,2}Y_{1,2} \dots X_{1,r_1}Y_{1,r_1}, \text{ and} \\ \mathcal{M}_2 &= Y_{2,0}X_{2,1}Y_{2,1}X_{2,2}Y_{2,2} \dots X_{2,r_2}Y_{2,r_2} \end{aligned}$$

for some $r_1, r_2 \geq 0$ be two maximal upper partitionings, for which $f_{T_{\downarrow}}(\mathcal{M}_1) = f_{T_{\downarrow}}(\mathcal{M}_2) = (t_1, \dots, t_r)$ for some $r \geq 0$. By the definition of $f_{T_{\downarrow}}$, $r_1 = r_2 = r$.

We now prove that for $j = 0, \dots, r$, $Y_{1,j} = Y_{2,j}$. We do this by induction on j .

- If $j = 0$, then by the definition of a maximal upper partitioning, $Y_{1,0} = Y_{2,0}$ is the maximal upper prefix Y'_0 of X .
- Let $0 \leq j \leq r - 1$, and suppose that $Y_{1,j} = Y_{2,j}$ (induction hypothesis). We now consider $Y_{1,j+1}$ and $Y_{2,j+1}$. Because \mathcal{M}_1 and \mathcal{M}_2 are maximal upper partitionings of the same formal DNA molecule X , we have

$$Y_{1,0}X_{1,1}Y_{1,1} \dots X_{1,j}Y_{1,j} = Y_{2,0}X_{2,1}Y_{2,1} \dots X_{2,j}Y_{2,j} \quad (4.56)$$

$Y_{1,j}$ is succeeded in \mathcal{M}_1 by $X_{1,j+1}$ and $Y_{2,j}$ is succeeded in \mathcal{M}_2 by $X_{2,j+1}$. By Lemma 4.51(3a), both $X_{1,j+1}$ and $X_{2,j+1}$ are alternating sequences of separating lower sequences and maximal upper sequences of X , starting and ending with a separating lower sequence. Because of (4.56), they start with the same separating lower sequence. By assumption, $T_\downarrow(X_{1,j+1}) = T_\downarrow(X_{2,j+1}) = t_{j+1}$, and hence, by Corollary 4.52(1), $X_{1,j+1}$ and $X_{2,j+1}$ contain the same number of separating lower sequences of X . This implies that they also end with the same separating lower sequence, and thus that they are equal.

Now, if $j + 1 < r$, then both $Y_{1,j+1}$ and $Y_{2,j+1}$ are the internal maximal upper sequence succeeding $X_{1,j+1} = X_{2,j+1}$ in X . If $j + 1 = r$, then by the definition of a maximal upper partitioning, $Y_{1,j+1} = Y_{2,j+1}$ is the maximal upper suffix Y'_{r_0} of X . In both cases, $Y_{1,j+1} = Y_{2,j+1}$.

Because a maximal upper partitioning is defined by the (internal) maximal upper sequences Y_j occurring in it, \mathcal{M}_1 and \mathcal{M}_2 must be equal. We conclude that f_{T_\downarrow} is injective.

2. The proof of this claim is analogous to that of the previous claim.

□

By the above result, the number of different maximal upper partitionings of a certain nick free formal DNA molecule X must be equal to the number of ordered partitions of $T_\downarrow(X)$. By Lemma 4.48, the former number is $2^{n_{\text{imus}}(X)}$, which, by Lemma 4.10(2) and Lemma 4.46(2), is equal to

$$\begin{cases} 2^0 = 1 & \text{if } T_\downarrow(X) = 0, \\ 2^{T_\downarrow(X)-1} & \text{if } T_\downarrow(X) > 0. \end{cases}$$

There is also an elegant, direct way to compute the number of different ordered partitions of $T_\downarrow(X)$. It is a standard result in the theory of partitions.

Lemma 4.87 *Let $N \geq 0$ be an integer. Then the number of different ordered partitions of N is*

$$\begin{cases} 1 & \text{if } N = 0, \\ 2^{N-1} & \text{if } N > 0. \end{cases}$$

Proof: As we have observed before, if $N = 0$, then the only ordered partition of N is $()$.

For $N > 0$, we will systematically count the different sequences (t_1, \dots, t_r) of positive integers satisfying $t_1 + \dots + t_r = N$.

First we observe that r (the number of integers t_j) must be at least 1 and at most N . With $r = 0$, the sum $t_1 + t_2 + \dots + t_r$ would be 0 by definition, which is smaller than N . With $r > N$, the sum $t_1 + t_2 + \dots + t_r$ would be larger than N , because each t_j is at least 1.

Given an integer r with $1 \leq r \leq N$, we must count all possibilities to distribute the number N over the variables t_1, t_2, \dots, t_r , such that each t_j is at least 1. This is equivalent to counting the ways to put N indistinguishable balls into r distinguishable urns, such that each urn holds at least one ball. Because of this last condition, we may start by putting one ball into every urn. After this, we may freely distribute the remaining $N - r$ balls over the r urns. This second step is equivalent to lining up the $N - r$ balls from left to right, and then inserting $r - 1$ walls (separators) into this line to indicate how many balls will be put into the first urn, how many will be put into the second urn, etc.

For example, the ordered partition $(1, 4, 2, 5)$ of $N = 12$ (with $r = 4$ and $N - r = 8$) corresponds to the sequence $|000|0|0000$, where $|$ represents a wall and 0 represents a ball. Note that this particular sequence starts with a wall, which means that the first urn receives none of the $N - r = 8$ balls that we may freely distribute. Indeed, according to the ordered partition, the first urn is supposed to hold exactly 1 ball, i.e., only the obligatory ball.

We return to the general case. Inserting $r - 1$ walls into a line of $N - r$ balls is equivalent to selecting $r - 1$ positions for the walls from a total of $(N - r) + (r - 1) = N - 1$ positions. The number of ways to do this is $\binom{N - 1}{r - 1}$.

Now the total number of different ordered partitions of N is

$$\sum_{r=1}^N \binom{N - 1}{r - 1} = \sum_{r=0}^{N-1} \binom{N - 1}{r} = 2^{N-1}.$$

The last equation can be understood as follows. $\sum_{r=0}^{N-1} \binom{N - 1}{r}$ counts the number of subsets from a $(N - 1)$ -item set: first the subset containing $r=0$ items, then the subsets containing $r=1$ item, etc. However, to determine the number of subsets from a $(N - 1)$ -item set, it is sufficient to observe that for each item we have two possibilities: it is in the subset or it is not. Then in total we have $\underbrace{2 \cdot 2 \cdot \dots \cdot 2}_{N - 1 \text{ times}} = 2^{N-1}$ possibilities. \square

Lemma 4.86 relates maximal upper partitionings and maximal lower partitionings of a nick free formal DNA molecule to ordered partitions. We will now establish a relation between, on the one hand, operator minimal \uparrow -expressions and \downarrow -expressions denoting such formal DNA molecules and, on the other hand, certain sequences of brackets.

Definition 4.88 Let $p \geq 0$. A sequence of p well-nested pairs of brackets is a string $Z = x_1 \dots x_{2p}$ such that

- for $i = 1, \dots, 2p$, $x_i \in \{\langle, \rangle\}$, and
- $\#_{\langle}(Z) = \#_{\rangle}(Z)$, and
- for $i = 0, \dots, 2p$, $\#_{\langle}(x_1 \dots x_i) \geq \#_{\rangle}(x_1 \dots x_i)$.

The number p , i.e., the number of pairs of brackets in a sequence Z of well-nested pairs of brackets is denoted by $n_{pb}(Z)$.

The third condition of the definition intuitively says that, when we read Z from left to right, the number of closing brackets \rangle we have read so far never exceeds the number of opening brackets \langle we have read so far.

Note that the inequality in this third condition is automatically valid for $i = 0$ and for $i = 2p$. In the former case, $x_1 \dots x_i = \lambda$, and $\#_{\langle}(\lambda) = \#_{\rangle}(\lambda) = 0$. In the latter case, $x_1 \dots x_i = Z$, and $\#_{\langle}(Z) = \#_{\rangle}(Z)$ by the second condition.

Note further that if Z is a sequence of well-nested pairs of brackets, then so is $\langle Z \rangle$. The reverse statement is not true. For example, $\langle \rangle \langle \rangle$ is a sequence of well-nested pairs of brackets (with $p = 2$), but $\rangle \langle$ is not, because it violates the third condition of the definition.

Other examples of sequences of well-nested pairs of brackets are

$$\begin{aligned} \lambda & \quad (\text{the empty string, with } p = 0), \\ \langle \langle \rangle \rangle \langle \rangle & \quad (\text{with } p = 3) \text{ and} \\ \langle \rangle \langle \rangle \langle \langle \rangle \rangle & \quad (\text{with } p = 4). \end{aligned}$$

The term ‘sequence of well-nested pairs of brackets’ is explained by the following result:

Lemma 4.89 *A string Z is a sequence of well-nested pairs of brackets, if and only if there exist strings Z_1, Z_2, \dots, Z_r for some $r \geq 0$, such that*

1. $Z = \langle Z_1 \rangle \langle Z_2 \rangle \dots \langle Z_r \rangle$,
2. for $j = 1, \dots, r$, Z_j is itself a sequence of well-nested pairs of brackets.

In this case, the partitioning of Z as $\langle Z_1 \rangle \langle Z_2 \rangle \dots \langle Z_r \rangle$ is unique.

Proof: \implies Let $Z = x_1 \dots x_{2p}$ for some $p \geq 0$ be a sequence of well-nested pairs of brackets. Then let $i_0 = 0$, and let $1 \leq i_1 < i_2 < \dots < i_r \leq 2p$ for some $r \geq 0$ be all indices i such that $\#_{\langle}(x_1 \dots x_i) = \#_{\rangle}(x_1 \dots x_i)$.

Consider any j with $1 \leq j \leq r$. By definition, $\#_{\langle}(x_1 \dots x_{i_{j-1}}) = \#_{\rangle}(x_1 \dots x_{i_{j-1}})$ (also if $j = 1$, i.e., if $i_{j-1} = 0$), $1 \leq i_{j-1} + 1 \leq 2p$ and for $i = 1, \dots, 2p$, $\#_{\langle}(x_1 \dots x_i) \geq \#_{\rangle}(x_1 \dots x_i)$. Consequently, $x_{i_{j-1}+1}$ must be an opening bracket \langle . Analogously, x_{i_j} is a closing bracket \rangle . Because $i_{j-1} < i_j$ and $x_{i_{j-1}+1} \neq x_{i_j}$, we even have $i_{j-1} + 1 < i_j$.

Now define Z_j as the substring of Z between the opening bracket $x_{i_{j-1}+1}$ and the closing bracket x_{i_j} :

$$Z_j = x_{i_{j-1}+2} \dots x_{i_j-1}.$$

We will prove that Z_1, \dots, Z_r satisfy conditions (1) and (2).

If $p = 0$, then by definition $r = 0$. Indeed $Z = \lambda$ in this case, and conditions (1) and (2) are satisfied trivially.

Now assume that $p \geq 1$. By definition, $\#_{\langle}(Z) = \#_{\rangle}(Z)$, and thus $r \geq 1$ and $i_r = 2p$. Also by definition, $i_{r-1} + 1 = i_0 + 1 = 1$. Finally, for $j = 1, \dots, r$,

$$\langle Z_j \rangle = x_{i_{j-1}+1} x_{i_{j-1}+2} \dots x_{i_j-1} x_{i_j}.$$

This all implies

$$\langle Z_1 \rangle \langle Z_2 \rangle \dots \langle Z_r \rangle = x_{i_0+1} \dots x_{i_1} x_{i_1+1} \dots x_{i_2} \dots x_{i_{r-1}+1} \dots x_{i_r} = x_1 \dots x_{2p} = Z,$$

which gives condition (1).

Consider any j with $1 \leq j \leq r$. Obviously, Z_j consists only of opening brackets \langle and closing brackets \rangle . Their respective numbers of occurrences satisfy

$$\begin{aligned} \#_{\langle}(Z_j) &= \#_{\langle}(x_{i_{j-1}+2} \dots x_{i_j-1}) \\ &= \#_{\langle}(x_1 \dots x_{i_j}) - \#_{\langle}(x_1 \dots x_{i_{j-1}}) - \#_{\langle}(x_{i_{j-1}+1}) - \#_{\langle}(x_{i_j}) \\ &= \#_{\rangle}(x_1 \dots x_{i_j}) - \#_{\rangle}(x_1 \dots x_{i_{j-1}}) - 1 - 0 \\ &= \#_{\rangle}(x_{i_{j-1}+2} \dots x_{i_j-1}) \\ &= \#_{\rangle}(Z_j). \end{aligned}$$

This implies in particular that the length $|Z_j|$ must be even, i.e., $2p_j$ for some $p_j \geq 0$.

Finally, let $Z'_j = x_{i_{j-1}+2} \dots x_i$ with $i_{j-1} + 1 \leq i \leq i_j - 1$ be an arbitrary prefix of Z_j (which may be equal to λ , but may also be equal to Z_j). Because i is not equal to any of i_1, \dots, i_r , we have

$$\#_{\langle}(x_1 \dots x_i) \geq \#_{\rangle}(x_1 \dots x_i) + 1.$$

This implies that

$$\begin{aligned} \#_{\langle}(Z'_j) &= \#_{\langle}(x_{i_{j-1}+2} \dots x_i) \\ &= \#_{\langle}(x_1 \dots x_i) - \#_{\langle}(x_1 \dots x_{i_{j-1}}) - \#_{\langle}(x_{i_{j-1}+1}) \\ &\geq \#_{\rangle}(x_1 \dots x_i) + 1 - \#_{\rangle}(x_1 \dots x_{i_{j-1}}) - 1 \\ &= \#_{\rangle}(x_{i_{j-1}+2} \dots x_i) \\ &= \#_{\rangle}(Z'_j). \end{aligned}$$

Indeed, Z_j satisfies the definition of a sequence of well-nested pairs of brackets. Hence, also condition (2) is satisfied.

\Leftarrow Let Z be a string and let Z_1, \dots, Z_r for some $r \geq 0$ be sequences of well-nested pairs of brackets, such that $Z = \langle Z_1 \rangle \dots \langle Z_r \rangle$.

Then obviously, every letter of Z is an opening bracket \langle or a closing bracket \rangle . It can easily be shown by induction that for $j = 0, \dots, r$,

$$\#_{\langle}(\langle Z_1 \rangle \dots \langle Z_j \rangle) = \#_{\rangle}(\langle Z_1 \rangle \dots \langle Z_j \rangle). \quad (4.57)$$

In particular, when we take $j = r$, we have

$$\#_{\langle}(Z) = \#_{\langle}(\langle Z_1 \rangle \dots \langle Z_r \rangle) = \#_{\rangle}(\langle Z_1 \rangle \dots \langle Z_r \rangle) = \#_{\rangle}(Z).$$

This implies that the length of Z is even, i.e., $2p$ for some $p \geq 0$.

Consider any prefix Z' of Z that is different from the ones occurring in (4.57). Then Z' is of the form

$$Z' = \langle Z_1 \rangle \dots \langle Z_{j-1} \rangle \langle Z'_j$$

for some j with $1 \leq j \leq r$ and some prefix Z'_j of Z_j (which may be equal to λ , but may also be equal to Z_j). Because Z_j is a sequence of well-nested pairs of brackets,

$$\#_{\langle}(Z'_j) \geq \#_{\rangle}(Z'_j).$$

Hence,

$$\begin{aligned} \#_{\langle}(Z') &= \#_{\langle}(\langle Z_1 \rangle \dots \langle Z_{j-1} \rangle) + 1 + \#_{\langle}(Z'_j) \\ &\geq \#_{\rangle}(\langle Z_1 \rangle \dots \langle Z_{j-1} \rangle) + 1 + \#_{\rangle}(Z'_j) \\ &> \#_{\rangle}(Z') \end{aligned}$$

We have thus proved that Z is a sequence of well-nested pairs of brackets.

Moreover, $\langle Z_1 \rangle, \langle Z_1 \rangle \langle Z_2 \rangle, \dots, \langle Z_1 \rangle \dots \langle Z_r \rangle$ are the only non-empty prefixes Z' of Z for which $\#_{\langle}(Z') = \#_{\rangle}(Z')$. Consequently, when we apply the construction from the first part of the proof (the proof of \implies) to Z , we precisely obtain the partitioning $\langle Z_1 \rangle \dots \langle Z_r \rangle$.

Uniqueness Let Z be a sequence of well-nested pairs of brackets, let $r_1, r_2 \geq 0$ and let $Z_{1,1}, \dots, Z_{1,r_1}$ and $Z_{2,1}, \dots, Z_{2,r_2}$ be sequences of well-nested pairs of brackets, such that $Z = \langle Z_{1,1} \rangle \dots \langle Z_{1,r_1} \rangle$ and $Z = \langle Z_{2,1} \rangle \dots \langle Z_{2,r_2} \rangle$. By the concluding remark in the second part of the proof (the proof of \impliedby), we can obtain both partitionings by applying the construction from the first part of the proof to Z . Because this construction is unambiguous, the two partitionings must be equal. \square

Sequences of well-nested pairs of brackets are well-known in combinatorics. The number of such sequences is one of the many combinatorial interpretations of the Catalan numbers $C_p = \frac{1}{p+1} \binom{2p}{p}$ for $p \geq 0$:

Lemma 4.90 *The number of different sequences of $p \geq 0$ well-nested pairs of brackets is $\frac{1}{p+1} \binom{2p}{p}$.*

Proof: This result is included as Exercise 6.19(r) in [Stanley, 1999], where 1's are substituted for opening brackets and -1 's are substituted for closing brackets. Here, we give a well-known proof, which is simple and elegant.

Let $p \geq 0$. A sequence of p well-nested pairs of brackets is, in particular, a sequence of p opening brackets and p closing brackets. The number of different sequences of p opening brackets and p closing brackets is equal to the number of ways to select p positions for the opening brackets from a total of $2p$ positions, which is $\binom{2p}{p}$.

This number, however, also includes 'wrong' sequences, sequences that satisfy the first two conditions of Definition 4.88, but do not satisfy the third condition. Consider such a 'wrong' sequence $Z = x_1 \dots x_{2p}$. There exists at least one index i with $0 \leq i \leq 2p$ for which $\#_{\langle}(x_1 \dots x_i) < \#_{\rangle}(x_1 \dots x_i)$. Let i_0 be the smallest such index i . By definition, for $i = 0$ and $i = 2p$, $\#_{\langle}(x_1 \dots x_i) = \#_{\rangle}(x_1 \dots x_i)$. Hence, $1 \leq i_0 \leq 2p - 1$. Because of the minimality of i_0 , $\#_{\langle}(x_1 \dots x_{i_0-1}) = \#_{\rangle}(x_1 \dots x_{i_0-1})$, x_{i_0} is a closing bracket \rangle and

$$\#_{\langle}(x_1 \dots x_{i_0}) = \#_{\rangle}(x_1 \dots x_{i_0}) - 1. \quad (4.58)$$

Consequently,

$$\#_{\langle}(x_{i_0+1} \dots x_{2p}) = \#_{\rangle}(x_{i_0+1} \dots x_{2p}) + 1. \quad (4.59)$$

Let Z' be the sequence of opening brackets and closing brackets that results from Z by replacing each opening bracket x_i with $i > i_0$ by a closing bracket and replacing each closing bracket x_i with $i > i_0$ by an opening bracket: $Z' = x'_1 \dots x'_{2p}$, where

$$x'_i = \begin{cases} x_i & \text{if } i \leq i_0 \\ \rangle & \text{if } i > i_0 \text{ and } x_i = \langle \\ \langle & \text{if } i > i_0 \text{ and } x_i = \rangle \end{cases} \quad (i = 1, \dots, 2p).$$

It follows from (4.58) and (4.59) that

$$\#_{\langle}(Z') = \#_{\rangle}(Z') - 2,$$

p	C_p	p	C_p	p	C_p	p	C_p
0	1	5	42	10	16,796	15	9,694,845
1	1	6	132	11	58,786	16	35,357,670
2	2	7	429	12	208,012	17	129,644,790
3	5	8	1,430	13	742,900	18	477,638,700
4	14	9	4,862	14	2,674,440	19	1,767,263,190

Table 4.2: The Catalan numbers C_p for $p = 0, 1, \dots, 19$.

i.e., that Z' contains $p - 1$ opening brackets and $p + 1$ closing brackets.

It is easy to prove that each sequence Z' of $p - 1$ opening brackets and $p + 1$ closing brackets can be obtained in this way from exactly one ‘wrong’ sequence Z . Hence, the number of ‘wrong’ sequences of p opening brackets and p closing brackets equals the number of sequences of $p - 1$ opening brackets and $p + 1$ closing brackets, which is $\binom{2p}{p-1}$. This implies that the number of ‘correct’ sequences, sequences of p *well-nested* pairs of brackets is

$$\binom{2p}{p} - \binom{2p}{p-1} = \frac{(2p)!}{p!p!} - \frac{(2p)!}{(p-1)!(p+1)!} = \left(1 - \frac{p}{p+1}\right) \binom{2p}{p} = \frac{1}{p+1} \binom{2p}{p}.$$

□

Table 4.2 lists the Catalan numbers C_p for $p = 0, 1, \dots, 19$. As the table suggests, the sequence of the Catalan numbers exhibits an exponential growth. In fact, we have for $p \geq 1$,

$$C_p = \frac{1}{p+1} \binom{2p}{p} = \frac{2p \cdot (2p-1)}{(p+1)p} \cdot \frac{1}{p} \binom{2(p-1)}{p-1} = \left(4 - \frac{6}{p+1}\right) C_{p-1}.$$

Hence, the sequence grows almost as fast as the sequence 4^p for $p \geq 0$.

When we examined the lengths of \uparrow -expressions E denoting a formal DNA molecule X , the value $T_{\downarrow}(X)$ played an important role. For example, it occurs in the lower bound for $|E|$ from Corollary 4.19(1), a lower bound that is achieved by minimal and operator-minimal \uparrow -expressions. Likewise, the value $T_{\uparrow}(X)$ appeared to be important in our analysis of the lengths of \downarrow -expressions denoting X .

We will see that to calculate the *number* of operator-minimal \uparrow -expressions (or \downarrow -expressions) denoting a nick free formal DNA molecule X , we again need the value $T_{\downarrow}(X)$ (or $T_{\uparrow}(X)$, respectively). To have direct access to these values, we define a function T_{\neg} :

Definition 4.91 *Let E be a \uparrow -expression or a \downarrow -expression denoting a certain formal DNA molecule X (which may contain nick letters).*

$$T_{\neg}(E) = \begin{cases} T_{\downarrow}(X) & \text{if } E \text{ is a } \uparrow\text{-expression,} \\ T_{\uparrow}(X) & \text{if } E \text{ is a } \downarrow\text{-expression.} \end{cases}$$

The subscript \neg is used in the notation T_{\neg} , because, intuitively, the function counts components in $X = \mathcal{S}(E)$ that do *not* correspond to the outermost operator of E .

We also define mappings from operator-minimal \uparrow -expressions and \downarrow -expressions onto sequences of brackets.

Definition 4.92 Let E be an operator-minimal \uparrow -expression or an operator-minimal \downarrow -expression, denoting a certain nick free formal DNA molecule X .

If E is a \uparrow expression, then let E_1, \dots, E_r for some $r \geq 0$ be the \downarrow -arguments of E , in the order of their occurrence in E .

If E is a \downarrow expression, then let E_1, \dots, E_r for some $r \geq 0$ be the \uparrow -arguments of E , in the order of their occurrence in E .

Then

$$\begin{aligned} f_1(E) &= \langle f_1(E_1) \dots f_1(E_r) \rangle, \\ f_2(E) &= f_1(E_1) \dots f_1(E_r). \end{aligned}$$

The definition of the function f_1 is recursive: $f_1(E)$ is defined in terms of $f_1(E_j)$ for specific arguments E_j of E . We will see soon that this recursion is well defined. Further, $f_2(E)$ can simply be obtained from $f_1(E)$ by removing both its first symbol and its last symbol.

Equation (4.20) contains a minimal (and thus operator-minimal) \uparrow -expression E for the formal DNA molecule depicted in Figure 4.8, corresponding to the maximal upper partitioning in Figure 4.7(a2). For this DNA expression E , we have $r = 2$ and

$$\begin{aligned} f_1(E) &= \langle \langle \langle \rangle \rangle \langle \rangle \rangle, \\ f_2(E) &= \langle \langle \rangle \rangle \langle \rangle. \end{aligned}$$

We can use Theorem 4.65(2) to construct the following operator-minimal \downarrow -expression E' denoting the same formal DNA molecule. It corresponds to the maximal lower partitioning in Figure 4.7(b):

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

For this DNA expression E' , $r = 3$ and

$$\begin{aligned} f_1(E') &= \langle \langle \rangle \langle \rangle \langle \langle \rangle \rangle \rangle, \\ f_2(E') &= \langle \rangle \langle \rangle \langle \langle \rangle \rangle. \end{aligned}$$

We now prove some properties of the functions f_1 and f_2 .

Lemma 4.93

1. The functions f_1 and f_2 are well defined.
2. For each operator-minimal \uparrow -expression or \downarrow -expression E denoting a nick free formal DNA molecule,
 - (a) $f_1(E)$ results from E by removing from E all letters except the brackets \langle and \rangle corresponding to operators \uparrow and \downarrow .
 - (b) $f_2(E)$ is a sequence of $T_-(E)$ well-nested pairs of brackets.

Because $f_1(E) = \langle f_2(E) \rangle$, $f_1(E)$ is a sequence of $1 + T_-(E)$ well-nested pairs of brackets. Indeed, by Corollary 4.81(2) and (3), each operator-minimal \uparrow -expression or \downarrow -expression, and in particular one denoting a nick free formal DNA molecule, contains $1 + T_-(E)$ occurrences of the operators \uparrow and \downarrow together.

Proof: We simultaneously prove all claims, by induction on $T_-(E)$, where E is an arbitrary operator-minimal \uparrow -expression or \downarrow -expression denoting a certain nick free formal DNA molecule X .

- If E is a \uparrow -expression and $T_{\downarrow}(E) = T_{\downarrow}(X) = 0$, then let E_1, \dots, E_r for some $r \geq 0$ be the \downarrow -arguments of E , and let for $j = 1, \dots, r$, $X_j = \mathcal{S}(E_j)$. By Lemma 4.74(1), for $j = 1, \dots, r$, $T_{\downarrow}(X_j) = T_{\uparrow}(X_j) + 1 \geq 1$. Further, by Lemma 4.74(2), $T_{\downarrow}(X_1) + \dots + T_{\downarrow}(X_r) = T_{\downarrow}(X) = 0$. Then, however, r must be 0, and by definition, $f_1(E) = \langle \rangle$, which is obviously well defined.

By Corollary 4.70, each argument of (the \uparrow -expression) E is either an \mathcal{N} -word α or a \uparrow -expression $\langle \uparrow \alpha \rangle$ for an \mathcal{N} -word α . Indeed, when we remove from E all letters except the brackets corresponding to operators \uparrow and \downarrow , we remove the outermost operator \uparrow and all arguments of E , leaving $\langle \rangle = f_1(E)$. In this case, $f_2(E) = \lambda$. Indeed, $f_2(E)$ is a sequence of $T_{\downarrow}(E) = 0$ well-nested pairs of brackets.

The proof for the case that E is a \downarrow -expression and $T_{\downarrow}(E) = T_{\uparrow}(X) = 0$ is analogous.

- Let $p \geq 0$, and suppose that for each operator-minimal \uparrow -expression or \downarrow -expression E denoting a nick free formal DNA molecule, for which $T_{\downarrow}(E) \leq p$, the claims are valid (induction hypothesis).

Now consider an operator-minimal \uparrow -expression E denoting a certain nick free formal DNA molecule X , for which $T_{\downarrow}(E) = T_{\downarrow}(X) = p + 1$. Let E_1, \dots, E_r for some $r \geq 0$ ¹ be the \downarrow -arguments of E , in the order of their occurrence in E .

Consider E_j for an arbitrary j with $1 \leq j \leq r$. We first observe that E_j is an operator-minimal (even minimal) \downarrow -expression, because E is operator-minimal. Hence, the functions f_1 and f_2 apply to E_j . Let $X_j = \mathcal{S}(E_j)$. By Lemma 4.74(1) and (2),

$$T_{\downarrow}(E_j) = T_{\uparrow}(X_j) = T_{\downarrow}(X_j) - 1 \leq T_{\downarrow}(X) - 1 = p.$$

Hence, by the induction hypothesis, $f_1(E_j)$ and $f_2(E_j)$ are well defined, $f_1(E_j)$ results from E_j by removing from E_j all letters except the brackets corresponding to operators \uparrow and \downarrow , and $f_2(E_j)$ is a sequence of $T_{\downarrow}(E_j) = T_{\uparrow}(X_j)$ well-nested pairs of brackets.

Because j was arbitrary, also $f_1(E) = \langle f_1(E_1) \dots f_1(E_r) \rangle$ and $f_2(E) = f_1(E_1) \dots f_1(E_r)$ are well defined. This is Claim 1 for E .

Obviously, a bracket \langle or \rangle occurring in an argument E_j of E corresponds to an operator \uparrow or \downarrow in E if and only if it does so in E_j . Further, by Corollary 4.70, each argument of E that is not a \downarrow -expression, is either an \mathcal{N} -word α or a \uparrow -expression $\langle \uparrow \alpha \rangle$ for an \mathcal{N} -word α . Such an argument does not contain brackets corresponding to an operator \uparrow or \downarrow . Hence, when we remove from (the \uparrow -expression) E all letters except the brackets corresponding to operators \uparrow and \downarrow , we remove the outermost operator \uparrow of E , the arguments of E which are not a \downarrow -expression, and all letters in the \downarrow -expressions E_1, \dots, E_r which are not a bracket \langle or \rangle corresponding to an operator \uparrow or \downarrow . This leaves $\langle f_1(E_1) \dots f_1(E_r) \rangle = f_1(E)$. Consequently, Claim 2a is also valid for E .

By Lemma 4.89,

$$f_2(E) = \langle f_2(E_1) \rangle \dots \langle f_2(E_r) \rangle$$

¹In fact, since $T_{\downarrow}(X) \geq 1$, it follows from Lemma 4.74(2) that $r \geq 1$. This is, however, not important for the proof.

is again a sequence of well-nested pairs of brackets. The only thing left to be done is to calculate the number of pairs of brackets in $f_2(E)$:

$$\begin{aligned}
 n_{\text{pb}}(f_2(E)) &= (1 + n_{\text{pb}}(f_2(E_1))) + \cdots + (1 + n_{\text{pb}}(f_2(E_r))) \\
 &= (1 + T_{\uparrow}(X_1)) + \cdots + (1 + T_{\uparrow}(X_r)) \\
 &= T_{\uparrow}(X_1) + \cdots + T_{\uparrow}(X_r) \\
 &= T_{\uparrow}(X).
 \end{aligned}$$

Here, we again used Lemma 4.74(1) and (2) to establish the last two equations. Thus, we have also proved Claim 2b for E .

The proof for the case that E is an operator-minimal \downarrow -expression denoting a certain nick free formal DNA molecule X , for which $T_{\downarrow}(E) = T_{\uparrow}(X) = p + 1$, is analogous.

□

We now concentrate on the function f_2 . We extend Lemma 4.93(2b) to the following, important result:

Theorem 4.94 *Let X be a nick free formal DNA molecule.*

1. *The function f_2 is a bijection from the operator-minimal \uparrow -expressions denoting X onto the sequences of $T_{\downarrow}(X)$ well-nested pairs of brackets.*
2. *The function f_2 is a bijection from the operator-minimal \downarrow -expressions denoting X onto the sequences of $T_{\uparrow}(X)$ well-nested pairs of brackets.*

Proof: By Lemma 4.93(2b), the function f_2 is indeed a mapping from the operator-minimal \uparrow -expressions (or \downarrow -expressions) denoting X into the set of sequences of $T_{\downarrow}(X)$ ($T_{\uparrow}(X)$, respectively) well-nested pairs of brackets.

Now, we will prove that f_2 is surjective and injective, both for the operator-minimal \uparrow -expressions and for the operator-minimal \downarrow -expressions. Hence, we will prove that

- 1' for each sequence Z of $p = T_{\downarrow}(X)$ well-nested pairs of brackets, there is exactly one operator-minimal \uparrow -expression E denoting X such that $f_2(E) = Z$, and
- 2' for each sequence Z of $p = T_{\uparrow}(X)$ well-nested pairs of brackets, there is exactly one operator-minimal \downarrow -expression E denoting X such that $f_2(E) = Z$.

We do this by induction on p .

- If $T_{\downarrow}(X) = 0$, then the only sequence of $p = T_{\downarrow}(X)$ well-nested pairs of brackets is $Z = \lambda$.

By Lemma 4.10(2) and Lemma 4.50, X does not contain any lower component and there is exactly one maximal upper partitioning of X , namely $\mathcal{M} = Y_0 = X$. Then by Theorem 4.65(1) and Theorem 4.78, there is exactly one operator-minimal \uparrow -expression E denoting X , each of whose arguments is an \mathcal{N} -word α_i or a \uparrow -expression $\langle \uparrow \alpha_i \rangle$ for an \mathcal{N} -word α_i . For this \uparrow -expression E , indeed $f_2(E) = Z = \lambda$ and we have proved Claim 1' for $p = 0$.

The proof of Claim 2' for $p = 0$ is analogous.

- Let $p \geq 0$, and suppose that Claim 1' is valid for each nick free formal DNA molecule X with $T_{\downarrow}(X) \leq p$, and that Claim 2' is valid for each nick free formal DNA molecule X with $T_{\uparrow}(X) \leq p$ (induction hypothesis).

Now consider a nick free formal DNA molecule X with $T_{\downarrow}(X) = p + 1$. Let Z be an arbitrary sequence of $p + 1$ well-nested pairs of brackets.

By Lemma 4.89, there is a (unique) partitioning of Z as $\langle Z_1 \rangle \dots \langle Z_r \rangle$ for some $r \geq 0$ ² such that for $j = 1, \dots, r$, Z_j is a sequence of well-nested pairs of brackets.

For $j = 1, \dots, r$, let $t_j = n_{\text{pb}}(\langle Z_j \rangle)$. Obviously, for $j = 1, \dots, r$, $t_j \geq 1$. Also, $t_1 + \dots + t_r = n_{\text{pb}}(Z) = p + 1 = T_{\downarrow}(X)$. Hence, (t_1, \dots, t_r) is an ordered partition of $T_{\downarrow}(X)$.

By Lemma 4.86(1), there is exactly one maximal upper partitioning $\mathcal{M} = Y_0 X_1 Y_1 \cdot X_2 Y_2 \dots X_{r'} Y_{r'}$ of X such that

$$f_{T_{\downarrow}}(\mathcal{M}) = (T_{\downarrow}(X_1), \dots, T_{\downarrow}(X_{r'})) = (t_1, \dots, t_r).$$

Obviously, $r' = r$ here.

Consider an arbitrary j with $1 \leq j \leq r$. We have $n_{\text{pb}}(Z_j) = t_j - 1 \leq T_{\downarrow}(X) - 1 = p$. By Corollary 4.52(2) $T_{\uparrow}(X_j) = T_{\downarrow}(X_j) - 1 = t_j - 1$. Then by the induction hypothesis there is exactly one operator-minimal \downarrow -expression E_j denoting X_j such that $f_2(E_j) = Z_j$. By Corollary 4.59(1), E_j is even minimal.

We use E_1, \dots, E_r to construct an operator-minimal \uparrow -expression E denoting X based on the maximal upper partitioning \mathcal{M} , as described in Theorem 4.65(1). According to the construction, the only \downarrow -arguments of E are E_1, \dots, E_r . Then by definition,

$$f_2(E) = f_1(E_1) \dots f_1(E_r) = \langle f_2(E_1) \rangle \dots \langle f_2(E_r) \rangle = Z.$$

We have thus created an operator-minimal \uparrow -expression E denoting X for which $f_2(E) = Z$. Because Z was an arbitrary sequence of $p + 1$ well-nested pairs of brackets, the function f_2 is surjective for operator-minimal \uparrow -expressions E denoting X .

In order to complete the proof of Claim 1' for $T_{\downarrow}(X) = p + 1$, we must show that f_2 is also injective for such \uparrow -expressions. For this, let E_1 and E_2 be two operator-minimal \uparrow -expressions denoting X for which $f_2(E_1) = f_2(E_2) = Z$ for a certain sequence Z of $p + 1 = T_{\downarrow}(X)$ well-nested pairs of brackets.

We first analyse what it means that $f_2(E_1) = f_2(E_2) = Z$. Let $E_{1,1}, \dots, E_{1,r_1}$ for some $r_1 \geq 0$ be the \downarrow -arguments of E_1 , in the order of their occurrence in E_1 , and let $E_{2,1}, \dots, E_{2,r_2}$ for some $r_2 \geq 0$ be the \downarrow -arguments of E_2 , in the order of their occurrence in E_2 . By definition,

$$Z = f_2(E_1) = f_1(E_{1,1}) \dots f_1(E_{1,r_1}) = \langle f_2(E_{1,1}) \rangle \dots \langle f_2(E_{1,r_1}) \rangle \quad (4.60)$$

$$= f_2(E_2) = f_1(E_{2,1}) \dots f_1(E_{2,r_2}) = \langle f_2(E_{2,1}) \rangle \dots \langle f_2(E_{2,r_2}) \rangle. \quad (4.61)$$

By Lemma 4.89, there is a unique partitioning of Z as $\langle Z_1 \rangle \dots \langle Z_r \rangle$ for some $r \geq 0$, such that for $j = 1, \dots, r$, Z_j is a sequence of well-nested pairs of brackets.

²In fact, since $p + 1 \geq 1$ and thus $Z \neq \lambda$, we have $r \geq 1$. This is, however, not important for the proof.

Because both (4.60) and (4.61) provide such a partitioning, we must have $r_1 = r_2 = r$ and for $j = 1, \dots, r$, $f_2(E_{1,j}) = f_2(E_{2,j}) = Z_j$. Then also, for $j = 1, \dots, r$, $f_1(E_{1,j}) = \langle f_2(E_{1,j}) \rangle = \langle f_2(E_{2,j}) \rangle = f_1(E_{2,j})$.

We proceed with the implications of E_1 and E_2 being operator-minimal \uparrow -expressions denoting X . By Theorem 4.78, E_1 and E_2 are based on maximal upper partitionings of X , as described in Theorem 4.65(1). For $i = 1, 2$, let \mathcal{M}_i be the maximal upper partitioning corresponding to E_i . We concentrate on \mathcal{M}_1 first. Let $\mathcal{M}_1 = Y_0 X_1 Y_1 X_2 Y_2 \dots X_{r'} Y_{r'}$ for some $r' \geq 0$. By the construction from Theorem 4.65(1) and Corollary 4.59(1), the \downarrow -arguments $E_{1,1}, \dots, E_{1,r}$ of E_1 are minimal DNA expressions denoting $X_1, X_2, \dots, X_{r'}$, respectively. This implies in particular that $r' = r$.

By Corollary 4.52(2) and Lemma 4.93(2b), for $j = 1, \dots, r$,

$$T_{\downarrow}(X_j) = T_{\uparrow}(X_j) + 1 = n_{\text{pb}}(f_2(E_{1,j})) + 1 = n_{\text{pb}}(f_1(E_{1,j})).$$

Hence,

$$f_{T_{\downarrow}}(\mathcal{M}_1) = (T_{\downarrow}(X_1), \dots, T_{\downarrow}(X_r)) = (n_{\text{pb}}(f_1(E_{1,1})), \dots, n_{\text{pb}}(f_1(E_{1,r}))). \quad (4.62)$$

Completely analogously, we obtain

$$f_{T_{\downarrow}}(\mathcal{M}_2) = (n_{\text{pb}}(f_1(E_{2,1})), \dots, n_{\text{pb}}(f_1(E_{2,r}))). \quad (4.63)$$

As we observed before, for $j = 1, \dots, r$, $f_1(E_{1,j}) = f_1(E_{2,j})$. This implies in particular that the right-hand sides of (4.62) and (4.63) are equal. By Lemma 4.86(1), the function $f_{T_{\downarrow}}$ is a bijection from the maximal upper partitionings of X onto the ordered partitions of $T_{\downarrow}(X)$. Consequently, \mathcal{M}_1 and \mathcal{M}_2 are equal, say $\mathcal{M}_1 = \mathcal{M}_2 = \mathcal{M}$.

Consider an arbitrary j with $1 \leq j \leq r$. We have observed that $E_{1,j}$ is a minimal \downarrow -expression denoting X_j , and so is $E_{2,j}$. In particular, $E_{1,j}$ and $E_{2,j}$ are operator-minimal and $f_2(E_{1,j}) = f_2(E_{2,j}) = Z_j$. By Corollary 4.54(1), $T_{\uparrow}(X_j) = T_{\downarrow}(X_j) - 1 \leq T_{\downarrow}(X) - 1 = p$. This implies that we can apply the induction hypothesis to X_j , i.e., there exists exactly one operator-minimal \downarrow -expression E_j denoting X_j such that $f_2(E_j) = Z_j$. In other words, $E_{1,j} = E_{2,j} = E_j$.

We have thus proved that E_1 and E_2 are based on the same maximal upper partitioning $\mathcal{M} = Y_0 X_1 Y_1 X_2 Y_2 \dots X_r Y_r$ and that, in the construction from Theorem 4.65(1), they use the same minimal \downarrow -expressions E_1, \dots, E_r denoting X_1, \dots, X_r , respectively. Because, in this construction, the arguments ε_i corresponding to the substrings Y_0, Y_1, \dots, Y_r (\mathcal{N} -words α_i and \uparrow -expressions $\langle \downarrow \alpha_i \rangle$ for \mathcal{N} -words α_i) are fixed, E_1 and E_2 must be equal. We conclude that f_2 is also injective, and thus bijective, for operator-minimal \uparrow -expressions denoting X . Hence, Claim 1' is valid for formal DNA molecules X with $T_{\downarrow}(X) = p + 1$.

The proof of Claim 2' for formal DNA molecules X with $T_{\uparrow}(X) = p + 1$ is analogous.

□

When we combine Theorem 4.94 with Lemma 4.90, we obtain

Corollary 4.95 *Let X be a nick free formal DNA molecule.*

1. *The number of operator-minimal \uparrow -expressions denoting X is $\frac{1}{p+1} \binom{2p}{p}$, with $p = T_{\downarrow}(X)$.*
2. *The number of operator-minimal \downarrow -expressions denoting X is $\frac{1}{p+1} \binom{2p}{p}$, with $p = T_{\uparrow}(X)$.*

In Lemma 4.83, we could not specify values for *all* numbers of (operator-)minimal DNA expressions denoting certain types of expressible formal DNA molecules. We now can:

Corollary 4.96 *Let X be an expressible formal DNA molecule.*

1. *If X is double-complete, then*

$$\begin{aligned}
 n_{\min\uparrow}(X) &= 0, \\
 n_{\operatorname{opermin}\uparrow}(X) &= 1, \\
 n_{\min\downarrow}(X) &= 0, \\
 n_{\operatorname{opermin}\downarrow}(X) &= 1, \\
 n_{\min\downarrow}(X) = n_{\operatorname{opermin}\downarrow}(X) &= 1 \text{ and} \\
 n_{\min}(X) &= 1.
 \end{aligned}$$

2. *If X is nick free, contains at least one single-stranded component and $T_{\uparrow}(X) = T_{\downarrow}(X) = p$ for some $p \geq 1$, then*

$$\begin{aligned}
 n_{\min\uparrow}(X) = n_{\operatorname{opermin}\uparrow}(X) &= \frac{1}{p+1} \binom{2p}{p}, \\
 n_{\min\downarrow}(X) = n_{\operatorname{opermin}\downarrow}(X) &= \frac{1}{p+1} \binom{2p}{p}, \\
 n_{\min\downarrow}(X) = n_{\operatorname{opermin}\downarrow}(X) &= 0 \quad \text{and} \\
 n_{\min}(X) &= \frac{2}{p+1} \binom{2p}{p}.
 \end{aligned}$$

3. *If X is nick free, $T_{\uparrow}(X) = p_1$ and $T_{\downarrow}(X) = p_2$ for some p_1 and p_2 with $p_1 > p_2 \geq 0$, then*

$$\begin{aligned}
 n_{\min\uparrow}(X) = n_{\operatorname{opermin}\uparrow}(X) &= \frac{1}{p_2+1} \binom{2p_2}{p_2}, \\
 n_{\min\downarrow}(X) &= 0, \\
 n_{\operatorname{opermin}\downarrow}(X) &= \frac{1}{p_1+1} \binom{2p_1}{p_1}, \\
 n_{\min\downarrow}(X) = n_{\operatorname{opermin}\downarrow}(X) &= 0 \quad \text{and} \\
 n_{\min}(X) &= \frac{1}{p_2+1} \binom{2p_2}{p_2}.
 \end{aligned}$$

4. If X is nick free, $T_{\uparrow}(X) = p_1$ and $T_{\downarrow}(X) = p_2$ for some p_1 and p_2 with $p_2 > p_1 \geq 0$, then

$$\begin{aligned} n_{\min\uparrow}(X) &= 0, \\ n_{\text{opermin}\uparrow}(X) &= \frac{1}{p_2 + 1} \binom{2p_2}{p_2}, \\ n_{\min\downarrow}(X) = n_{\text{opermin}\downarrow}(X) &= \frac{1}{p_1 + 1} \binom{2p_1}{p_1}, \\ n_{\min\uparrow}(X) = n_{\text{opermin}\uparrow}(X) &= 0 \quad \text{and} \\ n_{\min}(X) &= \frac{1}{p_1 + 1} \binom{2p_1}{p_1}. \end{aligned}$$

5. If X contains at least one lower nick letter, then let $Z_{1\Delta}Z_{2\Delta}\dots_{\Delta}Z_m$ for some $m \geq 2$ be the nick free decomposition of X , and let for $h = 1, \dots, m$, $p_h = T_{\downarrow}(Z_h)$.

$$\begin{aligned} n_{\min\uparrow}(X) = n_{\text{opermin}\uparrow}(X) &= \frac{1}{p_1 + 1} \binom{2p_1}{p_1} \times \dots \times \frac{1}{p_m + 1} \binom{2p_m}{p_m} \\ n_{\min\downarrow}(X) = n_{\text{opermin}\downarrow}(X) &= 0, \\ n_{\min\uparrow}(X) &= 0 \quad \text{and} \\ n_{\min}(X) &= \frac{1}{p_1 + 1} \binom{2p_1}{p_1} \times \dots \times \frac{1}{p_m + 1} \binom{2p_m}{p_m}. \end{aligned}$$

- (a) If X does not contain any single-stranded component, then

$$n_{\text{opermin}\uparrow}(X) = |Z_1| \times |Z_m|.$$

- (b) If X contains at least one single-stranded component, then

$$n_{\text{opermin}\uparrow}(X) = 0.$$

6. If X contains at least one upper nick letter, then let $Z_1^{\nabla}Z_2^{\nabla}\dots^{\nabla}Z_m$ for some $m \geq 2$ be the nick free decomposition of X , and let for $h = 1, \dots, m$, $p_h = T_{\uparrow}(Z_h)$.

$$\begin{aligned} n_{\min\uparrow}(X) = n_{\text{opermin}\uparrow}(X) &= 0, \\ n_{\min\downarrow}(X) = n_{\text{opermin}\downarrow}(X) &= \frac{1}{p_1 + 1} \binom{2p_1}{p_1} \times \dots \times \frac{1}{p_m + 1} \binom{2p_m}{p_m}, \\ n_{\min\uparrow}(X) &= 0 \quad \text{and} \\ n_{\min}(X) &= \frac{1}{p_1 + 1} \binom{2p_1}{p_1} \times \dots \times \frac{1}{p_m + 1} \binom{2p_m}{p_m}. \end{aligned}$$

- (a) If X does not contain any single-stranded component, then

$$n_{\text{opermin}\uparrow}(X) = |Z_1| \times |Z_m|.$$

- (b) If X contains at least one single-stranded component, then

$$n_{\text{opermin}\uparrow}(X) = 0.$$

4.2.5 Recurrence relation for the number of operator-minimal \uparrow -expressions and \downarrow -expressions

In the previous subsection we determined the number of operator-minimal \uparrow -expressions (or \downarrow -expressions) denoting a certain nick free formal DNA molecule by establishing a bijection between such DNA expressions and sequences of well-nested pairs of brackets. We could have chosen another way to achieve the same result, and we will briefly describe this alternative now. We leave it to the reader to prove the correctness of the alternative.

By Theorem 4.65(1) and Theorem 4.78, an operator-minimal \uparrow -expression denoting a nick free formal DNA molecule X is based on a maximal upper partitioning of X . By the construction from Theorem 4.65(1), different maximal upper partitionings yield different operator-minimal \uparrow -expressions. Let $\mathcal{M} = Y_0X_1Y_1X_2Y_2 \dots X_rY_r$ for some $r \geq 0$ be a maximal upper partitioning of X . A corresponding operator-minimal \uparrow -expression E has arguments ε_i for all components x'_i of a Y_j , and arguments E_1, \dots, E_r that are minimal DNA expressions denoting X_1, \dots, X_r , respectively. By Corollary 4.59(1), E_1, \dots, E_r are \downarrow -expressions. The arguments ε_i are fixed, but an argument E_j may be any (operator-)minimal \downarrow -expression denoting X_j . Because the choice of E_{j_1} is independent of the choice of E_{j_2} if $1 \leq j_1 \neq j_2 \leq r$, the number of different operator-minimal \uparrow -expressions corresponding to \mathcal{M} is

$$n_{\text{opermin}\downarrow}(X_1) \times \dots \times n_{\text{opermin}\downarrow}(X_r).$$

Hence, the total number of operator-minimal \uparrow -expressions denoting X is

$$\sum_{\substack{\text{max. upper part.} \\ Y_0X_1Y_1 \dots X_rY_r \text{ of } X}} n_{\text{opermin}\downarrow}(X_1) \times \dots \times n_{\text{opermin}\downarrow}(X_r).$$

Analogously, an operator-minimal \downarrow -expression denoting a nick free formal DNA molecule X' is based on a maximal lower partitioning of X' , different maximal lower partitionings of X' yield different operator-minimal \downarrow -expressions, and the total number of operator-minimal \downarrow -expressions denoting X' is

$$\sum_{\substack{\text{max. lower part.} \\ Y'_0X'_1Y'_1 \dots X'_rY'_r \text{ of } X'}} n_{\text{opermin}\uparrow}(X'_1) \times \dots \times n_{\text{opermin}\uparrow}(X'_r).$$

As we have seen in Lemma 4.86, each maximal upper partitioning $\mathcal{M} = Y_0X_1Y_1X_2Y_2 \dots X_rY_r$ of X can be identified with the ordered partition $(t_1, \dots, t_r) = f_{T_\downarrow}(\mathcal{M}) = (T_\downarrow(X_1), \dots, T_\downarrow(X_r))$ of $T_\downarrow(X)$, and each maximal lower partitioning \mathcal{M}' of X' can be identified with the ordered partition $f_{T_\uparrow}(\mathcal{M}')$ of $T_\uparrow(X')$.

Now, we can prove by induction on $T_\downarrow(X)$ and $T_\uparrow(X')$ that the number of operator-minimal \uparrow -expressions denoting X only depends on $T_\downarrow(X)$, that the number of operator-minimal \downarrow -expressions denoting X' only depends on $T_\uparrow(X')$, that these numbers are equal if $T_\downarrow(X) = T_\uparrow(X') = p$, say $C(p)$, and that

$$C(p) = \begin{cases} 1 & \text{if } p = 0, \\ \sum_{\substack{\text{ordered partitions} \\ (t_1, \dots, t_r) \text{ of } p}} C(t_1 - 1) \times \dots \times C(t_r - 1) & \text{if } p \geq 1. \end{cases} \quad (4.64)$$

³This summation would also give the right value 1 for $p = 0$. For the sake of clearness, however, we mention the case $p = 0$ separately.

When we recall that, by Lemma 4.89, each sequence of p well-nested pairs of brackets, has a unique partitioning as $\langle Z_1 \rangle \langle Z_2 \rangle \dots \langle Z_r \rangle$ for some $r \geq 0$ and sequences of well-nested pairs of brackets Z_1, \dots, Z_r , we find that recurrence relation (4.64) is also applicable to the number of sequences of p well-nested pairs of brackets. Hence, indeed for every $p \geq 0$, $C(p)$ is equal to this number, which is the Catalan number $\frac{1}{p+1} \binom{2p}{p}$.

Recurrence relation (4.64) is also related to trees. Consider an arbitrary ordered, directed tree with $p + 1$ nodes for some $p \geq 0$. This tree consists of a root and $r \geq 0$ ordered subtrees. Each of the subtrees contains at least one node and together they contain p nodes. Hence, the respective numbers of nodes in the subtrees form an ordered partition (t_1, \dots, t_r) of p . Now it is not hard to see that the *number* of different ordered, directed trees with $p + 1$ nodes satisfies recurrence relation (4.64). Indeed, the number of these trees is another well-known combinatorial interpretation of the Catalan numbers $\frac{1}{p+1} \binom{2p}{p}$ (see exercise 6.19(e) in [Stanley, 1999]).

4.2.6 Recognition of minimal DNA expressions

In the previous subsections, we have learned how to construct a minimal DNA expression denoting an arbitrary (expressible) formal DNA molecule. We have also observed that we can calculate the length of a minimal DNA expression directly from the formal DNA molecule. We do not have to explicitly construct a minimal DNA expression for this.

Now suppose that we are given a DNA expression E and that we want to decide whether or not it is minimal. Then we can use the following three-stage approach: we first determine the semantics $\mathcal{S}(E)$ of E , we subsequently calculate the length of a minimal DNA expression denoting $\mathcal{S}(E)$, and we finally count the length of E and check if it is equal to this minimal length.

There is, however, also a direct method to check if a DNA expression is minimal. This method is based on a characterization of minimal DNA expressions by six conditions on (the arguments of) the operators occurring in them. Before we establish a relation with minimality, we formally define the set of DNA expressions that satisfy these conditions.

Definition 4.97 *The set \mathcal{D}_{Min} is the set of DNA expressions $E \in \mathcal{D}$ such that*

- ($\mathcal{D}_{Min}.1$) *each occurrence of the operator \Downarrow in E has as its argument an \mathcal{N} -word α (and not a DNA expression), and*
- ($\mathcal{D}_{Min}.2$) *no occurrence of the operator \Uparrow in E has a \Uparrow -argument, and no occurrence of the operator \Downarrow in E has a \Downarrow -argument, and*
- ($\mathcal{D}_{Min}.3$) *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*
- ($\mathcal{D}_{Min}.4$) *for each inner occurrence of an operator \Uparrow or \Downarrow in E , the arguments are maximal \mathcal{N} -word occurrences and DNA expressions, alternately, and*
- ($\mathcal{D}_{Min}.5$) *for each inner occurrence of an operator \Uparrow or \Downarrow in E ,*
 - *the first argument is either an \mathcal{N} -word α or a \Downarrow -expression $\langle \Downarrow \alpha \rangle$ for an \mathcal{N} -word α ,*

- and the last argument is either an \mathcal{N} -word α or a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α ,

and

($\mathcal{D}_{\text{Min.6}}$) 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 α or a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α ,
- or its last argument is an \mathcal{N} -word α or a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α .

Membership of \mathcal{D}_{Min} carries over to DNA subexpressions:

Lemma 4.98 *A DNA expression E is in \mathcal{D}_{Min} if and only if each DNA subexpression of E is in \mathcal{D}_{Min} .*

Proof: Let E be an arbitrary DNA expression.

Clearly, if each DNA subexpression of E is in \mathcal{D}_{Min} , then so is E , because E is a DNA subexpression of itself.

Now assume that $E \in \mathcal{D}_{\text{Min}}$ and let E^s be a proper DNA subexpression of E . We will prove that E^s satisfies all conditions in Definition 4.97, and thus is in \mathcal{D}_{Min} .

Each occurrence of an operator in E^s is also an occurrence of that operator in E , with the same arguments. In particular, each inner occurrence of an operator \uparrow or \downarrow in E^s is also an inner occurrence of that operator in E , with the same arguments. Hence, the Conditions ($\mathcal{D}_{\text{Min.1}}$), ($\mathcal{D}_{\text{Min.2}}$), ($\mathcal{D}_{\text{Min.4}}$) and ($\mathcal{D}_{\text{Min.5}}$) are valid for E^s , simply because they are valid for E .

Because we assume that E^s is a *proper* DNA subexpression of E , E cannot be equal to $\langle \uparrow \alpha \rangle$ or $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α . Such DNA expressions do not have proper DNA subexpressions. Hence, by Condition ($\mathcal{D}_{\text{Min.3}}$) for E , each occurrence of an operator \uparrow or \downarrow in E has at least two arguments. Of course, the same holds for such an occurrence in E^s , so Condition ($\mathcal{D}_{\text{Min.3}}$) is also valid for E^s . Note that indeed, E^s cannot be equal to $\langle \uparrow \alpha \rangle$ or $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α .

If the outermost operator of E^s is \uparrow or \downarrow , then it is an inner occurrence of that operator in E . Hence, by Condition ($\mathcal{D}_{\text{Min.5}}$), both its first argument is an \mathcal{N} -word α or a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α , and its last argument is an \mathcal{N} -word α or a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α . This implies that also Condition ($\mathcal{D}_{\text{Min.6}}$) is valid for E^s . Note that by Condition ($\mathcal{D}_{\text{Min.4}}$) for E , the outermost operator of E^s cannot have two consecutive arguments which are DNA expressions. \square

We use the conditions in Definition 4.97 to derive other properties of elements of \mathcal{D}_{Min} .

Lemma 4.99 *Let E be a DNA expression in \mathcal{D}_{Min} .*

1. (a) For each proper \uparrow -subexpression of E , the parent operator is \downarrow .
(b) For each proper \downarrow -subexpression of E , the parent operator is \uparrow .
2. Each proper \uparrow -subexpression or \downarrow -subexpression of E has at least two arguments.

3. Each proper DNA subexpression of E has at least one \mathcal{N} -word-argument α .
4. (a) Each proper \uparrow -subexpression or \downarrow -subexpression of E which is not the first argument of its parent operator has as its (own) first argument a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α .
 (b) Each proper \uparrow -subexpression or \downarrow -subexpression of E which is not the last argument of its parent operator has as its (own) last argument a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α .
5. For each proper \uparrow -subexpression or \downarrow -subexpression of E , either the first argument, or the last argument is a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α .
6. Each proper \uparrow -subexpression or \downarrow -subexpression of E which is neither the first argument, nor the last argument of E , has an odd number of arguments (at least three), the first one and the last one of which are \downarrow -expressions $\langle \downarrow \alpha \rangle$ for \mathcal{N} -words α .

Proof:

1. (a) Consider an arbitrary proper \uparrow -subexpression E^s of E . By Condition ($\mathcal{D}_{\text{Min.1}}$), its parent operator cannot be \downarrow , and by Condition ($\mathcal{D}_{\text{Min.2}}$), its parent operator cannot be \uparrow . Hence, the parent operator of E^s is \downarrow .
 (b) The proof of this subclaim is analogous to that of the previous subclaim.
2. If E is equal to $\langle \uparrow \alpha \rangle$ or $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α , then the claim is trivially true, because such DNA expressions do not have proper DNA subexpressions.
 If E is not equal to $\langle \uparrow \alpha \rangle$ or $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α , then the claim follows from Condition ($\mathcal{D}_{\text{Min.3}}$).
3. Let E^s be a proper DNA subexpression of E . If E^s is a \downarrow -expression, then the claim follows from Condition ($\mathcal{D}_{\text{Min.1}}$). If E^s is a \uparrow -expression or a \downarrow -expression, then by Claim 2, E^s has at least two arguments. By Condition ($\mathcal{D}_{\text{Min.4}}$), at least one of these arguments is an \mathcal{N} -word.
4. (a) Consider an arbitrary proper \uparrow -subexpression E_2^s of E which is not the first argument of its parent operator. Let E_1^s be the preceding argument of the parent operator.
 By Claim 1a, the parent operator is \downarrow . By Condition ($\mathcal{D}_{\text{Min.5}}$), the first argument of E_2^s is either an \mathcal{N} -word α or a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α . If it were an \mathcal{N} -word α , then $L(E_2^s)$ would be in \mathcal{A}_+ and E_1^s and E_2^s would not fit together by lower strands, which is required by \downarrow . Hence, the first argument of E_2^s is a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α .
 The proof for a proper \downarrow -subexpression of E which is not the first argument of its parent operator is analogous.
 (b) The proof of this subclaim is analogous to that of the previous subclaim.
5. If E is equal to $\langle \uparrow \alpha \rangle$ or $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α , then again the claim is trivially true.
 Now assume that E is not equal to $\langle \uparrow \alpha \rangle$ or $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α . Consider an arbitrary proper \uparrow -subexpression E^s of E .

By Claim 1a, the parent operator of E^s is \downarrow , and by Condition ($\mathcal{D}_{\text{Min.3}}$), this operator \downarrow has at least two arguments. Hence, either E^s is not the first argument, or E^s is not the last argument of its parent operator (or both). By Claim 4, in the former case, the first argument of E^s is $\langle \uparrow \alpha \rangle$ for an \mathcal{N} -word α , and in the latter case, the last argument of E^s is $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α .

The proof for a proper \downarrow -subexpression E^s of E is analogous.

6. Let E_1^s be a proper \uparrow -subexpression of E which is neither the first argument, nor the last argument of E . By Claim 1a, E_1^s is the argument of a \downarrow -subexpression E_0^s of E .

If $E_0^s = E$, then E_1^s is neither the first argument, nor the last argument of its parent operator \downarrow . Hence by Claim 4, both the first argument and the last argument of E_1^s are \downarrow -expressions $\langle \downarrow \alpha \rangle$ for \mathcal{N} -words α .

If $E_0^s \neq E$, then the outermost operator of E_0^s is an inner occurrence of \downarrow . By Condition ($\mathcal{D}_{\text{Min.5}}$), (the \uparrow -expression) E_1^s cannot be the first argument or the last argument of E_0^s . Again by Claim 4, both the first argument and the last argument of E_1^s are \downarrow -expressions $\langle \downarrow \alpha \rangle$ for \mathcal{N} -words α .

By Claim 2, E_1^s has at least two arguments, which by Condition ($\mathcal{D}_{\text{Min.4}}$) are maximal \mathcal{N} -word occurrences and DNA expressions, alternately. Now the claim follows immediately.

The proof for a \downarrow -subexpression E_1^s of E is analogous.

□

As the notation suggests, \mathcal{D}_{Min} is exactly the language of minimal DNA expressions:

Theorem 4.100 *A DNA expression E is minimal if and only if $E \in \mathcal{D}_{\text{Min}}$*

Proof: \implies Let E be a minimal DNA expression, and let $X = \mathcal{S}(E)$. If E is a \uparrow -expression, then by Theorem 4.23, $E = \langle \uparrow \alpha \rangle$ for an \mathcal{N} -word α . It is easily verified that such a DNA expression satisfies all conditions from Definition 4.97, and thus is in \mathcal{D}_{Min} .

We will now prove that E also satisfies these conditions if it is a \uparrow -expression. The proof for a \downarrow -expression E is analogous.

($\mathcal{D}_{\text{Min.1}}$) Consider an arbitrary occurrence of the operator \uparrow in E and let E^s be the DNA subexpression of E governed by it. Because E is minimal, so is E^s . This implies that E^s is of the form $\langle \uparrow \alpha \rangle$ for an \mathcal{N} -word α .

($\mathcal{D}_{\text{Min.2}}$) Consider an arbitrary occurrence of the operator \uparrow in E and let E^s be the DNA subexpression of E governed by it. Because E is minimal, so is E^s . By Lemma 4.69, E^s does not have any \uparrow -argument.

Analogously, no occurrence of the operator \downarrow in E has a \downarrow -argument.

($\mathcal{D}_{\text{Min.3}}$) Assume that E is not equal to $\langle \uparrow \alpha \rangle$ or $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α .

Consider an arbitrary occurrence of the operator \uparrow in E and let E^s be the DNA subexpression of E governed by it. Because E is minimal, so is E^s .

Suppose that E^s has only one argument ε_1 : $E^s = \langle \uparrow \varepsilon_1 \rangle$. By Lemma 4.71, $\mathcal{S}^+(\varepsilon_1)$ is nick free. Now if ε_1 were a DNA expression E_1 , then by definition,

$\mathcal{S}(E^s) = \mathcal{S}(\langle \uparrow E_1 \rangle) = \nu^+(\mathcal{S}(E_1)) = \mathcal{S}(E_1)$. Because E_1 is 3 letters shorter than E^s , E^s would not be minimal. Consequently, ε_1 must be an \mathcal{N} -word α_1 : $E^s = \langle \uparrow \alpha_1 \rangle$.

Because $E \neq \langle \uparrow \alpha \rangle$ for any \mathcal{N} -word α , $E^s \neq E$, i.e., E^s is a proper DNA subexpression of E . By Conditions $(\mathcal{D}_{\text{Min.1}})$ and $(\mathcal{D}_{\text{Min.2}})$, the parent operator of (the \uparrow -expression) E^s is \downarrow . Because $\mathcal{S}(E^s) = \begin{pmatrix} \alpha_1 \\ - \end{pmatrix}$ does not fit together by lower strands with any other formal DNA molecule, E^s must be the only argument of its parent operator \downarrow . Hence, E has a DNA subexpression $\langle \downarrow E^s \rangle = \langle \downarrow \langle \uparrow \alpha_1 \rangle \rangle$. This DNA subexpression is, however, equivalent to the shorter DNA subexpression $\langle \uparrow \alpha_1 \rangle$, which contradicts the minimality of E .

Hence, our hypothesis that E^s has only one argument must be wrong.

The proof for an occurrence of the operator \downarrow in E is analogous.

$(\mathcal{D}_{\text{Min.4}})$ Consider an arbitrary inner occurrence of the operator \uparrow in E and let E_0^s be the DNA subexpression of E governed by it. E_0^s is the argument of a DNA subexpression E_1^s of E . Both E_0^s and E_1^s are minimal. Again, because E_0^s is a \uparrow -expression, Conditions $(\mathcal{D}_{\text{Min.1}})$ and $(\mathcal{D}_{\text{Min.2}})$ imply that E_1^s is a \downarrow -expression.

By (the analogue for \downarrow -expressions of) Lemma 4.71, the argument E_0^s of E_1^s is nick free. Then by Lemma 4.76(3), the arguments of E_0^s are maximal \mathcal{N} -word occurrences and DNA expressions, alternately.

The proof for an inner occurrence of the operator \downarrow in E is analogous.

$(\mathcal{D}_{\text{Min.5}})$ Consider an arbitrary inner occurrence of the operator \uparrow in E and let E_0^s be the DNA subexpression of E governed by it. E_0^s is the argument of a DNA subexpression E_1^s of E . Both E_0^s and E_1^s are minimal. Again, E_1^s is a \downarrow -expression and E_0^s is nick free.

Let $X_0^s = \mathcal{S}(E_0^s)$, and let $x'_1 \dots x'_k$ for some $k \geq 1$ be the decomposition of X_0^s . By (the analogue for \downarrow -expressions of) Lemma 4.74(1), X_0^s contains at least one single-stranded component and both the first single-stranded component and the last single-stranded component of X_0^s are upper components. Now by Lemma 4.45, both the maximal upper prefix and the maximal upper suffix of X_0^s are not empty.

By Theorem 4.77 and the construction from Theorem 4.53(1), the first argument of E_0^s corresponds to the first component of the maximal upper prefix of X_0^s , and it is either an \mathcal{N} -word α or a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α . Analogously, the last argument of E_0^s corresponds to the last component of the maximal upper suffix of X_0^s , and it is either an \mathcal{N} -word α or a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α .

The proof for an inner occurrence of the operator \downarrow in E is analogous.

$(\mathcal{D}_{\text{Min.6}})$ Assume that the outermost operator of E is \uparrow and that E does not have two consecutive arguments that are DNA expressions. Then by the definition of a maximal \mathcal{N} -word occurrence, the arguments of (the outermost operator of) E are maximal \mathcal{N} -word occurrences and DNA expressions, alternately. This implies that the outermost operator \uparrow of E does not introduce nick letters into $\mathcal{S}(E)$. Then by Corollary 4.72, $X = \mathcal{S}(E)$ is nick free.

The fact that E is a minimal \uparrow -expression restricts the possibilities for X . By Theorem 4.23, X must contain at least one single-stranded component, and by Lemma 4.58(2), $T_\uparrow(X) \geq T_\downarrow(X)$. Now by Lemma 4.12(1), either the first single-stranded component or the last single-stranded component of X (or both) is an upper component.

If the first single-stranded component of X is an upper component, then by Lemma 4.45, the maximal upper prefix of X is not empty. Then just like we did for E^s in the proof of Condition ($\mathcal{D}_{\text{Min.5}}$), we can prove that the first argument of E is either an \mathcal{N} -word α or a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α . Analogously, if the last single-stranded component of X is an upper component, then the last argument of E is either an \mathcal{N} -word α or a \downarrow -expression $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α .

The proof for the case that the outermost operator of E is \downarrow is analogous.

\Leftarrow Let E be an arbitrary DNA expression in \mathcal{D}_{Min} . We prove that E is minimal, by induction on the number p of operators occurring in E .

- If $p = 1$, then apparently the outermost operator of E is the only operator occurring in E . E does not have any argument that is a DNA expression. Hence, either $E = \langle \uparrow \alpha \rangle$, or $E = \langle \downarrow \alpha \rangle$ or $E = \langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α . Indeed, these are minimal DNA expressions, which denote $\binom{\alpha}{-}$, $\binom{-}{\alpha}$, or $\binom{\alpha}{c(\alpha)}$, respectively.
- Let $p \geq 1$, and suppose that each DNA expression in \mathcal{D}_{Min} that contains at most p operators is minimal (induction hypothesis).

Now consider a DNA expression E in \mathcal{D}_{Min} that contains $p+1$ operators. Because E contains $p+1 \geq 2$ operators, Condition ($\mathcal{D}_{\text{Min.1}}$) implies that E is not a \downarrow -expression.

Assume that E is a \uparrow -expression: $E = \langle \uparrow \varepsilon_1 \dots \varepsilon_n \rangle$ for some $n \geq 1$ and maximal \mathcal{N} -word occurrences and DNA expressions $\varepsilon_1, \dots, \varepsilon_n$. In fact, since E contains $p+1 \geq 2$ operators, it follows from Condition ($\mathcal{D}_{\text{Min.3}}$) that $n \geq 2$. Let $X = \mathcal{S}(E)$ and for $i = 1, \dots, n$, let $X_i = \mathcal{S}^+(\varepsilon_i)$. By definition,

$$X = \nu^+(X_1)y_1\nu^+(X_2)y_2 \dots y_{n-1}\nu^+(X_n),$$

where the y_i 's are as in (2.15).

By Condition ($\mathcal{D}_{\text{Min.2}}$), E does not have \uparrow -arguments. Hence, its arguments are maximal \mathcal{N} -word occurrences, \downarrow -expressions and \downarrow -expressions. Consider an arbitrary argument ε_i of E .

If ε_i is a maximal \mathcal{N} -word occurrence α , then $X_i = \binom{\alpha}{-}$. Because in this case $T_\downarrow(X_i) = n_\uparrow(X_i) = 0$,

$$|\varepsilon_i| = |\alpha| = |\nu(X_i)| = 3 \cdot T_\downarrow(X_i) + 3 \cdot n_\uparrow(X_i) + |\nu(X_i)|.$$

If ε_i is a \downarrow -expression, then by Condition ($\mathcal{D}_{\text{Min.1}}$), it is of the form $\langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α . In this case, $X_i = \binom{\alpha}{c(\alpha)}$, $T_\downarrow(X_i) = 0$ and $n_\uparrow(X_i) = 1$. Hence,

$$|\varepsilon_i| = |\langle \downarrow \alpha \rangle| = 3 + |\nu(X_i)| = 3 \cdot T_\downarrow(X_i) + 3 \cdot n_\uparrow(X_i) + |\nu(X_i)|.$$

Finally, assume that ε_i is a \downarrow -expression E_i . Because E_i does not contain the outermost operator \uparrow of E , it contains at most p operators. By Lemma 4.98, $E_i \in \mathcal{D}_{\text{Min}}$. Hence, by the induction hypothesis, E_i is a minimal DNA expression. Now by (the analogue for \downarrow -expressions of) Lemma 4.71, the arguments of E_i are nick free.

The outermost operator \downarrow of E_i is an inner occurrence of \downarrow in E . By Condition (\mathcal{D}_{Min} .4), its arguments are maximal \mathcal{N} -word occurrences and DNA expressions, alternately. This implies that the operator does not introduce nick letters. Consequently, $X_i = \mathcal{S}(E_i)$ does not contain any nick letters, at all.

By Lemma 4.99(2), E_i has at least two arguments. By Condition (\mathcal{D}_{Min} .5), the first one is either an \mathcal{N} -word α_1 or a \downarrow -expression $\langle \downarrow \alpha_1 \rangle$ for an \mathcal{N} -word α_1 . In the latter case, the second argument is an \mathcal{N} -word α_2 . In both cases, X_i contains at least one single-stranded component and the first single-stranded component is a lower component. Analogously, the last single-stranded component of X_i is a lower component.

Then by Lemma 4.12(4), $T_{\downarrow}(X_i) = T_{\uparrow}(X_i) + 1$. Consequently, by Theorem 4.53(2),

$$\begin{aligned} |\varepsilon_i| = |E_i| &= 3 + 3 \cdot T_{\uparrow}(X_i) + 3 \cdot n_{\downarrow}(X_i) + |\nu(X_i)| \\ &= 3 \cdot T_{\downarrow}(X_i) + 3 \cdot n_{\uparrow}(X_i) + |\nu(X_i)|. \end{aligned}$$

We are now ready to calculate the length of E :

$$\begin{aligned} |E| &= 3 + \sum_{i=1}^n |\varepsilon_i| \\ &= 3 + 3 \cdot \sum_{i=1}^n T_{\downarrow}(X_i) + 3 \cdot \sum_{i=1}^n n_{\uparrow}(X_i) + \sum_{i=1}^n |\nu(X_i)|. \end{aligned}$$

For $i = 1, \dots, n$, X_i is nick free and in particular, $\#_{\nabla}(X_i) = 0$. But then Equations (4.2), (4.3) and (4.4) from Lemma 4.16(1) imply that

$$|E| = 3 + 3 \cdot T_{\downarrow}(X) + 3 \cdot n_{\uparrow}(X) + |\nu(X)|. \quad (4.65)$$

If X contains nick letters, then by definition, these must be lower nick letters. By Theorem 4.67, $|E|$ equals the length of a minimal DNA expression denoting X . Hence, E is minimal itself.

Now assume that X is nick free. By Theorem 4.65(1), the length $|E|$ of (the \uparrow -expression) E is equal to the length of an operator-minimal \uparrow -expression denoting X . Hence, E is operator-minimal itself. The only thing left to be proved is that E is also *minimal* in this case.

We recall that $n \geq 2$. By Lemma 4.76(3), the arguments of E are maximal \mathcal{N} -word occurrences and DNA expressions, alternately. Further, by Condition (\mathcal{D}_{Min} .6), either the first argument, or the last argument of E (or both arguments) is an \mathcal{N} -word α_1 or a \downarrow -expression $\langle \downarrow \alpha_1 \rangle$ for an \mathcal{N} -word α_1 . Without loss of generality, assume that the first argument of E is an \mathcal{N} -word α_1 or a \downarrow -expression $\langle \downarrow \alpha_1 \rangle$ for an \mathcal{N} -word α_1 . In the latter case, the second argument must be an \mathcal{N} -word α_2 . In both cases, $X = \mathcal{S}(E)$ contains at least one single-stranded component and the first single-stranded component is an upper

Cond.	E	$X = \mathcal{S}(E)$	E^*
$(\mathcal{D}_{\text{Min.1}})$	$\langle \downarrow \langle \downarrow \alpha_1 \rangle \rangle$	$\begin{pmatrix} \alpha_1 \\ c(\alpha_1) \end{pmatrix}$	$\langle \downarrow \alpha_1 \rangle$
$(\mathcal{D}_{\text{Min.1}})$	$\langle \uparrow \alpha_1 \langle \downarrow \langle \uparrow \alpha_2 \langle \downarrow \alpha_3 \rangle \rangle \rangle \rangle$	$\begin{pmatrix} \alpha_1 \\ - \end{pmatrix} \begin{pmatrix} \alpha_2 \alpha_3 \\ c(\alpha_2 \alpha_3) \end{pmatrix}$	$\langle \uparrow \alpha_1 \langle \downarrow \alpha_2 \alpha_3 \rangle \rangle$
$(\mathcal{D}_{\text{Min.2}})$	$\langle \uparrow \alpha_1 \langle \uparrow \langle \downarrow \alpha_2 \rangle \alpha_3 \rangle \rangle$	$\begin{pmatrix} \alpha_1 \\ - \end{pmatrix} \begin{pmatrix} \alpha_2 \\ c(\alpha_2) \end{pmatrix} \begin{pmatrix} \alpha_3 \\ - \end{pmatrix}$	$\langle \uparrow \alpha_1 \langle \downarrow \alpha_2 \rangle \alpha_3 \rangle$
$(\mathcal{D}_{\text{Min.3}})$	$\langle \uparrow \langle \downarrow \alpha_1 \rangle \rangle$	$\begin{pmatrix} \alpha_1 \\ c(\alpha_1) \end{pmatrix}$	$\langle \downarrow \alpha_1 \rangle$
$(\mathcal{D}_{\text{Min.3}})$	$\langle \downarrow \alpha_1 \langle \uparrow \langle \downarrow \alpha_2 \rangle \rangle \rangle$	$\begin{pmatrix} - \\ \alpha_1 \end{pmatrix} \begin{pmatrix} \alpha_2 \\ c(\alpha_2) \end{pmatrix}$	$\langle \downarrow \alpha_1 \langle \downarrow \alpha_2 \rangle \rangle$
$(\mathcal{D}_{\text{Min.4}})$	$\langle \uparrow \alpha_1 \langle \downarrow \langle \downarrow \alpha_2 \rangle \langle \downarrow \alpha_3 \rangle \rangle \rangle$	$\begin{pmatrix} \alpha_1 \\ - \end{pmatrix} \begin{pmatrix} \alpha_2 \alpha_3 \\ c(\alpha_2 \alpha_3) \end{pmatrix}$	$\langle \uparrow \alpha_1 \langle \downarrow \alpha_2 \alpha_3 \rangle \rangle$
$(\mathcal{D}_{\text{Min.5}})$	$\langle \uparrow \alpha_1 \langle \downarrow \langle \uparrow \alpha_2 \langle \downarrow \alpha_3 \rangle \rangle \alpha_4 \rangle \rangle$	$\begin{pmatrix} \alpha_1 \alpha_2 \\ - \end{pmatrix} \begin{pmatrix} \alpha_3 \\ c(\alpha_3) \end{pmatrix} \begin{pmatrix} - \\ \alpha_4 \end{pmatrix}$	$\langle \uparrow \alpha_1 \alpha_2 \langle \downarrow \langle \downarrow \alpha_3 \rangle \alpha_4 \rangle \rangle$
$(\mathcal{D}_{\text{Min.5}})$	$\langle \uparrow \langle \downarrow \alpha_1 \langle \uparrow \langle \downarrow \alpha_2 \rangle \alpha_3 \rangle \rangle \alpha_4 \rangle$	$\begin{pmatrix} - \\ \alpha_1 \end{pmatrix} \begin{pmatrix} \alpha_2 \\ c(\alpha_2) \end{pmatrix} \begin{pmatrix} \alpha_3 \alpha_4 \\ - \end{pmatrix}$	$\langle \uparrow \langle \downarrow \alpha_1 \langle \downarrow \alpha_2 \rangle \rangle \alpha_3 \alpha_4 \rangle$
$(\mathcal{D}_{\text{Min.6}})$	$\langle \uparrow \langle \downarrow \alpha_1 \langle \downarrow \alpha_2 \rangle \rangle \alpha_3 \langle \downarrow \langle \downarrow \alpha_4 \rangle \alpha_5 \rangle \rangle$	$\begin{pmatrix} - \\ \alpha_1 \end{pmatrix} \begin{pmatrix} \alpha_2 \\ c(\alpha_2) \end{pmatrix} \begin{pmatrix} \alpha_3 \\ - \end{pmatrix} \begin{pmatrix} \alpha_4 \\ c(\alpha_4) \end{pmatrix} \begin{pmatrix} - \\ \alpha_5 \end{pmatrix}$	$\langle \downarrow \alpha_1 \langle \uparrow \langle \downarrow \alpha_2 \rangle \alpha_3 \langle \downarrow \alpha_4 \rangle \rangle \alpha_5 \rangle$

Table 4.3: Examples of DNA expressions that satisfy all conditions from Definition 4.97 except one. The first column mentions the condition that is not satisfied, the second column contains a corresponding DNA expression E , the third column gives the formal DNA molecule X denoted by E , and the fourth column contains a minimal DNA expression E^* denoting X . As usual, the α_i 's occurring represent (arbitrary) \mathcal{N} -words.

component. Then it follows from Lemma 4.12(1) and (3) that $T_{\uparrow}(X) \geq T_{\downarrow}(X)$, and from Theorem 4.53(1) that there exists at least one minimal \uparrow -expression denoting X . Consequently, the operator-minimal \uparrow -expression E is also minimal.

The proof for the case that E is a \downarrow -expression is analogous.

□

Definition 4.97 and Theorem 4.100 provide us with a characterization of the minimal DNA expressions by six conditions, Conditions $(\mathcal{D}_{\text{Min.1}}) - (\mathcal{D}_{\text{Min.6}})$. Of course, one may replace one or more of these conditions by other, new conditions, and yet retain a valid characterization. However, none of the six conditions can simply be omitted. For each of the conditions, there exist DNA expressions that do not satisfy that particular condition and thus are not minimal, although they do satisfy the other five conditions. Examples of this are given in Table 4.3.

It is instructive to *directly* relate each of the six conditions occurring in the definition of \mathcal{D}_{Min} to the concept of minimality. We do this in an intuitive way, using the qualification 'not efficient' for a certain type of DNA expression to indicate that it cannot be minimal. Because the arguments for occurrences of \downarrow in a DNA expression are analogous to those for occurrences of \uparrow , we will not consider occurrences of the operator \downarrow separately.

$(\mathcal{D}_{\text{Min.1}})$ The operator \downarrow complements all upper \mathcal{A} -letters and lower \mathcal{A} -letters occurring in its argument. Any argument that is a DNA expression denotes a formal DNA molecule, which either contains upper \mathcal{A} -letters or lower \mathcal{A} -letters, or does not contain any of these types of \mathcal{A} -letters. It is not efficient to first generate upper \mathcal{A} -letters or lower \mathcal{A} -letters by means of the operators \uparrow and \downarrow , and then to complement them by means of \downarrow . It is not efficient either to apply \downarrow to an argument having only double \mathcal{A} -letters and nick letters, because then the operator

would not have any effect.

($\mathcal{D}_{\text{Min.2}}$) It is not efficient to apply the operator \uparrow to a \uparrow -argument, because all effects of the inner occurrence of \uparrow (creating upper \mathcal{A} -words, removing upper nick letters and joining the arguments) can also be achieved by the outermost occurrence of \uparrow .

($\mathcal{D}_{\text{Min.3}}$) When the operator \uparrow is applied to a single argument that is a DNA expression, the only effect is that upper nick letters occurring in the argument (if any) are removed. It is certainly not efficient to apply \uparrow to a nick free DNA expression, because then the operator would have no effect at all. It is not efficient either to first generate upper nick letters (which separate double components) and then to remove them (and to join the double components).

When the operator \uparrow is applied to a single argument that is an \mathcal{N} -word α , it generates an upper \mathcal{A} -word $\binom{\alpha}{-}$. For an inner occurrence of \uparrow in a minimal DNA expression, the parent operator of the corresponding DNA subexpression $\langle \uparrow \alpha \rangle$ is \downarrow . Because the arguments of \downarrow must fit together by lower strands, $\langle \uparrow \alpha \rangle$ must be the only argument of \downarrow . In that case, the operator \downarrow does not have any effect, which is not efficient.

($\mathcal{D}_{\text{Min.4}}$) As we mentioned before, it is not efficient to first generate lower nick letters by the application of \uparrow , and then to remove them by the application of \downarrow . Consider an inner occurrence \uparrow_1 of the operator \uparrow in a minimal DNA expression, and let E^s be the DNA subexpression governed by it.

Because E^s is the argument of an occurrence of \downarrow , $\mathcal{S}(E^s)$ must not contain lower nick letters (and thus must be nick free altogether). In particular, \uparrow_1 must not introduce lower nick letters. We can achieve this by requiring that the arguments of this occurrence of \uparrow are maximal \mathcal{N} -word occurrences and DNA expressions, alternately.

($\mathcal{D}_{\text{Min.5}}$) Let \uparrow_1 be an inner occurrence of the operator \uparrow in a minimal DNA expression, and let $E_1^s = \langle \uparrow_1 \varepsilon_1 \dots \varepsilon_n \rangle$ for some $n \geq 1$ and \mathcal{N} -words and DNA expressions $\varepsilon_1, \dots, \varepsilon_n$ be the DNA subexpression governed by it. E_1^s is the argument of a \downarrow -expression E_0^s .

Suppose that the last argument ε_n of \uparrow_1 is a \downarrow -argument, hence that E_0^s has the following shape:

$$E_0^s = \langle \downarrow_0 \dots \langle \uparrow_1 \varepsilon_1 \dots \varepsilon_{n-1} \langle \downarrow_2 \varepsilon_{n,1} \dots \varepsilon_{n,m_n} \rangle \rangle \dots \rangle$$

for some $m_n \geq 1$ and \mathcal{N} -words and DNA expressions $\varepsilon_{n,1}, \dots, \varepsilon_{n,m_n}$. Then the effects of \downarrow_2 can as well be achieved by the operator \downarrow_0 , by taking the last $m_n - 1$ arguments away from \downarrow_2 and making them direct arguments of \downarrow_0 . Indeed, by Theorem 3.12 (and Lemma 3.7 and Lemma 3.6),

$$\begin{aligned} E_0^s &= \langle \downarrow_0 \dots \langle \uparrow_1 \varepsilon_1 \dots \varepsilon_{n-1} \langle \downarrow_2 \varepsilon_{n,1} \dots \varepsilon_{n,m_n} \rangle \rangle \dots \rangle \\ &\equiv \langle \downarrow_0 \dots \langle \downarrow_2 \langle \uparrow_1 \varepsilon_1 \dots \varepsilon_{n-1} \varepsilon_{n,1} \rangle \varepsilon_{n,2} \dots \varepsilon_{n,m_n} \rangle \dots \rangle \\ &\equiv \langle \downarrow_0 \dots \langle \uparrow_1 \varepsilon_1 \dots \varepsilon_{n-1} \varepsilon_{n,1} \rangle \varepsilon_{n,2} \dots \varepsilon_{n,m_n} \dots \rangle \end{aligned} \tag{4.66}$$

We may assume that in E_0^s , the operator \uparrow_1 does not remove upper nick letters from its arguments. In particular, its last argument $\varepsilon_n = \langle \downarrow_2 \varepsilon_{n,1} \dots \varepsilon_{n,m_n} \rangle$ is nick free. This implies that E_0^s is (strictly) equivalent to the result in (4.66).

Consequently, the operator \downarrow_2 in E_0^s is superfluous. We conclude that it is not efficient if the last argument of \uparrow_1 is a \downarrow -argument. Analogously, it is not efficient if the first argument of \uparrow_1 is a \downarrow -argument.

($\mathcal{D}_{\text{Min.6}}$) Let $E = \langle \uparrow_0 \varepsilon_1 \dots \varepsilon_n \rangle$ for some $n \geq 1$ and maximal \mathcal{N} -word occurrences and DNA expressions $\varepsilon_1, \dots, \varepsilon_n$ be a minimal \uparrow -expression and assume that E does not have two consecutive arguments that are DNA expressions. Then $\varepsilon_1, \dots, \varepsilon_n$ are maximal \mathcal{N} -word occurrences and DNA expressions, alternately. Hence, the outermost operator \uparrow_0 of E does not introduce nick letters.

We may assume that the DNA expressions among the arguments of E are \downarrow -expressions and \uparrow -expressions $\langle \uparrow \alpha \rangle$ for an \mathcal{N} -word α . By definition, the semantics of such arguments do not contain lower nick letters. Hence (the \uparrow -expression) E is nick free altogether.

Suppose that both the first argument and the last argument of E are \downarrow -expressions:

$$\begin{aligned} \varepsilon_1 &= \langle \downarrow_1 \varepsilon_{1,1} \dots \varepsilon_{1,m_1} \rangle, \text{ and} \\ \varepsilon_n &= \langle \downarrow_n \varepsilon_{n,1} \dots \varepsilon_{n,m_n} \rangle \end{aligned}$$

for $m_1, m_n \geq 1$ and \mathcal{N} -words and DNA expressions $\varepsilon_{1,1}, \dots, \varepsilon_{1,m_1}, \varepsilon_{n,1}, \dots, \varepsilon_{n,m_n}$. We may assume that $n \geq 2$. Because the arguments of \uparrow_0 must fit together by upper strands, the arguments ε_{1,m_1} and $\varepsilon_{n,1}$ are DNA expressions (and in fact, we may assume that they are \uparrow -expressions $\langle \uparrow \alpha \rangle$ for an \mathcal{N} -word α).

Now by a twofold application of Theorem 3.12 (combined with Lemma 3.7 and Lemma 3.6), we find

$$\begin{aligned} E &= \langle \uparrow_0 \langle \downarrow_1 \varepsilon_{1,1} \dots \varepsilon_{1,m_1} \rangle \varepsilon_2 \dots \varepsilon_{n-1} \langle \downarrow_n \varepsilon_{n,1} \dots \varepsilon_{n,m_n} \rangle \rangle \\ &\overline{\nabla} \langle \downarrow_1 \varepsilon_{1,1} \dots \varepsilon_{1,m_1-1} \langle \uparrow_0 \varepsilon_{1,m_1} \varepsilon_2 \dots \varepsilon_{n-1} \langle \downarrow_n \varepsilon_{n,1} \dots \varepsilon_{n,m_n} \rangle \rangle \rangle \\ &\overline{\nabla} \langle \downarrow_1 \varepsilon_{1,1} \dots \varepsilon_{1,m_1-1} \langle \downarrow_n \langle \uparrow_0 \varepsilon_{1,m_1} \varepsilon_2 \dots \varepsilon_{n-1} \varepsilon_{n,1} \rangle \varepsilon_{n,2} \dots \varepsilon_{n,m_n} \rangle \rangle \\ &\equiv \langle \downarrow \varepsilon_{1,1} \dots \varepsilon_{1,m_1-1} \langle \uparrow_1 \varepsilon_{1,m_1} \varepsilon_2 \dots \varepsilon_{n-1} \varepsilon_{n,1} \rangle \varepsilon_{n,2} \dots \varepsilon_{n,m_n} \rangle \end{aligned} \quad (4.67)$$

We may assume that in E , the operator \uparrow_0 does not remove upper nick letters. In particular, its arguments $\varepsilon_1 = \langle \downarrow_1 \varepsilon_{1,1} \dots \varepsilon_{1,m_1} \rangle$ and $\varepsilon_n = \langle \downarrow_n \varepsilon_{n,1} \dots \varepsilon_{n,m_n} \rangle$ are nick free. This implies that also the resulting DNA expression E' in (4.67) is nick free. Hence, it is (strictly) equivalent to E .

In E' , one occurrence of the operator \downarrow has the same effect as the two occurrences \downarrow_1 and \downarrow_n in E . We conclude that it is not efficient if both the first argument and the last argument of E are \downarrow -expressions.

4.2.7 Trees of minimal DNA expressions

As we observed in § 2.11, each DNA expression has a unique representation as an ordered, directed, node-labelled tree. In particular, such a unique tree-representation exists for every minimal DNA expression. The resulting trees are also called minimal.

In § 4.2.6, we have proved that minimal DNA expressions can be characterized by six conditions on the operators occurring in them. These conditions can be directly translated into conditions characterizing minimal trees. Also other results on minimal DNA expressions can be translated into tree-terminology. For several results, we will now give the tree versions.

Let t be the tree of a DNA expression E .

Corollary 4.21 t is minimal if and only if the tree of every DNA expression E' with $E' \equiv E$ contains at least as many internal nodes as t .

Lemma 4.22 and Lemma 4.98 t is minimal if and only if each subtree of t rooted in an internal node of t is minimal.

Theorem 4.100 (and Definition 4.97) t is minimal if and only if

- ($\mathcal{D}_{\text{Min.1}}$) each node labelled by \updownarrow in t has a child labelled by an \mathcal{N} -word α , and
- ($\mathcal{D}_{\text{Min.2}}$) no node labelled by \uparrow in t has a child labelled by \uparrow and no node labelled by \downarrow in t has a child labelled by \downarrow , and
- ($\mathcal{D}_{\text{Min.3}}$) unless $E = \langle \uparrow \alpha \rangle$ or $E = \langle \downarrow \alpha \rangle$ for an \mathcal{N} -word α , each node labelled by either \uparrow or \downarrow in t has at least two children, and
- ($\mathcal{D}_{\text{Min.4}}$) for each non-root labelled by either \uparrow or \downarrow in t , the children are labelled by an \mathcal{N} -word α or by an operator, alternately, and
- ($\mathcal{D}_{\text{Min.5}}$) for each non-root labelled by either \uparrow or \downarrow in t , the first child is labelled by either an \mathcal{N} -word α or the operator \updownarrow , and also the last child is labelled by either an \mathcal{N} -word α or the operator \updownarrow , and
- ($\mathcal{D}_{\text{Min.6}}$) if the root of t is labelled by either \uparrow or \downarrow , then either it has two consecutive children labelled by an operator, or its first child is labelled by an \mathcal{N} -word α or the operator \updownarrow , or its last child is labelled by an \mathcal{N} -word α or the operator \updownarrow .

Lemma 4.99 If t is minimal, then

1. (a) each non-root labelled by \uparrow in t has as parent a node labelled by \downarrow ;
(b) each non-root labelled by \downarrow in t has as parent a node labelled by \uparrow ;
2. each non-root labelled by either \uparrow or \downarrow in t has at least two children;
3. each internal node which is not the root of t has at least one child labelled by an \mathcal{N} -word α ;
4. (a) each non-root labelled by either \uparrow or \downarrow in t which is not the the first child of its parent has as its (own) first child a node labelled by \updownarrow ;
(b) each non-root labelled by either \uparrow or \downarrow in t which is not the the last child of its parent has as its (own) last child a node labelled by \updownarrow ;
5. for each non-root labelled by either \uparrow or \downarrow in t , either the first child, or the last child is labelled by \updownarrow ;
6. each non-root labelled by either \uparrow or \downarrow in t which is neither the first child of the root, nor the last child of the root, has an odd number of children (at least three), the first one and the last one of which are labelled by \updownarrow .

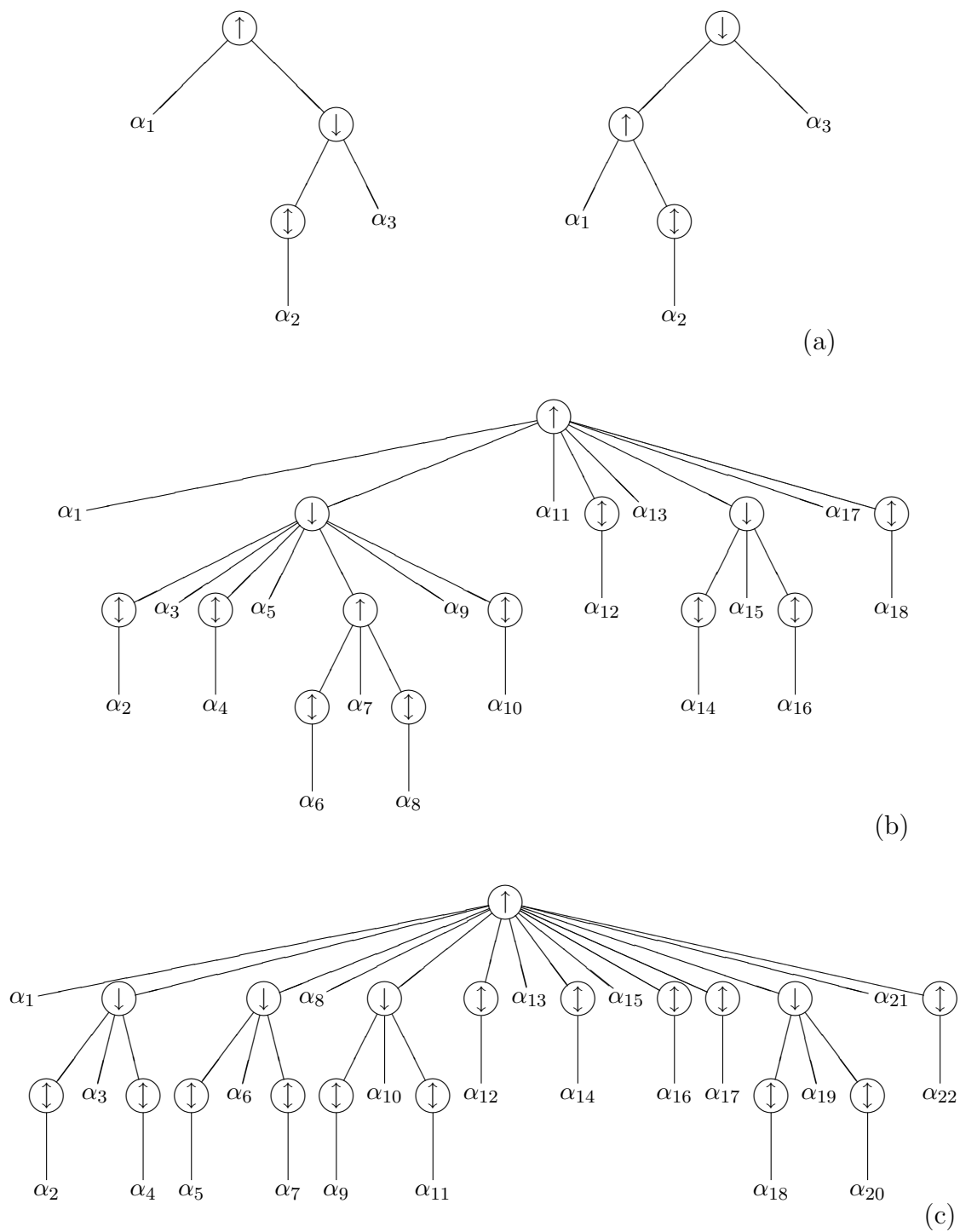


Figure 4.10: Trees of four minimal DNA expressions: (a) the trees of the two equivalent minimal DNA expressions from Equation (4.16), denoting the nick free formal DNA molecule $X = \begin{pmatrix} \alpha_1 \\ - \end{pmatrix} \begin{pmatrix} \alpha_2 \\ c(\alpha_2) \end{pmatrix} \begin{pmatrix} - \\ \alpha_3 \end{pmatrix}$; (b) the tree of the minimal DNA expression from Equation (4.20), denoting the nick free formal DNA molecule from (a.o.) Figure 4.4; (c) the tree of the minimal DNA expression from Equation (4.36), denoting the formal DNA molecule from Figure 4.9, which contains four lower nick letters.

In Figure 4.10, we have drawn the trees of four minimal DNA expressions we have considered or constructed in the course of § 4.2. One can verify that these trees exhibit all properties we have just listed. In the last two trees, especially the last property, corresponding to Lemma 4.99(6), comes out clearly.

Chapter 5

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 do 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 some ideas about 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 extend the set of operators that can be used 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.

Appendix A

Recognition of DNA expressions (an implementation)

```
// check if a string over the operators, opening and closing brackets,  
// the letters A, C, G and T and words alpha is a DNA-expression.  
  
//*****  
  
enum Symbol {Upa, Doa, Uda, Openbr, Closebr, A, C, G, T, alpha, NulSymbol};  
  
//*****  
  
bool Operator (Symbol X)  
  // check if the symbol X is an operator  
{  
  return ((X>=Upa) && (X<=Uda));  
  
} // Operator  
  
//*****  
  
bool WordElt (Symbol X)  
  // check if the symbol X is (an element of) a word alpha  
{  
  return ((X>=A) && (X<=alpha));  
  
} // WordElt  
  
//*****  
  
bool CheckBrackets (String E)  
  // check if the brackets and the operators in E are positioned correctly  
{ bool Check;  
  int i, // position in the string  
      Level; // Level of the string  
  Symbol LastSymbol;
```

```

if (E[1]!=Openbr)
{ cout << "The string does not start with an opening bracket.\n";
  // note that this covers the empty string, as well
  Check = false;
}
else // E starts with an opening bracket,
      // so in the first iteration of the following do-loop,
      // the level is raised to 1
{ Check = true;
  i=1;
  LastSymbol=NulSymbol;
  Level=0;
  do
  { if ((LastSymbol==Openbr) && (!Operator(E[i])))
    { cout << "Opening bracket at position " << i-1
      << " is not followed by an operator.\n";
      Check = false;
    }
    else // nothing wrong
    { switch (E[i])
      { case Openbr: Level ++;
        break;
        case Closebr: Level --;
        break;
        default: ;
      } // switch

      LastSymbol = E[i];
      i++;
    } // else: nothing wrong

  } while (Check && (E[i]!=NulSymbol) && (Level!=0));

} // else: E starts with an opening bracket

if (Check)
  if (Level!=0) // we must have E[i]==NulSymbol
  { cout << "The string ended at level " << Level << ".\n";
    return false;
  }
  else // Level=0
  { if (E[i] != NulSymbol)
    { cout << "Level 0 was reached at position " << i-1
      << " before the end of the string.\n";
      return false;
    }
    else // Check && E[i]==NulSymbol && Level=0
      return true;
  }
}

```

```

    else // !Check
        return false;

} // CheckBrackets

//*****

bool CheckExpr (String E, int &i, NPTYPE &L0, NPTYPE &R0)
// check if the substring between E[i] (an opening bracket)
// and the corresponding closing bracket is a DNA-expression;
// assumptions:
// * E satisfies conditions related to the opening brackets
//   and the closing brackets;
// * E[i] is an opening bracket
{ Symbol Oper;
  int Operi, // position of the operator
    n; // number of arguments of the operator
  bool Check;
  NPTYPE L1, R1; // types of leftmost and rightmost nucleotide pair
                // of an argument of the operator

  i++; // go to next symbol (past the opening bracket)
  Oper = E[i]; // by assumption, Oper is indeed an operator
  Operi = i; // remember this position

  i++; // go to next symbol (past the operator)
  Check = true;
  n=0;
  while (Check && (E[i] != Closebr))
  { n++; // another argument

    if ((Oper==Uda) && (n>1))
    { cout << "Operator Uda with more than one argument at position "
      << Operi << ".\n";
      Check = false;
    }
    else // this argument is in principle allowed

    { if (WordElt(E[i])) // this argument is a word
      { do
          i++;
          while (WordElt(E[i]));

          switch (Oper)
          { case Upa: L1 = Pp;
              R1 = Pp;
              break;
            case Doa: L1 = Pm;
              R1 = Pm;
          }
      }
    }
  }
}

```

```

        break;
    case Uda: L1 = P;
             R1 = P;
    } // switch
}
else // this argument is a DNA-expression
    Check = Expression (E, i, L1, R1);

if (Check) // nothing went wrong yet
    if (Oper==Uda) // apparently, n==1
    { L0 = P;
      R0 = P;
    }
    else // operator = Upa or Doa
    if (n==1) // first argument
    { L0 = L1;
      R0 = R1;
    }
    else // n>1
    switch (Oper)
    { case Upa: if ((R0!=Pm) && (L1!=Pm)) // upper prefit
                R0 = R1;
              else // not an upper prefit
                { cout << n-1
                  << "-th Argument is not an upper prefit for "
                  << n
                  << "-th argument of operator Upa at position "
                  << Operi << ".\n";
                  Check = false;
                }
              break;
        case Doa: if ((R0!=Pp) && (L1!=Pp)) // lower prefit
                  R0 = R1;
                else // not a lower prefit
                  { cout << n-1
                    << "-th Argument is not a lower prefit for "
                    << n
                    << "-th argument of operator Doa at position "
                    << Operi << ".\n";
                    Check = false;
                  }
        } // switch

} // else: argument in principle allowed

} // while

if (Check) // E[i] is a closing bracket
    if (n==0)

```



```

    { cout << "Operator with no arguments at position " << Operi << ".\n";
      return false;
    }
    else // n>=1
    { i++; // go to next symbol (past the closing bracket)
      return true;
    }
  else
    return false;
} // CheckExpr

//*****

bool Check (String E)
// check if the string E is a DNA-expression
{ int i;
  NPType L1, R1; // types of leftmost and rightmost nucleotide pair

  if (CheckBrackets (E))
  { i=1;
    if (CheckExpr (E, i, L1, R1))
      return true;
    else
      return false;
  }
  else
    return false;
} // Check

//*****

```

List of symbols

symbol	introduced on page	symbol	introduced on page
\sqsubseteq	10	$\kappa(X)$	11
$\overline{\square}$	10	λ	3
\square	10	$L(X)$	3
\equiv	27	\mathcal{M}	75
$\overline{\nabla}$	27	$\nu(X)$	11
$\nabla \equiv$	27	$\nu^+(X)$	11
$\equiv \nabla$	27	$\nu^-(X)$	11
\uparrow	13	$n_{\uparrow}(X)$	44
\downarrow	14	$n_{\text{imus}}(X)$	74
\updownarrow	14	$n_{\text{min}}(X)$	112
∇	6	$n_{\text{min}\uparrow}(X)$	111
\triangle	6	$n_{\text{min}\downarrow}(X)$	111
$\#_a(X)$	4	$n_{\text{min}\uparrow}(X)$	111
$\#_{a,b}(X)$	4	$n_{\text{opermin}\uparrow}(X)$	112
α, α_i	3	$n_{\text{opermin}\downarrow}(X)$	112
a, a_i	3	$n_{\text{opermin}\uparrow}(X)$	112
\mathcal{A}	6	$n_{\text{pb}}(Z)$	121
\mathcal{A}_{\pm}	6	\mathcal{N}	3
\mathcal{A}_+	6	\mathcal{O}	12
\mathcal{A}_-	6	$R(X)$	3
$\mathcal{A}_{\nabla\Delta}$	6	xRy	26
$c(a)$	5	$\Sigma_{\mathcal{D}}$	12
C_p	123	Σ^*	3
\mathcal{D}	12, 18	Σ^+	3
\mathcal{D}_{Min}	133	$\mathcal{S}(E)$	13
$\varepsilon, \varepsilon_i$	12	$\mathcal{S}^+(\varepsilon)$	15
E, E_i	12	$\mathcal{S}^-(\varepsilon)$	16
$\text{Exp}^+(\varepsilon)$	17	(t_1, \dots, t_r)	117
$\text{Exp}^-(\varepsilon)$	17	$T_{\uparrow}(X)$	44
$f_1(E)$	125	$T_{\downarrow}(X)$	44
$f_2(E)$	125	$T_-(E)$	124
$f_{T_{\uparrow}}(\mathcal{M})$	116	$\mathcal{W}_{\mathcal{A}}(\alpha)$	8
$f_{T_{\downarrow}}(\mathcal{M})$	116	$ X $	3
\mathcal{F}	7	$(X^s)^k$	4

Index

term	introduced on page
\uparrow -argument	20
\uparrow -component	41
closing \sim	41
corresponding \sim	42
opening \sim	41
corresponding \sim	42
\uparrow -expression	16
\uparrow -subexpression	19
\downarrow -argument	20
\downarrow -component	41
closing \sim	41
corresponding \sim	43
opening \sim	41
corresponding \sim	43
\downarrow -expression	16
\downarrow -subexpression	19
\updownarrow -argument	20
\updownarrow -expression	16
\updownarrow -subexpression	19
\mathcal{A} -letter	6
double \sim	6
lower \sim	6
upper \sim	6
alphabet	3
ancestor of node	24
antisymmetric binary relation	26
apply operator	13
argument	
\uparrow - \sim	20
\downarrow - \sim	20
\updownarrow - \sim	20
\sim of DNA expression	19
\sim of operator	13
\mathcal{N} -word- \sim	20
\mathcal{A} -word	6
\sim s determined by α	8
double \sim	6
lower \sim	6
upper \sim	6
balls in urns	120
base pair	5
complete \sim	5
binary relation	26
antisymmetric \sim	26
reflexive \sim	26
symmetric \sim	26
transitive \sim	26
block	62
brackets	
sequence of well-nested	
pairs of \sim	120
Catalan numbers	123, 124, 133
child of node	24
closing	
\sim \uparrow -component	41
corresponding \sim	42
\sim \downarrow -component	41
corresponding \sim	43
\sim bracket of operator	12
\sim lower component	61
\sim upper component	61
commutative composition	
of functions	11
complement function	5
complete	
\sim base pair	5
\sim maximal	
upper partitioning	75
component of formal	
DNA molecule	9
\uparrow - \sim	41
closing \sim	41
opening \sim	41
\downarrow - \sim	41
closing \sim	41
opening \sim	41
double \sim	9
lower \sim	9
closing \sim	61
opening \sim	61
single-stranded \sim	9
upper \sim	9
closing \sim	61
opening \sim	61
composition of functions	

- commutative \sim 11
- concatenation
 - \sim of DNA expressions 24
 - \sim of formal DNA molecules ... 10
 - \sim of strings 3
 - \sim of words 3
- contain
 - substring \sim s other substring ... 4
- corresponding
 - \sim closing \uparrow -component 42
 - \sim closing \downarrow -component 43
 - \sim opening \uparrow -component 42
 - \sim opening \downarrow -component 43
- counting function 41, 44
- cover to the left/right 10
 - strictly \sim 10
- decomposition of formal
 - DNA molecule 8
 - nick free \sim 89
- depth first search walk 26
- descendant of node 24
- determined
 - \mathcal{A} -words \sim by α 8
- directed
 - \sim tree 24
 - ordered \sim 24
 - ordered, \sim ,
 - node-labelled tree 24
- disjoint substrings 4
- DNA expression 13, 15
 - argument of \sim 19
 - concatenation of \sim s 24
 - \sim is not efficient 140
 - \sim of tree 26
 - equivalent \sim s 27
 - length of \sim 39
 - level of \sim 20
 - minimal \sim 58
 - operator-minimal \sim 91
 - semantics of \sim 13, 15
 - tree of \sim 25
 - type of \sim 18
- DNA molecule
 - formal \sim 7
- DNA subexpression 19
 - proper \sim 19
- DNA submolecule
 - formal \sim 7
- double
 - \sim \mathcal{A} -letter 6
 - \sim \mathcal{A} -word 6
 - \sim complete formal
 - DNA molecule 60
 - \sim component of formal
 - DNA molecule 9
- dual relation 26
- efficient
 - DNA expression is not \sim 140
- empty string 3
- endomorphism 5
- equivalence relation 26
- equivalent DNA expressions 27
 - \sim modulo nicks 27
 - \sim post-modulo nicks 27
 - \sim pre-modulo nicks 27
 - strictly \sim 27
- expressible formal
 - DNA molecule 29
- expression
 - \uparrow - \sim 16
 - \downarrow - \sim 16
 - \updownarrow - \sim 16
- extension of relation 26
- fit together 10
- formal
 - \sim DNA molecule 7
 - component of \sim 9
 - concatenation of \sim s 10
 - decomposition of \sim 8
 - double-complete \sim 60
 - expressible \sim 29
 - lower strand of \sim 7
 - transition in \sim 41
 - upper strand of \sim 7
 - \sim DNA submolecule 7
- govern 19
- homomorphism 4
- inner occurrence of operator 19
- internal
 - \sim maximal upper sequence 74
 - \sim node 24
- intersecting substrings 4
- labelled
 - ordered, directed,
 - node- \sim tree 24
- language 3
- leaf in tree 24
- length
 - of DNA expression 39
 - lower bound on \sim 40, 57
 - upper bound on \sim 40
- of string 3
 - upper bound on \sim of
 - minimal DNA expression ... 86

- letter 3
- level
 - ~ of DNA expression 20
 - nesting ~ 20
- lower
 - ~ \mathcal{A} -letter 6
 - ~ \mathcal{A} -word 6
 - ~ bound on length of
 - DNA expression 40, 57
 - tight ~ 40
 - ~ component of formal
 - DNA molecule 9
 - closing ~ 61
 - opening ~ 61
 - ~ nick letter 6
 - ~ postfit 10
 - ~ prefit 10
 - ~ semantics of argument 17
 - ~ strand 7
- maximal ~
 - ~ partitioning 76
 - ~ prefix 76
 - ~ sequence 76
 - ~ suffix 76
- separating ~ sequence 66
- maximal
 - ~ lower
 - ~ partitioning 76
 - ~ prefix 76
 - ~ sequence 76
 - ~ suffix 76
 - ~ \mathcal{N} -word occurrence 19
 - ~ upper
 - ~ partitioning 75
 - complete ~ 75
 - ~ prefix 72
 - ~ sequence 61
 - internal ~ 74
 - ~ suffix 73
- minimal
 - ~ DNA expression 58
 - operator-~ 91
 - upper bound
 - on length of ~ 86
 - ~ tree 142
- modulo
 - equivalent ~ nicks 27
- nesting level 20
- nick
 - equivalent modulo ~s 27
 - equivalent post-modulo ~s 27
 - equivalent pre-modulo ~s 27
- ~ free 7
- ~ decomposition 89
- ~ letter 6
 - lower ~ 6
 - upper ~ 6
- \mathcal{N} -letter 3
- node
 - ancestor of ~ 24
 - child of ~ 24
 - descendant of ~ 24
 - internal ~ 24
 - ordered, directed,
 - ~labelled tree 24
 - parent of ~ 24
- non-root 24
- notation
 - prefix ~ 17
 - simplified ~ 8
- \mathcal{N} -word 3
 - concatenation of ~s 3
 - maximal ~ occurrence 19
 - ~argument 20
- occurrence
 - inner ~ of operator 19
 - maximal \mathcal{N} -word ~ 19
 - ~ of substring 3
 - preceding ~ 3
- opening
 - ~ \uparrow -component 41
 - corresponding ~ 42
 - ~ \downarrow -component 41
 - corresponding ~ 43
 - ~ bracket of operator 12
 - ~ lower component 61
 - ~ upper component 61
- operator 12
 - apply ~ 13
 - argument of ~ 13
 - closing bracket of ~ 12
 - inner occurrence of ~ 19
 - opening bracket of ~ 12
 - ~minimal DNA expression ... 91
 - outermost ~ 19
 - parent ~ 19
 - scope of ~ 12
- ordered
 - ~, directed
 - ~ tree 24
 - ~, node-labelled tree 24
 - ~ partition 117
- outermost operator 19
- overlapping substrings 4

- parent
 - ~ of node 24
 - ~ operator 19
- partial order 26
- partition
 - ordered ~ 117
- partitioning
 - maximal lower ~ 76
 - maximal upper ~ 75
 - complete ~ 75
- postfit
 - lower ~ 10
 - ~ by lower strands 10
 - ~ by upper strands 10
 - upper ~ 10
- post-modulo
 - equivalent ~ nicks 27
- preceding occurrence of substring .. 3
- prefit 10
 - lower ~ 10
 - ~ by lower strands 10
 - ~ by upper strands 10
 - upper ~ 10
- prefix
 - maximal lower ~ 76
 - maximal upper ~ 72
 - ~ notation 17
 - ~ of string 3
- pre-modulo
 - equivalent ~ nicks 27
- preorder walk 26
- proper
 - ~ DNA subexpression 19
 - ~ substring 3
- refinement 26
- reflexive binary relation 26
- relation
 - binary ~ 26
 - antisymmetric ~ 26
 - reflexive ~ 26
 - symmetric ~ 26
 - transitive ~ 26
 - dual ~ 26
 - equivalence ~ 26
- root of tree 24
- rotation in tree 37
- scope of operator 12
- semantics
 - lower ~ of argument 17
 - ~ of DNA expression 13, 15
 - upper ~ of argument 17
- separating
 - ~ lower sequence 66
 - ~ upper sequence 76
- sequence of well-nested
 - pairs of brackets 120
- single-stranded component of
 - formal DNA molecule 9
- strand
 - lower ~ 7
 - upper ~ 7
- strictly
 - ~ cover to the left/right 10
 - ~ equivalent
 - DNA expressions 27
- string 3
 - empty ~ 3
 - length of ~ 3
- subexpression
 - \uparrow -~ 19
 - \downarrow -~ 19
 - \updownarrow -~ 19
- substring 3
 - disjoint ~s 4
 - intersecting ~s 4
 - occurrence of ~ 3
 - overlapping ~s 4
 - proper ~ 3
 - ~ contains other ~ 4
- subtree rooted in node 24
- subword
 - maximal ~ 19
- suffix
 - maximal lower ~ 76
 - maximal upper ~ 73
 - ~ of string 3
- symbol 3
- symmetric binary relation 26
- tight lower bound 40
- transition between operators 41
- transitive binary relation 26
- tree
 - directed ~ 24
 - ordered, ~ 24
 - DNA expression of ~ 26
 - leaf in ~ 24
 - minimal ~ 142
 - ordered, directed,
 - node-labelled ~ 24
 - root of ~ 24
 - rotation in ~ 37
 - ~ of DNA expression 25
- type of DNA expression 18
- upper

- maximal ~
 - ~ partitioning 75
 - complete ~ 75
 - ~ prefix 72
 - ~ sequence 61
 - internal ~ 74
 - ~ suffix 73
 - separating ~ sequence 76
 - ~ \mathcal{A} -letter 6
 - ~ \mathcal{A} -word 6
 - ~ bound on length of
 - ~ DNA expression 40
 - ~ minimal
 - DNA expression 86
 - ~ component of formal
 - DNA molecule 9
 - closing ~ 61
 - opening ~ 61
 - ~ nick letter 6
 - ~ postfit 10
 - ~ prefit 10
 - ~ semantics of argument 17
 - ~ strand 7
- urns
- balls in ~ 120
- well-nested
- sequence of ~ pairs
 - of brackets 120

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, April 4, 1995, Princeton University* (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, DNA9, Madison, WI, USA, June 1-3, 2003 – Revised papers*, Lecture Notes in Computer Science **2943**, Springer-Verlag, Berlin, 2004.
- M. Daley, L. Kari, I. McQuillan: Families of languages defined by ciliate bio-operations, Technical Report #2004-a, Theory & Formal Bioinformatics Group, University of Saskatchewan (2003).
- 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, DNA8, Sapporo, Japan, June 2002 – Revised Papers*, Lecture Notes in Computer Science **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, University of Pennsylvania, Philadelphia, USA, June 15-19, 1998*, *BioSystems* **52** (1999), 3-13.
- Z. Li: Algebraic properties of DNA operations, *Proceedings of the fourth international meeting on DNA based computers, University of Pennsylvania, Philadelphia, USA, June 15-19, 1998*, *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).