The handle http://hdl.handle.net/1887/37052 holds various files of this Leiden University dissertation.

Author: Vliet, Rudy van
Title: DNA expressions : a formal notation for DNA
Issue Date: 2015-12-10
Part II

Minimal DNA Expressions
Chapter 6

The Length of a DNA Expression

The complexity of an algorithm is often expressed as a function of the length of its input (see Section 2.1). 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 to know how long a DNA expression denoting a certain formal DNA molecule may be. Therefore, in this and later chapters, we examine the length of a DNA expression.

We concentrate on lower bounds for the length of a DNA expression denoting a given (expressible) formal DNA molecule. Obviously, there does not exist an upper bound on the length of such a DNA expression. Indeed, consider an arbitrary DNA expression \( E = \langle |_0 \varepsilon_1 \ldots \varepsilon_n \rangle \), where \( |_0 \) is an operator, \( n \geq 1 \) and \( \varepsilon_1, \ldots, \varepsilon_n \) are the arguments of \( E \). Then \( E' = \langle |_0 E \rangle = \langle |_0 \langle |_0 \varepsilon_1 \ldots \varepsilon_n \rangle \rangle \) is an equivalent DNA expression, for which \( |E'| = |E| + 3 \). This way, we can find arbitrarily long, equivalent DNA expressions.

In Section 6.1, we relate the length of a DNA expression to the number of operators occurring in it. After that, in Section 6.2, we focus on the semantics of a DNA expression, the formal DNA molecule. In particular, we identify specific (blocks of) components in a molecule, count these (blocks of) components and analyse the counting functions. We use the results of this to derive lower bounds for the length of a DNA expression with the desired semantics, in Section 6.3.

6.1 The operators in a DNA expression

Let \( X \) be a string over \( A_{\subseteq a} \). We use \( |X|_A \) to denote the number of \( A \)-letters occurring in \( X \). One can easily verify that \( | \cdot |_A \) is a homomorphism from \( A^*_{\subseteq a} \) to the non-negative integers. Obviously, if \( X \) is a nick free formal DNA molecule, then \( |X|_A \) equals the length of \( X \). It is also easy to see that for an arbitrary formal DNA molecule \( X \), \( |X|_A \) equals the length of \( \nu(X) \). One may wonder why we introduce the new notation \( |X|_A \), while we could as well use the notation \( |\nu(X)| \). The reason is that we often know that a certain formal DNA molecule \( X \) is nick free. In that case, it is useless to apply the function \( \nu \) to \( X \), whereas the notation \( |\nu(X)| \) would suggest that we do that.

We make a basic observation.

Lemma 6.1 Let \( E \) be a DNA expression denoting a formal DNA molecule \( X \), and let \( p \) be the number of operators occurring in \( E \). Then

\[ |E| = 3 \cdot p + |X|_A. \]
Note that a DNA expression consists of operators and corresponding brackets on the one hand, and \( \mathcal{N} \)-letters on the other hand. Hence, Lemma 6.1 implies that \(|X|_A\) does not only count the number of \( \mathcal{A} \)-letters occurring in the formal DNA molecule \( X \), but also the number of \( \mathcal{N} \)-letters occurring in any DNA expression \( E \) denoting \( X \).

**Proof:** By induction on \( p \).

- If \( p = 1 \), then \( E \) is \( \langle \uparrow \alpha_1 \rangle \), \( \langle \downarrow \alpha_1 \rangle \) or \( \langle \uparrow \downarrow \alpha_1 \rangle \) for an \( \mathcal{N} \)-word \( \alpha_1 \). The corresponding formal DNA molecule \( X \) is \( \langle \alpha_1 \rangle \), \( \langle \alpha_1 \rangle \) or \( \langle \alpha_1 \rangle \), respectively. In each of the cases,
  \[ |E| = 3 + |\alpha_1| = 3 \cdot p + |X|_A. \]

- 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 an \( \uparrow \)-expression, or a \( \downarrow \)-expression or an \( \uparrow \downarrow \)-expression.
  - If \( E \) is an \( \uparrow \)-expression, hence \( E = \langle \uparrow \varepsilon_1 \ldots \varepsilon_n \rangle \) for some \( n \geq 1 \) and \( \mathcal{N} \)-words and DNA expressions \( \varepsilon_1, \ldots, \varepsilon_n \), then by definition,
    \[ X = S(E) = \nu^+(S^+(\varepsilon_1)) \ y_1 \ \nu^+(S^+(\varepsilon_2)) \ y_2 \ldots y_{n-1} \ \nu^+(S^+(\varepsilon_n)), \]
    where the \( y_i \)'s are \( \triangle \) or \( \lambda \) (see (4.3)). The function \( \nu^+ \) removes all upper nick letters occurring in its argument, but it does not affect the occurring \( \mathcal{A} \)-letters. Obviously, for \( i = 1, \ldots, n - 1 \), \(|y_i|_A = 0\). This implies that
    \[
    |X|_A = |\nu^+(S^+(\varepsilon_1))|_A + |y_1|_A + |\nu^+(S^+(\varepsilon_2))|_A + |y_2|_A + \cdots + |y_{n-1}|_A
    + |\nu^+(S^+(\varepsilon_n))|_A
    = |S^+(\varepsilon_1)|_A + |S^+(\varepsilon_2)|_A + \cdots + |S^+(\varepsilon_n)|_A.
    
    Apart from the outermost operator, all operators in \( E \) occur in the arguments \( \varepsilon_1, \ldots, \varepsilon_n \). For \( i = 1, \ldots, n \), let \( p_i \) be the number of operators occurring in \( \varepsilon_i \). Then
    \[ p_1 + p_2 + \cdots + p_n = p. \]
    If an argument \( \varepsilon_i \) is an \( \mathcal{N} \)-word \( \alpha_i \), then \( S^+(\varepsilon_i) = \langle \alpha_i \rangle \), \( p_i = 0 \) and
    \[ |\varepsilon_i| = |\alpha_i| = 3 \cdot p_i + |S^+(\varepsilon_i)|_A. \]
    If, on the other hand, an argument \( \varepsilon_i \) is a DNA expression, then \( S^+(\varepsilon_i) = S(\varepsilon_i) \), \( 1 \leq p_i \leq p \) and by the induction hypothesis,
    \[ |\varepsilon_i| = 3 \cdot p_i + |S(\varepsilon_i)|_A = 3 \cdot p_i + |S^+(\varepsilon_i)|_A. \]
    When we combine all equations, we obtain
    \[
    |E| = 3 + |\varepsilon_1| + \cdots + |\varepsilon_n|
    = 3 + (3 \cdot p_1 + |S^+(\varepsilon_1)|_A) + \cdots + (3 \cdot p_n + |S^+(\varepsilon_n)|_A)
    = 3 \cdot (p + 1) + |X|_A.
    

If $E$ is a ↓-expression containing $p + 1$ operators, then the proof is analogous.

Finally, if $E$ is an ↑-expression, then $E = \langle \downarrow E_1 \rangle$ for a DNA expression $E_1$ containing $p \geq 1$ operators. Hence,

$$X = S(E) = \kappa(S(E_1)).$$

Because the function $\kappa$ does not change the number of $A$-letters occurring in its argument, we have

$$|X|_A = |\kappa(S(E_1))|_A = |S(E_1)|_A.$$

We can apply the induction hypothesis to $E_1$:

$$|E_1| = 3 \cdot p + |S(E_1)|_A.$$

We then find

$$|E| = 3 + |E_1| = 3 + 3 \cdot p + |S(E_1)|_A = 3 \cdot (p + 1) + |X|_A.$$

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

---

### 6.2 Blocks of components of a formal DNA molecule

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 three simple counting functions of $X$. In this section, we study these counting functions. Two of the functions count certain subsequences (or blocks) of components of $X$. We first introduce these subsequences.

By Lemma 3.7, the components of a formal DNA molecule are double components and non-double components alternately. The non-double components are upper components, lower components, upper nick letters and lower nick letters. In Section 3.3, we categorized these components as single-stranded components and nick letters (see Figure 3.2). We now make a different categorization:

**Definition 6.2** Let $X$ be a formal DNA molecule and let $x'_1 \ldots x'_k$ for some $k \geq 1$ be the decomposition of $X$.

- An ↑-component $x'_i$ of $X$ is an upper component or a lower nick letter occurring in $X$.

- A ↓-component $x'_i$ of $X$ is a lower component or an upper nick letter occurring in $X$.
Figure 6.1: Relations between different types of components. Components can be divided into double components and non-double components, non-double components can in turn be divided into $\uparrow$-components and $\downarrow$-components, et cetera.

Recall that if $X = S(E)$ for a DNA expression $E$, then upper components and lower nick netters 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. Intuitively, one may regard an $\uparrow$-component as a component that ‘breaks’ the lower strand of a molecule. There is, of course, an analogous interpretation of a $\downarrow$-component. The new categorization is depicted in Figure 6.1.

We consider sequences of components of $X$. We make a simple observation, which follows immediately from the definition of the decomposition of a formal DNA molecule, Definition 3.5.

Lemma 6.3 Let $X$ be a formal DNA molecule, let $x'_1 \ldots x'_k$ for some $k \geq 1$ be the decomposition of $X$ and let $X^s$ be a formal DNA submolecule of $X$ which is a subsequence of the components of $X$: $X^s = x'_{a_0} \ldots x'_{a_1}$ with $1 \leq a_0 \leq a_1 \leq k$.

Then the decomposition of $X^s$ is $x'_{a_0} \ldots x'_{a_1}$.

Hence, the components of $X^s$ are simply the components of $X$ that $X^s$ is built up of.

When we ignore the double components, a formal DNA molecule consists of $\uparrow$-components and $\downarrow$-components. A (maximal) series of $\uparrow$-components is succeeded by a (maximal) series of $\downarrow$-components, which in turn is succeeded by a (maximal) series of $\uparrow$-components, and so on. We are interested in these maximal series, which we call primitive $\uparrow$-blocks and primitive $\downarrow$-blocks, respectively. A formal definition of these blocks also includes the double components occurring in the formal DNA molecule:

Definition 6.4 Let $X$ be a formal DNA molecule and let $x'_1 \ldots x'_k$ for some $k \geq 1$ be the decomposition of $X$.

A primitive $\uparrow$-block of $X$ is an occurrence $(Y_1, Y_2)$ of a non-empty substring $X_1$ of $X$ such that $Y_1 = x'_1 \ldots x'_{a_0-1}$ and $Y_2 = x'_{a_0+1} \ldots x'_k$ for some $a_0$ and $a_1$ with $1 \leq a_0 \leq a_1 \leq k$ (hence $X_1 = x'_{a_0} \ldots x'_{a_1}$), and

- $X_1$ contains at least one non-double component,
6.2 Blocks of components of a formal DNA molecule

![Diagram of DNA blocks]

**Figure 6.2:** Primitive ↑-blocks and primitive ↓-blocks. (a) An example formal DNA molecule \( X \) that contains (upper) nick letters. (b) The primitive ↑-blocks of \( X \). Note that the upper nick letters are not part of these blocks. (c) The primitive ↓-blocks of \( X \).

- each non-double component of \( X_1 \) is an ↑-component,
- either \( a_0 = 1 \) (hence \( Y_1 \) is empty),
  - or \( a_0 \geq 2 \) and \( x_{a_0-1}' \) is a ↓-component,
  and
- either \( a_1 = k \) (hence \( Y_2 \) is empty),
  - or \( a_1 \leq k - 1 \) and \( x'_{a_1+1} \) is a ↓-component.

Note that a primitive ↑-block starts with the double component preceding the series of ↑-components (if such a double component exists) and it ends with the double component succeeding the series of ↑-components (again, if such a double component exists).

A primitive ↑-block of a formal DNA molecule \( X \) is formally defined as an occurrence \((Y_1, Y_2)\) of a substring \( X_1 \) of \( X \) satisfying certain conditions. However, when the occurrence is clear from the context, we will often refer to a primitive ↑-block by the substring \( X_1 \) itself.

The definition of a primitive ↓-block is completely analogous to that of a primitive ↑-block. We may use the term primitive block to refer to either a primitive ↑-block, or a primitive ↓-block.

In Figure 6.2 we have indicated the primitive ↑-blocks and the primitive ↓-blocks of a certain formal DNA molecule containing upper nick letters.

Our first result on primitive blocks, dealing with certain simple types of formal DNA molecules, follows immediately from the definition.

**Lemma 6.5** Let \( X \) be a formal DNA molecule.

1. If \( X = \left( \frac{\alpha_1}{c(\alpha_1)} \right) \) for an \( N \)-word \( \alpha_1 \), then \( X \) does not have any primitive block.
2. If \( X \) has at least one ↑-component, but does not have any ↓-component, then \( X \) is a primitive ↑-block of itself and \( X \) does not have any primitive ↓-block.
3. If \( X \) has at least one ↓-component, but does not have any ↑-component, then \( X \) is a primitive ↓-block of itself and \( X \) does not have any primitive ↑-block.
Formal DNA molecules of the form $\left(\frac{\alpha_1}{c(\alpha_1)}\right)$ for an $N$-word $\alpha_1$ will come back frequently in the remainder of this chapter and in later chapters. Often, we are not interested in the actual $N$-letters occurring in such a molecule (hence in $\alpha_1$), but only in the shape of the molecule, for example, 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 $\left(\frac{\alpha_1}{c(\alpha_1)}\right)$ for an $N$-word $\alpha_1$.

We already mentioned that a primitive ↑-block of a formal DNA molecule $X$ starts with the double component preceding a series of ↑-components, and ends with the double component succeeding this series (if these double components exist). We formalize and extend this observation for a primitive ↑-block which is not equal to $X$.

**Lemma 6.6** Let $X$ be a formal DNA molecule, let $x'_1 \ldots x'_k$ for some $k \geq 1$ be the decomposition of $X$, and let $X_1 = x'_{a_0} \ldots x'_{a_1}$ for some $a_0$ and $a_1$ with $1 \leq a_0 \leq a_1 \leq k$ be a primitive ↑-block of $X$.

1. If $a_0 \geq 2$, then $a_0 < a_1 \leq k$, $x'_{a_0-1}$ is a ↓-component, $x'_{a_0}$ is a double component and $x'_{a_0+1}$ is an ↑-component of $X$.

2. If $a_1 \leq k - 1$, then $1 \leq a_0 < a_1$, $x'_{a_1+1}$ is a ↓-component, $x'_{a_1}$ is a double component and $x'_{a_1-1}$ is an ↑-component of $X$.

**Proof:**

1. Assume that $a_0 \geq 2$. Then, by definition, $x'_{a_0-1}$ is a ↓-component, and by Lemma 3.6, $x'_{a_0}$ is a double component. Because $X_1$ contains at least one ↑-component, we must have $a_0 < a_1$. Hence, $x'_{a_0+1}$ is part of $X_1$, and by the definition of a primitive ↑-block, it is a double component or an ↑-component. By Lemma 3.7, it has to be an ↑-component.

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

For the formal DNA molecule from Figure 6.2, the following result is clear from the picture. We will prove it in general.

**Lemma 6.7** Let $X$ be a formal DNA molecule and let $x'_1 \ldots x'_k$ for some $k \geq 1$ be the decomposition of $X$.

1. The primitive ↑-blocks of $X$ are pairwise disjoint.

2. Each ↑-component of $X$ occurs in a (exactly one) primitive ↑-block.

3. Let $X_1 = x'_{a_0} \ldots x'_{a_1}$ and $X_2 = x'_{a_2} \ldots x'_{a_3}$ with $1 \leq a_0 \leq a_1 < a_2 \leq a_3 \leq k$ be two consecutive primitive ↑-blocks of $X$. Then $x'_{a_1}$ and $x'_{a_2}$ are double components of $X$, $a_2 - a_1 \geq 2$ and $x'_{a_1} \ldots x'_{a_2}$ is a primitive ↓-block of $X$.

4. Let $X_1 = x'_{a_0} \ldots x'_{a_1}$ with $1 \leq a_0 \leq a_1 \leq k$ be the first primitive ↑-block of $X$. If $a_0 \geq 2$, then $x'_{a_0}$ is a double component and $x'_{a_1} \ldots x'_{a_0}$ is a primitive ↓-block of $X$. 
5. Let $X_1 = x_{a_0} \ldots x_{a_1}$ with $1 \leq a_0 \leq a_1 \leq k$ be the last primitive \(\uparrow\)-block of \(X\). If $a_1 \leq k - 1$, then $x_{a_1}'$ is a double component and $x_{a_1}' \ldots x_{a_k}'$ is a primitive \(\downarrow\)-block of \(X\).

6. Let $X_1 = x_{a_0} \ldots x_{a_1}$ with $1 \leq a_0 \leq a_1 \leq k$ be a primitive \(\uparrow\)-block of \(X\) and let $X_2 = x_{a_2} \ldots x_{a_3}$ with $1 \leq a_2 \leq a_3 \leq k$ be a primitive \(\downarrow\)-block of \(X\). If $X_1$ and $X_2$ have a non-empty intersection, then

- either $1 \leq a_0 < a_1 = a_2 < a_3 \leq k$ and $x_{a_1}' (= x_{a_2}')$ is a double component,
- or $1 \leq a_2 < a_3 = a_0 < a_1 \leq k$ and $x_{a_1}' (= x_{a_0}')$ is a double component.

Because of the analogous definitions of primitive \(\uparrow\)-blocks and primitive \(\downarrow\)-blocks, we have analogous results for primitive \(\downarrow\)-blocks.

By Claim 1, we can unambiguously order the different primitive \(\uparrow\)-blocks of a formal DNA molecule \(X\), according to their occurrence in \(X\). We can speak of the first, the second, \ldots, the last primitive \(\uparrow\)-block of a formal DNA molecule. Hence, Claims 3 and 5 make sense.

Claim 2 allows us to speak of the primitive \(\uparrow\)-block of a \(\uparrow\)-component, being the (unique) primitive \(\uparrow\)-block that the \(\uparrow\)-component is part of. Analogously, we have the primitive \(\downarrow\)-block of a \(\downarrow\)-component.

Finally, by Claim 3, if a primitive \(\uparrow\)-block and a primitive \(\downarrow\)-block of \(X\) have a non-empty intersection, then they overlap and the intersection consists only of the double component that one of the blocks ends with and the other one starts with. Therefore, it should not be confusing to say, e.g., that a primitive \(\uparrow\)-block is succeeded by a primitive \(\downarrow\)-block. For example, the primitive \(\uparrow\)-block $X_2$ of the formal DNA molecule from Figure 6.2 is succeeded by the primitive \(\downarrow\)-block $X_3'$, which in turn is succeeded by the primitive \(\uparrow\)-block $X_3$.

**Proof of Lemma 6.7:**

1. Let $X_1 = x_{a_0} \ldots x_{a_1}$ with $1 \leq a_0 \leq a_1 \leq k$ and $X_2 = x_{a_2} \ldots x_{a_3}$ with $1 \leq a_2 \leq a_3 \leq k$ be two different primitive \(\uparrow\)-blocks of \(X\). Without loss of generality, assume that $a_0 \neq a_2$ (otherwise consider $a_1$ and $a_3$ and mirror the arguments). In particular, assume that $a_0 < a_2$.

Because $1 \leq a_0 < a_2$, we have $a_2 \geq 2$ and thus, by definition, $x_{a_2-1}'$ is a \(\downarrow\)-component. Further, because $a_0 \leq a_2 - 1$, we must have $a_1 < a_2 - 1$, because otherwise (the primitive \(\uparrow\)-block) $X_1$ would contain the \(\downarrow\)-component $x_{a_2-1}'$.

Consequently, $a_0 \leq a_1 < a_2 - 1 < a_2 \leq a_3$, and $X_1$ and $X_2$ are disjoint.

2. Let $x_{i_0}'$ be an arbitrary \(\uparrow\)-component of \(X\). By Lemma 3.7 each component $x_i'$ with $i \equiv i_0 \pmod{2}$ is an \(\uparrow\)-component or a \(\downarrow\)-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 index with $1 \leq b_0 \leq i_0$ and $b_0 \equiv i_0 \pmod{2}$ such that each of $x_{i_0}'$, $x_{i_0+2}'$, $x_{i_0+4}'$, $x_{i_0+6}'$ is an \(\uparrow\)-component. Further, let $b_1$ be the largest index with $i_0 \leq b_1 \leq k$ and $b_1 \equiv i_0 \pmod{2}$ such that each of $x_{i_0}'$, $x_{i_0+2}'$, $x_{i_0+4}'$, $x_{i_0+6}'$ is an \(\uparrow\)-component. Because $x_{i_0}'$ itself is an \(\uparrow\)-component, $b_0$ and $b_1$ are well defined.

We subsequently define indices $a_0$ and $a_1$ by

\[
  a_0 = \begin{cases} 
    1 & \text{if } b_0 \leq 2 \\
    b_0 - 1 & \text{if } b_0 \geq 3
  \end{cases} \quad \text{and} \quad a_1 = \begin{cases} 
    k & \text{if } b_1 \geq k - 1 \\
    b_1 + 1 & \text{if } b_1 \leq k - 2
  \end{cases}
\]
and let $X_1 = x'_{a_0} \ldots x'_{a_1}$. It is easy to see that $1 \leq a_0 \leq b_0 \leq i_0 \leq b_1 \leq a_1 \leq k$. Indeed, $X_1$ contains $x'_{i_0}$.

By the definition of $b_0$ and $b_1$ and by the fact that $b_0 - 1 \leq a_0 \leq b_0$ and $b_1 \leq a_1 \leq b_1 + 1$, each of the components $x'_{a_0}, \ldots, x'_{a_1}$ is either an ↑-component, or a double component. Further, either $a_0 = 1$, or $a_0 \geq 2$. In the latter case, $a_0 = b_0 - 1$, and hence $x'_{a_0 - 1} = x'_{i_0 - 2}$ is a ↓-component. Similarly, either $a_1 = k$, or $a_1 \leq k - 1$. In the latter case, $a_1 = b_1 + 1$ and $x'_{a_1 + 1} = x'_{b_1 + 2}$ is a ↓-component. Consequently, $X_1$ is a primitive ↑-block.

Because, by Claim [1], primitive ↑-blocks are pairwise disjoint, the ↑-component $x'_{i_0}$ does not occur in any other primitive ↑-block of $X$.

3. In the proof of Claim [1], we have established that $a_1 < a_2 - 1 < a_2$, hence that $a_2 - a_1 \geq 2$. By Lemma 6.6, $1 \leq a_0 < a_1$, $a_2 < a_3 \leq k$, both $x'_{a_1 + 1}$ and $x'_{a_2 - 1}$ are ↓-components, both $x'_{a_1}$ and $x'_{a_2}$ are double components and both $x'_{a_1 - 1}$ and $x'_{a_2 + 1}$ are ↑-components.

By Claim [2], each ↑-component occurs in a primitive ↑-block of $X$. Now, the existence of an ↑-component between $X_1$ and $X_2$ would contradict the assumption that they are consecutive. Hence, the sequence $x'_{a_1} \ldots x'_{a_2}$ only consists of double components and ↓-components, and at least one of them is a ↓-component.

This implies that $x'_{a_1} \ldots x'_{a_2}$ satisfies all conditions of a primitive ↓-block.

4. The proof of this claim is similar to that of the previous claim.

We first establish that if $a_0 \geq 2$, then $x'_{a_0 - 1}$ is a ↓-component, $x'_{a_0}$ is a double component and $x'_{a_0 + 1}$ is an ↑-component. We subsequently prove that $x'_{1} \ldots x'_{a_0}$ does not contain any ↑-component, because otherwise we could find a primitive ↑-block before the first primitive ↑-block $X_1$. These properties together imply that $x'_{1} \ldots x'_{a_0}$ is a primitive ↓-block.

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

6. Assume that $X_1$ and $X_2$ have a non-empty intersection. Because both $X_1$ and $X_2$ are built up of (complete) components of $X$, so is their intersection. Let $x'_{i_0}$ be a component of $X$ that occurs both in $X_1$ and $X_2$. Then $a_0 \leq i_0 \leq a_1$ and $a_2 \leq i_0 \leq a_3$.

By the definition of (the primitive ↑-block) $X_1$, $x'_{i_0}$ is not a ↓-component, and by the definition of (the primitive ↓-block) $X_2$, $x'_{i_0}$ is not an ↑-component. Hence, it is a double component. Because $X_1$ contains at least one ↑-component, either $a_0 < i_0$, or $i_0 < a_1$ (or both).

Assume that $a_0 < i_0$. Then by Lemma 6.1, $x'_{i_0 - 1}$ is an ↑-component. This component cannot be part of (the primitive ↓-block) $X_2$. Because $x'_{i_0}$ is part of $X_2$, we must have $a_2 = i_0$. Now, because $X_2$ contains at least one ↓-component, $i_0 < a_3$ and $x'_{i_0 + 1}$ is a ↓-component. This component, in turn cannot be part of $X_1$, which implies that $a_1 = i_0$. In this case, we have the first subclaim.

If, on the other hand, we assume that $i_0 < a_1$, then we obtain the second subclaim.
Corollary 6.8 A formal DNA molecule does not have any primitive block, if and only if it is double-complete.

We now consider formal DNA molecules which are not double-complete.

Lemma 6.9 Let \( X \) be a formal DNA molecule which is not double-complete.

1. \( X \) can be considered as an alternating sequence of (all its) primitive \( \uparrow \)-blocks and (all its) primitive \( \downarrow \)-blocks. Any two consecutive primitive blocks in this sequence share (only) a double component of \( X \).

2. (a) The first non-double component of \( X \) is an \( \uparrow \)-component, if and only if the alternating sequence from Claim 1 starts with a primitive \( \uparrow \)-block.

(b) The last non-double component of \( X \) is an \( \uparrow \)-component, if and only if the alternating sequence from Claim 1 ends with a primitive \( \uparrow \)-block.

Note that in Claim 1 we say “\( X \) can be considered as an alternating sequence . . .,” rather than “\( X \) is an alternating sequence . . .” The reason for this is that consecutive primitive blocks in this sequence (e.g., a primitive \( \uparrow \)-block and the primitive \( \downarrow \)-block succeeding it) are not disjoint. As we say in the second part of the claim, they share a double component. In the alternating sequence, we must, of course, include this double component only once.

It is easily verified that all claims are valid for the formal DNA molecule depicted in Figure 6.2. For this molecule, the alternating sequence is \( X_0, X_1, X'_1, X_2, X'_2, X_3, X'_3 \).

Proof:

1. Without loss of generality, assume that \( X \) contains at least one \( \uparrow \)-component, and let \( X_1, \ldots, X_r \) for some \( r \geq 0 \) be all primitive \( \uparrow \)-blocks of \( X \), in the order of their occurrence in \( X \). By Lemma 6.7[2], \( r \geq 1 \). By Lemma 6.7[1], the primitive \( \uparrow \)-blocks of \( X \) are pairwise disjoint. Now, consider a primitive \( \uparrow \)-block \( X_j \) of \( X \) with \( 1 \leq j \leq r - 1 \) (if this exists). By Lemma 6.7[3], there exists a primitive \( \downarrow \)-block \( X'_j \) of \( X \) that starts with the last (double) component of \( X_j \) and ends with the first (double) component of \( X_{j+1} \).

Assume that neither \( X_1 \) is a prefix of \( X \), nor \( X_r \) is a suffix of \( X \). Then by Lemma 6.7[1] and [5], both the prefix \( X'_0 \) of \( X \) that ends with the first (double) component of \( X_1 \) and the suffix \( X'_r \) of \( X \) that starts with the last (double) component of \( X_r \) are primitive \( \downarrow \)-blocks of \( X \). We then have the alternating sequence \( X'_0, X_1, X'_1, \ldots, X_r, X'_r \) of all primitive \( \uparrow \)-blocks \( X_1, \ldots, X_r \) and primitive \( \downarrow \)-blocks \( X'_0, X'_1, \ldots, X'_r \) which completely cover \( X \). In fact, when we include each double component which is shared by a primitive \( \uparrow \)-block and a primitive \( \downarrow \)-block only once, this sequence is equal to \( X \).

We still have to prove that the alternating sequence includes all primitive \( \downarrow \)-blocks of \( X \). Suppose that there is a primitive \( \downarrow \)-block \( X'_{00} \) of \( X \) different from the ones occurring in the sequence. By (the analogue for primitive \( \downarrow \)-blocks of) Lemma 6.7[1], it would have an empty intersection with each of \( X'_1, X'_1, \ldots, X'_r \). Then, because \( X'_{00} \) is a substring of \( X \), it should be contained in a primitive \( \uparrow \)-block \( X_j \) for some \( j \) with \( 1 \leq j \leq r \). By definition, the primitive \( \downarrow \)-block \( X'_{00} \) contains at least one \( \downarrow \)-component. This \( \downarrow \)-component would also be part of the primitive \( \uparrow \)-block \( X_j \), which is impossible.

We conclude that in this case, the claim holds. The proofs for the cases that either \( X_1 \) is a prefix of \( X \), or \( X_r \) is a suffix of \( X \) (or both) are analogous.
2. (a) Assume that the alternating sequence from Claim 1 starts with a primitive ↑-block \( X_0 \) of \( X \). By definition, a primitive ↑-block contains at least one ↑-component and does not contain any ↓-component. Hence, the first non-double component of \( X \) is the first non-double component of its prefix \( X_0 \), which is an ↑-component.

Conversely, if the first non-double component of \( X \) is an ↑-component, then the alternating sequence cannot start with a primitive ↓-block. Hence, it must start with a primitive ↑-block.

(b) The proof of this subclaim is analogous to that of the previous subclaim.

We now define functions that count the primitive ↑-blocks, the primitive ↓-blocks and the double components occurring in a formal DNA molecule \( X \).

**Definition 6.10** Let \( X \) be a formal DNA molecule.

- \( B_↑(X) \) is the number of primitive ↑-blocks of \( X \).
- \( B_↓(X) \) is the number of primitive ↓-blocks of \( X \).
- \( n_↕(X) \) is the number of double components of \( X \).

For the formal DNA molecule \( X \) from Figure 6.2 we have \( B_↑(X) = 3 \), \( B_↓(X) = 4 \) and \( n_↕(X) = 10 \).

We are interested in the values of the functions \( B_↑ \), \( B_↓ \) and \( n_↕ \) 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

\[
B_↑(\lambda) = B_↓(\lambda) = n_↕(\lambda) = 0.
\]

In addition to the three new counting functions, we will use \( \#_\triangledown(X) \), \( \#_\triangle(X) \), \( \#_\triangledown\triangle(X) \), \( \#_↑(E) \), \( \#_↓(E) \), \( \#↑↓(E) \) and \( \#↓↑(E) \). 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 3.7:

**Lemma 6.11** Let \( X \) be a formal DNA molecule and let \( x'_1 \ldots x'_k \) for some \( k \geq 1 \) be the decomposition of \( X \).

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

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

**Lemma 6.12** Let \( X \) be a formal DNA molecule.

1. \( B_↑(X) \leq B_↓(X) + 1 \).
2. \( B_↓(X) \leq B_↑(X) + 1 \).
3. If \( X \) does not contain any single-stranded component, then \( n_↕(X) = \#_\triangledown\triangle(X) + 1 \).
4. If \( X \) contains at least one nick letter, then \( n_↕(X) \geq \#_\triangledown\triangle(X) + 1 \).
6.2 Blocks of components of a formal DNA molecule

5. $n_4^\downarrow(X) \geq \#_{\nabla,\triangle}(X)$.

Proof:

1, 2. If $X$ is double-complete, then by Lemma 6.5(1), $X$ does not have any primitive block. Hence, Claims 1 and 2 are trivially valid.

If $X$ is not double-complete, then by Lemma 6.9(1), the primitive $\uparrow$-blocks and primitive $\downarrow$-blocks occur in $X$ alternately. Hence, the difference between their numbers of occurrences can be at most 1. Again, Claims 1 and 2 follow immediately.

3. This claim follows immediately from Corollary 3.9(2).

4. Let $x'_1 \ldots x'_k$ 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, $i \geq 2$ and $x'_{i-1}$ is a double component. Further, if $x'_i$ is the last nick letter occurring in $X$, then $i_0 \leq k - 1$ and also $x'_{i_0+1}$ is a double component. Obviously, all these double components are different, and thus $n_4^\downarrow(X) \geq \#_{\nabla,\triangle}(X) + 1$.

5. If $X$ is nick free (hence $\#_{\nabla,\triangle}(X) = 0$), then the claim holds because $n_4^\downarrow(X) \geq 0$. If $X$ is not nick free, then the claim follows from Claim 4.

In the next result, we consider nick free formal DNA molecules. In such formal DNA molecules, each non-double component is a single-stranded component (see Figure 3.2), each $\uparrow$-component is an upper component and each $\downarrow$-component is a lower component (see Figure 6.1). In fact, all claims below can directly be generalized to formal DNA molecules that may contain nick letters. However, because we will use the claims only in the nick free case, we give the nick free formulation.

Lemma 6.13 Let $X$ be a nick free formal DNA molecule.

1. (a) $B_1^\uparrow(X) = 0$ if and only if $X$ does not contain any upper component.
    
    (b) $B_1^\downarrow(X) = 0$ if and only if $X$ does not contain any lower component.

2. $X$ is not double-complete, if and only if $X$ contains at least one single-stranded component.

3. Assume that $X$ is not double-complete.

   (a) If both the first single-stranded component and the last single-stranded component of $X$ are upper components, then $B_1^\uparrow(X) = B_1^\downarrow(X) + 1$.

   (b) 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 $B_1^\uparrow(X) = B_1^\downarrow(X)$.

   (c) 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 $B_1^\uparrow(X) = B_1^\downarrow(X)$.

   (d) If both the first single-stranded component and the last single-stranded component of $X$ are lower components, then $B_1^\uparrow(X) = B_1^\downarrow(X) - 1$.

\qed
Proof:

1. (a) By the definition of a primitive $\uparrow$-block, if $X$ does not have any upper component, then $B_\uparrow(X) = 0$. Conversely, by Lemma 6.7(2), if $B_\uparrow(X) = 0$, then $X$ cannot have an upper component.

(b) The proof of this subclaim is analogous to that of the previous subclaim.

2. This claim is immediate from Corollary 3.8(1).

3. All four subclaims follow immediately from Lemma 6.9.

When a formal DNA molecule is denoted by a DNA expression, the values of the functions $B_\uparrow$, $B_\downarrow$ and $n_\updownarrow$ for the full molecule can be related to the values for its constituents. In order to establish these relations, we will systematically analyze the effects of the operators $\uparrow$, $\downarrow$ and $\updownarrow$ on the values of these counting functions.

Some of the relations we obtain, are not directly useful for the ultimate goal of this chapter, the determination of lower bounds on the length of a DNA expression. For the sake of completeness, we give these relations anyway.

In the definition of the semantics of an $\uparrow$-expression, a $\downarrow$-expression and an $\updownarrow$-expression, there is an important role for the functions $\nu^+$, $\nu^-$ and $\kappa$, respectively (see Definition 4.1).

We first consider the effects of $\nu^+$ on the values of the counting functions. Of course, the effects of $\nu^-$ are analogous. We will subsequently examine the effects of $\kappa$.

The function $\nu^+$ removes all upper nick letters occurring in its argument. The removal of even a single upper nick letter may affect the counting numbers.

**Lemma 6.14** Let $X$ be a formal DNA molecule containing at least one upper nick letter, and let $X'$ be the string that results from $X$ by removing one upper nick letter. Then $X'$ is a formal DNA molecule and

1. $B_\uparrow(X') \leq B_\uparrow(X)$,
2. $B_\uparrow(X') \geq B_\uparrow(X) - 1$,
3. $B_\downarrow(X') \leq B_\downarrow(X)$,
4. $B_\downarrow(X') \geq B_\downarrow(X) - 1$ and
5. $n_\updownarrow(X') = n_\updownarrow(X) - 1$.

**Proof:** Because $X$ is not double-complete, by Lemma 6.9(1), it can be considered as an alternating sequence of all its primitive $\uparrow$-blocks and all its primitive $\downarrow$-blocks.

Let $x'_1 \ldots x'_k$ for some $k \geq 1$ be the decomposition of $X$ and let $x'_i$ with $1 \leq i \leq k$ be the upper nick letter that is removed. Hence, $X' = x'_1 \ldots x'_{i_0-1}x'_{i_0}x'_{i_0+1} \ldots x'_k$.

By the definition of a formal DNA molecule (Definition 3.2), nick letters may occur only between two double components. Consequently, $2 \leq i_0 \leq k - 1$ and both $x'_{i_0-1}$ and $x'_{i_0+1}$ are double components. When $x'_{i_0}$ is removed, the two double components become adjacent and thus form one larger double component $x'_{i_0-1}x'_{i_0+1}$. Indeed, the resulting string $X'$ satisfies all conditions of a formal DNA molecule.

Because two double components merge into one and the other double components are not affected by the removal of $x'_{i_0}$, $n_\updownarrow(X') = n_\updownarrow(X) - 1$, which is Claim 5.
Figure 6.3: The effect on primitive blocks (indicated by the braces) of removing an upper nick letter in three different cases. (a) The upper nick letter removed is the first, but not the only ↓-component of its primitive ↑-block $X_1$. The last (double) component of the preceding primitive ↑-block is replaced by a larger double component. (b) The upper nick letter removed is the only ↓-component of its primitive ↓-block $X_1$. The primitive ↑-block preceding $X_1$ and the one succeeding $X_1$ merge into one. (c) The upper nick letter removed is the only ↓-component of its primitive ↓-block $X_1$. $X_1$ is not preceded by a primitive ↑-block. The first (double) component of the primitive ↑-block succeeding $X_1$ is replaced by a larger double component.

The upper nick letter $x'_{i_0}$ is a ↓-component. By Lemma 6.7(2), there is (exactly) one primitive ↓-block $X_1$ of $X$ containing $x'_{i_0}$, say $X_1 = x'_{a_1} \ldots x'_{i_0-1} x'_{i_0} x'_{i_0+1} \ldots x'_{a_2}$ for some $a_1$ and $a_2$ with $1 \leq a_1 \leq i_0 \leq a_2 \leq k$. Because $2 \leq i_0 \leq k-1$ and both $x'_{i_0-1}$ and $x'_{i_0+1}$ are double components, it follows from the definition of a primitive ↓-block that $a_1 \leq i_0 - 1$ and $i_0 + 1 \leq a_2$.

Now, there are two possibilities: either $X_1$ contains ↓-components other than $x'_{i_0}$, or it does not.

- In the former case, either $a_1 < i_0 - 1$ or $i_0 + 1 < a_2$ (or both). When $x'_{i_0}$ is removed from $X_1$, the result $X'_1 = x'_{a_1} \ldots x'_{i_0-1} x'_{i_0+1} \ldots x'_{a_2}$ still contains at least one ↓-component. It is easily verified that it also satisfies the other conditions of a primitive ↓-block of $X'$. Hence, the primitive ↓-block $X_1$ has been replaced by the primitive ↓-block $X'_1$.

If $x'_{i_0}$ was the first ↓-component of $X_1$ (hence, $a_1 = i_0 - 1$) and $X_1$ was preceded in $X$ by a primitive ↑-block, then the last component of this primitive ↑-block was the double component $x'_{i_0-1}$ and has now become the double component $x'_{i_0-1} x'_{i_0+1}$. This effect is depicted in Figure 6.3(a). Analogously, the first component of the primitive ↑-block succeeding $X_1$ may have changed. None of the other primitive ↑-blocks and primitive ↓-blocks of $X$ is affected by the removal of $x'_{i_0}$. Hence, in this case, $B_\tau(X') = B_\tau(X)$ and $B_1(X') = B_1(X)$.

- Now, consider the case that the upper nick letter $x'_{i_0}$ is the only ↓-component of $X_1$. Then $X_1$ only consists of the double component $x'_{i_0-1}$, the upper nick letter $x'_{i_0}$ and the double component $x'_{i_0+1}$: $a_1 = i_0 - 1$ and $a_2 = i_0 + 1$. When we remove $x'_{i_0}$, we
Figure 6.4: The result of applying $\nu^+$ to the formal DNA molecule $X$ from Figure 6.2. The primitive $\uparrow$-blocks and primitive $\downarrow$-blocks of $\nu^+(X)$ are also indicated.

lose the primitive $\downarrow$-block $X_1$. By Lemma 6.7(3), the other primitive $\downarrow$-blocks of $X$ (if any) are separated from $X_1$ by at least one $\uparrow$-component. They are not affected by the removal of $x'_{i_0}$, hence $B_{\uparrow}(X') = B_{\downarrow}(X) - 1$.

For the effect on $B_{\uparrow}(X)$, we distinguish a number of subcases. If $X_1$ is both preceded and succeeded in $X$ by a primitive $\uparrow$-block (hence $a_1 \geq 2$ and $a_2 \leq k - 1$), then these two blocks merge into one when the upper nick letter $x'_{i_0}$ separating them is removed (see Figure 6.3(b)). The other primitive $\uparrow$-blocks of $X$ (if any) are not affected. In this case, $B_{\uparrow}(X') = B_{\uparrow}(X) - 1$.

If, on the other hand, $X_1$ is the first primitive block of $X$ and is succeeded in $X$ by a primitive $\uparrow$-block ($a_1 = 1$ and $a_2 \leq k - 1$), then the effect of the removal of $x'_{i_0}$ is smaller: the first component of the succeeding primitive $\uparrow$-block, which used to be the double component $x'_{a_2}$, is replaced by the double component $x'_{a_1}x'_{a_2}$ (see Figure 6.3(c)). The number of primitive $\uparrow$-blocks remains the same: $B_{\uparrow}(X') = B_{\uparrow}(X)$. This is also the case if $X_1$ is preceded but not succeeded in $X$ by a primitive $\uparrow$-block ($a_1 \geq 2$ and $a_2 = k$) and if $X_1$ is the only primitive block of $X$ ($a_1 = 1$ and $a_2 = k$). Note that in the last case, $k = 3$, $B_{\uparrow}(X') = B_{\downarrow}(X) = 0$, $B_{\downarrow}(X) = 1$, $B_{\uparrow}(X') = 0$ and the resulting formal DNA molecule $X'$ consists only of the double component $x'_{1}x'_{3}$.

In all cases we considered, $B_{\uparrow}(X')$ and $B_{\downarrow}(X')$ satisfy the inequalities in Claims 1–4.

We now examine the effects of removing all upper nick letters, by means of the function $\nu^+$. We first consider an example.

Example 6.15 In Figure 6.4, we have depicted $\nu^+(X)$ for the formal DNA molecule $X$ from Figure 6.2. As we have established before,

$$B_{\uparrow}(X) = 3, \quad B_{\downarrow}(X) = 4 \quad \text{and} \quad n_{\uparrow}(X) = 10.$$  

After the removal of the $\#_{\nu}(X) = 4$ upper nick letters, the numbers are

$$B_{\uparrow}(\nu^+(X)) = 2, \quad B_{\downarrow}(\nu^+(X)) = 2 \quad \text{and} \quad n_{\uparrow}(\nu^+(X)) = 6.$$  

We use Lemma 6.16 to determine (upper bounds and lower bounds on) the effects of the function $\nu^+$ on the counting numbers for arbitrary formal DNA molecules.

Lemma 6.16 Let $X$ be a formal DNA molecule. Then $\nu^+(X)$ is a formal DNA molecule and

1. $B_{\uparrow}(\nu^+(X)) \leq B_{\uparrow}(X),$

2. $B_{\downarrow}(\nu^+(X)) \leq B_{\downarrow}(X),$

3. $B_{\uparrow}(X) \leq B_{\downarrow}(X) + 1,$

4. $B_{\downarrow}(X) \leq B_{\uparrow}(X) + 1.$
2. \( B_1(\nu^+(X)) \geq B_1(X) - \#_\varpi(X) \),
3. \( B_1(\nu^+(X)) \leq B_1(X) \),
4. \( B_1(\nu^+(X)) \geq B_1(X) - \#_\varpi(X) \) and 
5. \( n_2(\nu^+(X)) = n_2(X) - \#_\varpi(X) \).

**Proof:** Already at the definition of the function \( \nu^+ \), we observed that for each formal DNA molecule \( X \), also \( \nu^+(X) \) is a formal DNA molecule.

When we apply \( \nu^+ \) to \( X \), we remove all upper nick letters occurring in \( X \) at once. We may, however, also remove the upper nick letters one by one. In whatever order we remove the upper nick letters, the final result is the same: \( \nu^+(X) \). At each of the \( \#_\varpi(X) \) removals, we can apply Lemma 6.14. Now, each of the claims follows from the repeated application of the corresponding claim from Lemma 6.14. 

By the above result (and the analogue for \( \nu^-(X) \)), we can derive invariants for the functions \( \nu^+ \), \( \nu^- \) and \( \nu \): the function \( n_1(\cdot) - \#_\varpi(\cdot) \) is an invariant for the function \( \nu^+ \), the function \( n_1(\cdot) - \#_{\triangle}(\cdot) \) is an invariant for the function \( \nu^- \), and the function \( n_1(\cdot) - \#_{\varpi,\triangle}(\cdot) \) is an invariant for the function \( \nu \):

**Corollary 6.17** Let \( X \) be a formal DNA molecule.

1. \( n_1(\nu^+(X)) - \#_\varpi(\nu^+(X)) = n_1(X) - \#_\varpi(X) \)
2. \( n_1(\nu^-(X)) - \#_{\triangle}(\nu^-(X)) = n_1(X) - \#_{\triangle}(X) \)
3. \( n_1(\nu(X)) - \#_{\varpi,\triangle}(\nu(X)) = n_1(X) - \#_{\varpi,\triangle}(X) \)

**Proof:** By the definition of the function \( \nu^+ \), \( \#_\varpi(\nu^+(X)) = 0 \) and \( \#_{\triangle}(\nu^+(X)) = \#_{\triangle}(X) \). Now, Claim 1 follows immediately from Lemma 6.16. Then by analogy, we also have Claim 2. Finally, Claim 3 follows from the other two, because the function \( \nu \) is the composition of \( \nu^+ \) and \( \nu^- \):

\[
\begin{align*}
n_1(\nu(X)) - \#_{\varpi,\triangle}(\nu(X)) &= n_1(\nu^+(\nu^-(X))) - \#_\varpi(\nu^+(\nu^-(X))) - \#_{\triangle}(\nu^+(\nu^-(X))) \\
&= n_1(\nu^-(X)) - \#_\varpi(\nu^-(X)) - \#_{\triangle}(\nu^-(X)) \\
&= n_1(X) - \#_\varpi(X) - \#_{\triangle}(X) \\
&= n_1(X) - \#_{\varpi,\triangle}(X).
\end{align*}
\]

Instead of \( \nu^+ \) or \( \nu^- \), which remove nick letters from their arguments, we may apply \( \kappa \) to a formal DNA molecule. This function complements all single-stranded components of its argument, i.e., it substitutes them by the corresponding double \( \mathcal{A} \)-words. In some sense, we can also regard this as ‘removing’ non-double components.

We will formulate inequalities for the values of the three counting functions after the application of \( \kappa \). As we did with the removal of nick letters, we first examine the effects of the complementation of one single-stranded component, in particular of a lower component. For the complementation of an upper component, we have, of course, analogous (in fact: equal) inequalities.

\[\footnote{We do not say double components, because in general the corresponding double \( \mathcal{A} \)-words are not components of the resulting formal DNA molecule.}\]
Lemma 6.18 Let $X$ be a formal DNA molecule containing at least one lower component, and let $X'$ be the string that results from $X$ by complementing one lower component of $X$. Then $X'$ is a formal DNA molecule and

1. $B_\uparrow(X') \leq B_\uparrow(X)$,
2. $B_\uparrow(X') \geq B_\uparrow(X) - 1$,
3. $B_\downarrow(X') \leq B_\downarrow(X)$,
4. $B_\downarrow(X') \geq B_\downarrow(X) - 1$.
5. $n_\uparrow(X') \leq n_\uparrow(X) + 1$ and
6. $n_\uparrow(X') \geq n_\uparrow(X) - 1$.

Proof: Let $(\vec{c}(\alpha_1))$ for an $\mathcal{N}$-word $\alpha_1$ be the lower component that is complemented.

The main difference between Claims 1–4 of this result and Lemma 6.14(1)–(4) is that the ↓-component $(\vec{c}(\alpha_1))$ which is ‘removed’ here is not necessarily preceded and succeeded in $X$ by a double component. It may also be the first and/or the last component of $X$. This does, however, not change the structure of the proof. We distinguish the same cases and in each of the cases we have the same effect on $B_\uparrow(X)$ and $B_\downarrow(X)$ as in the proof of Lemma 6.14. For example, let $X_1$ be the primitive ↓-block of $(\vec{c}(\alpha_1))$. If the lower component $(\vec{c}(\alpha_1))$ is preceded in $X$ by a double component $(\vec{c}(\alpha_0))$ and succeeded in $X$ by a double component $(\vec{c}(\alpha_0\alpha_1\alpha_2))$, then we obtain one double component $(\vec{c}(\alpha_0\alpha_1\alpha_2))$ when we replace $(\vec{c}(\alpha_1))$ by the corresponding double $\mathcal{A}$-word. This may or may not affect $B_\uparrow(X)$ and $B_\downarrow(X)$, depending on whether or not $X_1$ contains other ↓-components, and on whether or not $X_1$ is preceded and/or succeeded in $X$ by a primitive ↑-block.

We turn to Claims 5 and 6 where we consider $n_\uparrow(X')$. If $(\vec{c}(\alpha_1))$ is the only component of $X$, hence if $X = (\vec{c}(\alpha_1))$, then $X' = (\vec{c}(\alpha_1))$, $n_\uparrow(X) = 0$ and $n_\uparrow(X') = 1$.

Now, assume that $(\vec{c}(\alpha_1))$ is not the only component of $X$. If it is the first component, then by Lemma 3.17 it is succeeded in $X$ by a double component. This double component is extended to the left by $(\vec{c}(\alpha_1))$ when we complement our lower component. Because the other double components of $X$ are not affected, the number of double components remains the same: $n_\uparrow(X') = n_\uparrow(X)$. Analogously, if $(\vec{c}(\alpha_1))$ is the last component of $X$, then $n_\uparrow(X') = n_\uparrow(X)$. Finally, if $(\vec{c}(\alpha_1))$ is neither the first component, nor the last component of $X$, then it is both preceded and succeeded in $X$ by a double component. These two double components merge into one when we complement $(\vec{c}(\alpha_1))$. Hence, we lose one double component: $n_\uparrow(X') = n_\uparrow(X) - 1$.

We can thus conclude that

$$n_\uparrow(X') = n_\uparrow(X) + b_\uparrow + b_\downarrow - 1,$$

where

$$b_\uparrow = \begin{cases} 1 & \text{if } (\vec{c}(\alpha_1)) \text{ is the first component of } X \\ 0 & \text{otherwise} \end{cases}$$
6.2 Blocks of components of a formal DNA molecule

\[ b_l = \begin{cases} 
1 & \text{if } (c_{1(\alpha_1)})^- \text{ is the last component of } X \\
0 & \text{otherwise.} 
\end{cases} \]

Indeed, \( n_t(X') \) satisfies the inequalities in Claims \( 5 \) and \( 6 \).

When we apply the function \( \kappa \) to a formal DNA molecule \( X \), all single-stranded components of \( X \) are complemented. The new values for the counting functions can be related to the old ones, as follows:

**Lemma 6.19** Let \( X \) be a formal DNA molecule, and let \( r \geq 0 \) be the number of single-stranded components of \( X \). Then \( \kappa(X) \) is a formal DNA molecule and

1. \( B_t(\kappa(X)) \leq B_t(X) \),
2. \( B_t(\kappa(X)) \geq B_t(X) - r \),
3. \( B_\downarrow(\kappa(X)) \leq B_\downarrow(X) \),
4. \( B_\downarrow(\kappa(X)) \geq B_\downarrow(X) - r \),
5. \( n_t(\kappa(X)) \leq n_t(X) + 2 - r \),
6. \( n_t(\kappa(X)) \leq n_t(X) + 1 \) and
7. \( n_t(\kappa(X)) \geq n_t(X) - r \).

One might think that Claim \( 6 \) does not add much to Claim \( 5 \). Only for the case that \( r = 0 \), it is (slightly) stronger than Claim \( 5 \). Sometimes, however, it is useful to have an upper bound on \( n_t(\kappa(X)) \) that does not depend on the number of single-stranded components of \( X \).

**Proof:** Already at the definition of the function \( \kappa \), we observed that for each formal DNA molecule \( X \), also \( \kappa(X) \) is a formal DNA molecule.

When we complement the \( r \) single-stranded components of \( X \) one by one, we obtain \( \kappa(X) \) (cf. the proof of Lemma \( 6.10 \)). At each of these steps, we can apply Lemma \( 6.18 \) (or the analogue for upper components). Now, each of Claims \( 1 \)–\( 4 \) is the result of the repeated application of the corresponding claim from Lemma \( 6.18 \). Likewise, Claim \( 6 \) follows from Lemma \( 6.18 \) \( 6 \).

5. When we complement one single-stranded component of \( X \), this does not affect the position of the remaining \( r - 1 \) single-stranded components. That is, let \( x'_{i_0} \) be one of these remaining single-stranded components. Then \( x'_{i_0} \) is the first component of \( X \), and if and only if it is the first component of the resulting formal DNA molecule \( X' \). Analogously, \( x'_{i_0} \) is the last component of \( X \), and if and only if it is the last component of \( X' \).

Hence, when we successively complement all single-stranded components of \( X \), there will be at most one for which the variable \( b_l \) from \( 6.1 \) is 1, and there will be at most one for which the variable \( b_l \) is 1. Then the total contribution of all variables \( b_l \) and \( b_l \) is at most 2, which proves the claim.
6. We may regard this claim as a corollary to the previous claim, when we separately observe that it holds trivially for the case that \( X \) does not have any single-stranded component (i.e., that \( \kappa(X) = X \)). We may also prove it as follows:

The function \( \kappa \) neither introduces, nor removes nick letters. In particular, it does not change the number of nick letters in the formal DNA molecule \( X \): \( \#\downarrow \triangle \) \( \kappa(X) = \#\downarrow \triangle \) \( X \). When we combine this with Lemma \( 6.12(3) \) and \( (5) \), we obtain

\[
n_\uparrow(\kappa(X)) = \#\downarrow \triangle(\kappa(X)) + 1 = \#\downarrow \triangle(X) + 1 \leq n_\uparrow(X) + 1.
\]

As an aside, we investigate which formal DNA molecules achieve the lower bounds for \( B_\uparrow(\kappa(X)) \) and \( B_\downarrow(\kappa(X)) \) from Lemma \( 6.19(2) \) and \( (4) \). Intuitively, whenever we complement an upper component in such a molecule, the corresponding primitive \( \uparrow \)-block disappears and there are a preceding and a succeeding primitive \( \downarrow \)-block that merge into one. The complementation of a lower component has analogous effects.

We first consider an example.

**Example 6.20** The formal DNA molecule depicted in Figure 6.5 has \( r = 4 \) single-stranded components. After application of the function \( \kappa \), both the number of primitive \( \uparrow \)-blocks and the number of primitive \( \downarrow \)-blocks have decreased by 4. Note that in the original molecule each upper component is ‘enclosed’ by two upper nick letters, and the (only) lower component is ‘enclosed’ by two lower nick letters.

In two steps, we will derive a formal characterization of (arbitrary) formal DNA molecules that achieve the two lower bounds. Some of the arguments we use in the proofs, in fact also underlay Lemma \( 6.18(1) \)–\( (4) \). Because we did not give a detailed proof of those claims, it does not suffice here to simply refer to their proof. Therefore, we work out the argumentation completely.

**Lemma 6.21** Let \( X \) be a formal DNA molecule. If \( X \) contains

- either a lower component that is not the only \( \downarrow \)-component of its primitive \( \downarrow \)-block,
- or an upper component that is not the only \( \uparrow \)-component of its primitive \( \uparrow \)-block

(or both), then \( X \) achieves neither of the lower bounds from Lemma \( 6.19(2) \) and Lemma \( 6.19(4) \).
6.2 Blocks of components of a formal DNA molecule

\[ X: \begin{array}{cccccc}
\alpha_1 & \alpha_2 & \alpha_3 & c(\alpha_4) & \alpha_5 & \alpha_6 \\
\end{array} \quad \kappa(X): \begin{array}{cccccc}
\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 \\
\end{array} \]

Figure 6.6: The formal DNA molecule from Example 6.22, which does not achieve the lower bounds from Lemma 6.19(2) and (1), see Lemma 6.21. (a) The original formal DNA molecule \( X \), with two single-stranded components, \( B_\uparrow(X) = 2 \) and \( B_\downarrow(X) = 1 \). The upper component is not the only \( \uparrow \)-component of its primitive \( \uparrow \)-block. (b) \( \kappa(X) \), with \( B_\uparrow(\kappa(X)) = 1 \) and \( B_\downarrow(\kappa(X)) = 0 \).

Example 6.22 Consider the formal DNA molecule \( X \) depicted in Figure 6.6(a), which has \( r = 2 \) single-stranded components: an upper component and a lower component.

\[ \text{The upper component is not the only \( \uparrow \)-component of its primitive \( \uparrow \)-block. Indeed, } \]

\[ B_\uparrow(\kappa(X)) = 1 > 2 - 2 = B_\uparrow(X) - r \text{ and } B_\downarrow(\kappa(X)) = 0 > 1 - 2 = B_\downarrow(X) - r. \]

Proof of Lemma 6.21 Let \( x_1' \ldots x_k' \) for some \( k \geq 1 \) be the decomposition of \( X \). Assume that \( x_{i_0}' \), with \( 1 \leq i_0 \leq k \) is a lower component of \( X \) which is not the only \( \downarrow \)-component of its primitive \( \downarrow \)-block \( X_1 \).

Let \( X' \) be the formal DNA molecule that results when we complement \( x_{i_0}' \), and let \( X_1' \) be the substring of \( X' \) that corresponds to \( X_1 \). By Lemma 3.7, we must have \( k \geq 3 \) and \( x_{i_0}' \) is preceded and/or succeeded in \( X \) by a double component. By the definition of a primitive \( \downarrow \)-block, these components are parts of \( X_1 \).

Consequently, when we complement \( x_{i_0}' \), the resulting (extended) double component is part of \( X_1' \). Because the other components of \( X \) are not affected by the complementation, \( X_1' \) is a sequence of components of \( X' \).

Because \( x_{i_0}' \) is not the only \( \downarrow \)-component of \( X_1 \), there is at least one \( \downarrow \)-component left in \( X_1' \). It is easy to verify that \( X_1' \) also satisfies the other conditions of a primitive \( \downarrow \)-block. Hence, one primitive \( \downarrow \)-block \( (X_1) \) has been replaced by another \( (X_1') \). Moreover, as in the proof of Lemma 6.14, the other primitive blocks of \( X \) (if any) are not affected, possibly apart from an extension of a double component. In particular, the number of primitive \( \uparrow \)-blocks and the number of primitive \( \downarrow \)-blocks are unchanged: \( B_\uparrow(X') = B_\uparrow(X) \) and \( B_\downarrow(X') = B_\downarrow(X) \).

Let us use \( r \) to denote the number of single-stranded components of \( X \). Clearly, \( r \geq 1 \), the number of single-stranded components of \( X' \) is \( r - 1 \) and \( \kappa(X) = \kappa(X') \). When we apply Lemma 6.19(2) and (1) to \( X' \), we obtain:

\[ B_\uparrow(\kappa(X)) = B_\uparrow(\kappa(X')) \geq B_\uparrow(X') - (r - 1) = B_\uparrow(X) - (r - 1) \quad \text{and} \]

\[ B_\downarrow(\kappa(X)) = B_\downarrow(\kappa(X')) \geq B_\downarrow(X') - (r - 1) = B_\downarrow(X) - (r - 1). \]

Hence, \( X \) achieves neither of the lower bounds from Lemma 6.19(2) and (1).

The proof for the case that \( X \) contains an upper component that is not the only \( \uparrow \)-component of its primitive \( \uparrow \)-block, is analogous.

\[ \square \]

Lemma 6.23 Let \( X \) be a formal DNA molecule and let \( x_1' \ldots x_k' \) for some \( k \geq 1 \) be the decomposition of \( X \).

1. \( X \) achieves the lower bound from Lemma 6.19(2), if and only if

(a) for each lower component \( x_{i_0}' \) of \( X \), \( 3 \leq i_0 \leq k - 2 \) and both \( x_{i_0 - 2}' \) and \( x_{i_0 + 2}' \) are lower nick letters, and
Figure 6.7: The formal DNA molecule from Example 6.24, which achieves the lower bound from Lemma 6.19(2), but does not achieve the lower bound from Lemma 6.19(4). (a) The original formal DNA molecule \( X \), with two single-stranded components, \( B_1(X) = 3 \) and \( B_2(X) = 2 \). The upper component is not enclosed by two upper nick letters. (b) \( \kappa(X) \), with \( B_1(\kappa(X)) = 1 \) and \( B_4(\kappa(X)) = 1 \).

(b) for each upper component \( x'_{i_0} \) of \( X \),

- if \( i_0 \geq 3 \), then \( x'_{i_0-2} \) is an upper nick letter, and
- if \( i_0 \leq k - 2 \), then \( x'_{i_0+2} \) is an upper nick letter.

2. \( X \) achieves the lower bound from Lemma 6.19(4), if and only if

(a) for each upper component \( x'_{i_0} \) of \( X \), \( 3 \leq i_0 \leq k - 2 \) and both \( x'_{i_0-2} \) and \( x'_{i_0+2} \) are upper nick letters, and

(b) for each lower component \( x'_{i_0} \) of \( X \),

- if \( i_0 \geq 3 \), then \( x'_{i_0-2} \) is a lower nick letter, and
- if \( i_0 \leq k - 2 \), then \( x'_{i_0+2} \) is a lower nick letter.

3. \( X \) achieves the lower bounds from both Lemma 6.19(2) and Lemma 6.19(4), if and only if

(a) for each lower component \( x'_{i_0} \) of \( X \), \( 3 \leq i_0 \leq k - 2 \) and both \( x'_{i_0-2} \) and \( x'_{i_0+2} \) are lower nick letters, and

(b) for each upper component \( x'_{i_0} \) of \( X \), \( 3 \leq i_0 \leq k - 2 \) and both \( x'_{i_0-2} \) and \( x'_{i_0+2} \) are upper nick letters.

Note that, if \( x'_0 \) with \( i_0 \geq 3 \) is a single-stranded component and \( x'_{i_0-2} \) is a nick letter, then in fact \( i_0 \geq 4 \), because (the formal DNA molecule) \( X \) cannot start with a nick letter. Analogously, if \( x'_0 \) with \( i_0 \leq k - 2 \) is a single-stranded component and \( x'_{i_0+2} \) is a nick letter, then in fact \( i_0 \leq k - 3 \).

Example 6.24 The formal DNA molecule \( X \) depicted in Figure 6.7(a) has \( r = 2 \) single-stranded components: an upper component and a lower component. The lower component is enclosed by two lower nick letters, but the upper component is not preceded by an upper nick letter. Figure 6.7(b) shows \( \kappa(X) \). We have \( B_1(\kappa(X)) = 1 = 3 - 2 = B_1(X) - r \) and \( B_4(\kappa(X)) = 1 > 2 = B_4(X) - r \).

Proof: Recall that if \( X \) is not double-complete, then by Lemma 6.9(1), \( X \) can be considered as an alternating sequence of all its primitive \( \uparrow \)-blocks and all its primitive \( \downarrow \)-blocks.

1. \( \implies \) Assume that \( X \) does not satisfy Condition 1a or Condition 1b (or both). There is at least one single-stranded component that is responsible for the violation of the condition concerned. Let \( r \geq 1 \) be the number of single-stranded components of \( X \). First, we assume that \( X \) does not satisfy Condition 1a. Then \( X \) has a lower component \( x'_{i_0} \), such that

\[ \alpha_1 \downarrow \alpha_3 \alpha_4 \alpha_6 \alpha_7 \]

\[ \alpha_1 \alpha_2 \alpha_3 \alpha_4 \alpha_5 \alpha_6 \alpha_7 \]
6.2 Blocks of components of a formal DNA molecule

- either $i_0 \leq 2$,
- or $i_0 \geq k - 1$,
- or $3 \leq i_0 \leq k - 2$ and either $x'_{i_0-2}$ or $x'_{i_0+2}$ (or both) is not a lower nick letter.

Let $X_1$ be the primitive $\downarrow$-block of $x'_{i_0}$. We consider several cases:

- If $x'_{i_0}$ is not the only $\downarrow$-component of $X_1$, then by Lemma [6.21] $X$ does not achieve the lower bound from Lemma [6.19][2].
- If $B_\uparrow(X) = 0$, i.e., if $X$ does not have any primitive $\uparrow$-block, then $B_\uparrow(X) - r \leq -1$ and $X$ certainly does not achieve the lower bound from Lemma [6.19][2].
- Now assume that $x'_{i_0}$ is the only $\downarrow$-component of $X_1$ and that $B_\uparrow(X) > 0$. Let $X'$ be the formal DNA molecule that we obtain from $X$ when we complement $x'_{i_0}$. Clearly, the number of single-stranded components of $X'$ is $r - 1$, and $\kappa(X) = \kappa(X')$.

If $i_0 \leq 2$, then $x'_{i_0}$ is the first non-double component of $X$. Because, by definition, each primitive $\uparrow$-block contains at least one $\uparrow$-component, the primitive $\downarrow$-block $X_1$ is not preceded in $X$ by a primitive $\uparrow$-block. By assumption, $B_\uparrow(X) > 0$. Hence, $X_1$ must be succeeded by a primitive $\uparrow$-block $X_2$, which starts with the double component $x'_{i_0+1}$. When we complement the lower component $x'_{i_0}$, we lose the primitive $\downarrow$-block $X_1$, and the first (double) component of the primitive $\uparrow$-block $X_2$ is simply extended to the left. The number of primitive $\uparrow$-blocks remains the same: $B_\uparrow(X') = B_\uparrow(X)$. Now, when we apply Lemma [6.19][2] to $X'$, we find:

$$B_\uparrow(\kappa(X')) = B_\uparrow(\kappa(X')) \geq B_\uparrow(X') - (r - 1) = B_\uparrow(X) - (r - 1).$$

In this case, $X$ does not achieve the lower bound from Lemma [6.19][2].

If $i_0 \geq k - 1$, then in a completely analogous way, we come to the same conclusion.

Finally, we assume that $3 \leq i_0 \leq k - 2$. Then either $x'_{i_0-2}$ or $x'_{i_0+2}$ (or both) is not a lower nick letter. Without loss of generality, assume that the component $x'_{i_0-2}$ is not a lower nick letter. By Lemma [3.7] both $x'_{i_0-1}$ and $x'_{i_0+1}$ are double components of $X$. Because $x'_{i_0}$ is the only $\downarrow$-component of $X_1$, $X_1 = x'_{i_0-1}x'_{i_0}x'_{i_0+1}$, and both $x'_{i_0-2}$ and $x'_{i_0+2}$ are $\uparrow$-components. Now, $x'_{i_0-2}$ must be an upper component, which is part of a primitive $\uparrow$-block $X_0$, ending with the double component $x'_{i_0-1}$. $X_1$ is succeeded in $X$ by a primitive $\uparrow$-block $X_2$, which starts with the double component $x'_{i_0+1}$. The existence of the upper component $x'_{i_0-2}$ implies that $r \geq 2$.

When we complement the lower component $x'_{i_0}$, we obtain an extended double component $x'_{i_0-1}\kappa(x'_{i_0})x'_{i_0+1}$. Thus, we lose the primitive $\downarrow$-block $X_1$, and the two primitive $\uparrow$-blocks $X_0$ and $X_2$ form one ‘large’ primitive $\uparrow$-block $X'_{i_0}$ of the resulting formal DNA molecule $X'$. Hence, $B_\uparrow(X') = B_\uparrow(X) - 1$. We now consider the upper component $x'_{i_0-2}$ of $X'$. Its primitive $\uparrow$-block $X'_{i_0}$ in $X'$ contains at least one more $\uparrow$-component, viz $x'_{i_0+2}$. By Lemma [6.21] $X'$ does not achieve the lower bound from Lemma [6.19][2]:

$$B_\uparrow(\kappa(X')) \geq B_\uparrow(X') - (r - 1) + 1.$$
Consequently,
\[ B_k(\kappa(X)) = B_k(\kappa(X')) \geq B_k(X') - (r - 1) + 1 = B_k(X) - (r - 1). \]

We conclude that also in this case, \( X \) does not achieve the lower bound from Lemma \[6.19\][2].

Now, we assume that \( X \) does satisfy Condition \[1\]a and hence does not satisfy Condition \[1\]b. Then \( X \) has an upper component \( x'_{i_0} \), such that either \( i_0 \geq 3 \) and \( x'_{i_0-2} \) is not an upper nick letter, or \( i_0 \leq k - 2 \) and \( x'_{i_0+2} \) is not an upper nick letter. Without loss of generality, we assume that \( i_0 \geq 3 \) and \( x'_{i_0-2} \) is not an upper nick letter. Let \( X_1 \) be the primitive ↑-block that \( x'_{i_0} \) is part of.

Because \( X \) satisfies Condition \[1\]a and \( x'_{i_0} \) is not a lower nick letter, \( x'_{i_0-2} \) cannot be a lower component. Hence, it must be either an upper component or a lower nick letter. In both cases, it is an ↑-component of \( X \), which implies that \( x'_{i_0} \) is not the only ↑-component of \( X_1 \). Again by Lemma \[6.21\], \( X \) does not achieve the lower bound from Lemma \[6.19\][2].

\[ \iff \] By induction on the number \( r \) of single-stranded components of \( X \).

- If \( r = 0 \), then \( X \) trivially satisfies Conditions \[1\]a and \[1\]b from the claim. On the other hand, in this case, the function \( \kappa \) has no effect on \( X \): \( \kappa(X) = X \). Hence, \( X \) also trivially achieves the lower bound from Lemma \[6.19\][2]:
  \[ B_k(\kappa(X)) = B_k(X) = B_k(X) - r. \]

- Let \( r \geq 0 \), and suppose that the lower bound is achieved by each formal DNA molecule satisfying Conditions \[1\]a and \[1\]b and containing \( r \) single-stranded components (induction hypothesis). Now let \( X \) be a formal DNA molecule that satisfies the two conditions and contains \( r + 1 \) single-stranded components.

Let \( x'_{i_0} \) with \( 1 \leq i_0 \leq k \) be an arbitrary single-stranded component of \( X \), and let \( X' \) be the formal DNA molecule that results after complementing \( x'_{i_0} \). We prove that \( B_k(X') = B_k(X) - 1 \).

First, we assume that \( x'_{i_0} \) is a lower component. By Condition \[1\]a, \( 3 \leq i_0 \leq k - 2 \) and \( x'_{i_0-2} \) and \( x'_{i_0+2} \) are lower nick letters, i.e., ↑-components. As before, both \( x'_{i_0-1} \) and \( x'_{i_0+1} \) are double components of \( X \), and the primitive ↓-block containing \( x'_{i_0} \) is \( X_1 = x'_{i_0-1}x'_{i_0}x'_{i_0+1} \).

In \( X' \), the primitive ↓-block \( X_1 \) has been replaced by the double component \( x'_{i_0-1}x'_{i_0}x'_{i_0+1} \). Further, the two primitive ↑-blocks containing the lower nick letters \( x'_{i_0-2} \) and \( x'_{i_0+2} \), respectively, have merged into one. Consequently, in this case, \( B_k(X') = B_k(X) - 1 \).

Now, we assume that \( x'_{i_0} \) is an upper component. Let \( X_1 \) be the primitive ↑-block of \( x'_{i_0} \). By Condition \[1\]b, either \( i_0 \leq 2 \), or \( i_0 \geq 3 \) and \( x'_{i_0-2} \) is an upper nick letter. In both cases, \( x'_{i_0} \) is the first ↑-component of \( X_1 \). In an analogous way, we can find that \( x'_{i_0} \) is the last, and thus the only ↑-component of \( X_1 \). Besides \( x'_{i_0} \), \( X_1 \) may only contain one or two double components.

When we complement \( x'_{i_0} \), the primitive ↑-block \( X_1 \) turns into a double component of \( X' \). Other primitive ↑-blocks of \( X \) are not affected by the complementation. Also in this case, \( B_k(X') = B_k(X) - 1 \).
6.2 Blocks of components of a formal DNA molecule

Before we can apply the induction hypothesis to \(X'\), we must verify that it satisfies Conditions 1a and 1b. For that purpose, we examine the single-stranded components of \(X'\). Obviously, these are the single-stranded components of \(X\) different from \(x'_{i_0}\). We no longer assume that \(x'_{i_0}\) is a lower component of \(X\) or that it is an upper component of \(X\). It may be either of the two.

Let \(x'_{i_1}\) with \(3 \leq i_1 \leq k - 2\) and \(i_1 \neq i_0\) be an arbitrary lower component of \(X\) different from \(x'_{i_0}\). By assumption, \(x'_{i_1-2}\) and \(x'_{i_1+2}\) are lower nick letters. Neither of these lower nick letters, nor the double components \(x'_{i_1-1}\) and \(x'_{i_1+1}\) or the lower component \(x'_{i_1}\) itself are affected by the complementation of \(x'_{i_0}\). They occur in \(X'\) like they do in \(X\). Hence, \(X'\) satisfies Condition 1a.

Now, let \(x'_{i_1}\) with \(1 \leq i_1 \leq k\) and \(i_1 \neq i_0\) be an arbitrary upper component of \(X\) different from \(x'_{i_0}\). This upper component also occurs in \(X'\). If \(i_1 \geq 3\), then \(x'_{i_1-1}\) is a double component and by assumption, \(x'_{i_1-2}\) is an upper nick letter. These components are not affected when we complement \(x'_{i_0}\). Analogously, if \(i_1 \leq k - 2\), then the double component \(x'_{i_1+1}\) and the upper nick letter \(x'_{i_1+2}\) are not affected. Obviously, the complementation does not introduce components before \(x'_{i_1}\) (relevant if \(i_1 \leq 2\)) or after \(x'_{i_1}\) (relevant if \(i_1 \geq k - 1\)). Hence, \(X'\) satisfies Condition 1b just like \(X\).

Because \(X'\) has \(r\) single-stranded components, we can apply the induction hypothesis to it. When we observe that \(\kappa(X) = \kappa(X')\), we find

\[
B_{\uparrow}(\kappa(X)) = B_{\uparrow}(\kappa(X')) = B_{\uparrow}(X') - r = B_{\uparrow}(X) - (r + 1).
\]

We conclude that \(X\) achieves the lower bound from Lemma 6.19[2].

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

3. The condition on the lower components from Claim 1 implies the condition on the lower components from Claim 2 and conversely for the conditions on the upper components. Hence, Conditions 1a 1b 2a and 2b (together) are equivalent to Conditions 3a and 3b. Now, the claim follows immediately from the other two claims.

\[\square\]

We can now conclude that many formal DNA molecules that achieve either of the lower bounds from Lemma 6.19[2] and (1) are not expressible:

**Corollary 6.25** Let \(X\) be a formal DNA molecule containing at least one upper component and at least one lower component. If \(X\) achieves either the lower bound from Lemma 6.19[2], or the one from Lemma 6.19[3] (or both), then \(X\) is not expressible.

**Proof:** Let \(x'_1 \ldots x'_k\) for some \(k \geq 1\) be the decomposition of \(X\). Assume that \(X\) achieves the lower bound from Lemma 6.19[2]. Then \(X\) satisfies the two conditions in Lemma 6.23[1].

Because \(X\) contains at least one lower component, by Condition 1a, it also contains at least two lower nick letters. Let \(x'_{i_0}\) with \(1 \leq i_0 \leq k\) be an upper component of \(X\). By Lemma 3.7, double components and non-double components alternate in \(X\). Because \(X\) also contains a lower component (and two lower nick letters), we must have either \(i_0 \geq 3\) or \(i_0 \leq k - 2\) (or both). Then by Condition 1b, \(X\) contains at least one upper nick letter.
By Theorem 5.4, formal DNA molecules containing nick letters in both strands are not expressible.

The proof for the case that $X$ achieves the lower bound from Lemma 6.19(1) is analogous.

This concludes the aside that we started after the proof of Lemma 6.19, i.e., the analysis of formal DNA molecules that achieve the lower bounds from Lemma 6.19(2) and (4).

6.3 Lower bounds for the length of a DNA expression

In the previous section, we have analysed the values of the counting functions $B_\uparrow$, $B_\downarrow$ and $n_\leftrightarrow$ for arbitrary formal DNA molecules. We now study these values for molecules denoted by DNA expressions. The results are useful to determine lower bounds for the length of a DNA expression.

We first examine $\uparrow$-expressions with (exactly) two arguments. Using induction, we will later extend the results to $\uparrow$-expressions with an arbitrary number of arguments. Of course, we can find analogous results for $\downarrow$-expressions.

Recall that the effect of the operator $\uparrow$ is threefold: (1) it produces upper $A$-words corresponding to arguments that are $N$-words, (2) it removes nick letters from the upper strands of its arguments, and (3) it connects the upper strands of consecutive arguments. When we examine the effect of $\uparrow$ on the values of the counting functions, we must take into account the contributions of each of these three aspects.

The values of the counting functions for upper $A$-words are independent of the $A$-word at hand and follow immediately from the definitions. In Lemma 6.16, we already considered the effects on the counting numbers of removing the upper nick letters from a (single) formal DNA molecule. Now, we will in particular study the effects of connecting the upper strands of the formal DNA molecules corresponding to the (two) arguments of an $\uparrow$-expression.

Lemma 6.26 Let $E = \langle \uparrow \varepsilon_1 \varepsilon_2 \rangle$ be an $\uparrow$-expression, where $\varepsilon_1$ and $\varepsilon_2$ are $N$-words or DNA expressions. Further, let $X_1 = S^+(\varepsilon_1)$, $X_2 = S^+(\varepsilon_2)$ and

$$X = S(E) = \nu^+(X_1)y_1\nu^+(X_2),$$

where $y_1 = \triangle$ if both $R(X_1) \in A_\pm$ and $L(X_2) \in A_\pm$, and $y_1 = \lambda$ otherwise (as in Definition 4.1).

1. $B_\uparrow(X) \leq B_\uparrow(\nu^+(X_1)) + B_\uparrow(\nu^+(X_2)) + |y_1|.$
2. $B_\uparrow(X) \geq B_\uparrow(\nu^+(X_1)) + B_\uparrow(\nu^+(X_2)) - 1.$
3. $B_\downarrow(X) = B_\downarrow(\nu^+(X_1)) + B_\downarrow(\nu^+(X_2)).$
4. $n_\leftrightarrow(X) = n_\leftrightarrow(\nu^+(X_1)) + n_\leftrightarrow(\nu^+(X_2)).$

Note that $|y_1| = 1$ if $y_1 = \triangle$, and $|y_1| = 0$ if $y_1 = \lambda$.

Proof: We first make a remark on the $\downarrow$-components of the formal DNA molecules we consider. Because the function $\nu^+$ removes the upper nick letters from its argument, each $\downarrow$-component of $\nu^+(X_1)$ and $\nu^+(X_2)$ (and thus of $X$) is in fact a lower component. For the
consistency of the terminology, we will, however, use the term ↓-component throughout the proof.

Recall again that if a formal DNA molecule is not double-complete, then by Lemma 6.9[11], it can be considered as an alternating sequence of all its primitive ↑-blocks and all its primitive ↓-blocks.

X is simply the concatenation of ν(X₁) and ν(X₂), possibly separated by a lower nick letter y₁. Clearly, none of the primitive ↑-blocks, primitive ↓-blocks and double components present in ν(X₁) or ν(X₂) is split up by this concatenation. We may have some effects, however, on the blocks and components at the border between ν(X₁) and ν(X₂).

By definition, X₁ and X₂ fit together by upper strands. Hence, both R(X₁) ∈ A⁺∪A⁻ and L(X₂) ∈ A⁺∪A⁻. Because, by Lemma 3.1 R(ν(X₁)) = R(X₁), the last component of ν(X₁) is either a double component or an upper component. Analogously, the first component of ν(X₂) is either a double component or an upper component.

Now assume that ν⁺(X₁) is not double-complete and ends with a primitive ↓-block. Then by Lemma 6.9[28], the last non-double component of ν⁺(X₁) is a ↓-component. This implies that the last component of ν⁺(X₁) cannot be an upper component; it must be a double component.

Analogously, if ν⁺(X₂) is not double-complete and starts with a primitive ↓-block, then the first component of ν⁺(X₂) is a double component.

We distinguish three cases:

• If ν⁺(X₁) does not end with a primitive ↑-block, then either ν⁺(X₁) is double-complete, or it ends with a primitive ↓-block. In both cases, the last component of ν⁺(X₁) is a double component x₁'. If, in addition, ν⁺(X₂) does not start with a primitive ↑-block, then, analogously, its first component is a double component x₂'. Consequently, R(X₁) ∈ A⁺ and L(X₂) ∈ A⁺ and y₁ = ∅. This lower nick letter is part of a primitive ↑-block X₁₂ of X, which also contains the double components x₁' (from ν⁺(X₁)) and x₂' (from ν⁺(X₂)).

Each ↑-component of ν⁺(X₁) (if any) is separated from y₁ in X by at least one ↓-component. Otherwise, the last non-double component of ν⁺(X₁) would be an ↑-component and, by Lemma 6.9[28], ν⁺(X₁) would end with a primitive ↑-block. Analogously, each ↑-component of ν⁺(X₂) (if any) is separated from y₁ in X by at least one ↓-component. Consequently, all primitive ↑-blocks of ν⁺(X₁) and ν⁺(X₂) are also primitive ↑-blocks of X, and X₁₂ = x₁'y₁x₂' is an additional primitive ↑-block:

\[ B_1(X) = B_1(ν⁺(X₁)) + B_1(ν⁺(X₂)) + 1. \]

• If ν⁺(X₁) does not end with a primitive ↑-block, then, as before, its last component is a double component x₁'. Either this component is the only component of ν⁺(X₁) (if ν⁺(X₁) is double-complete), or it is preceded in ν⁺(X₁) by a ↓-component (if ν⁺(X₁) ends with a primitive ↓-block). Now, if ν⁺(X₂) is not double-complete and starts with a primitive ↑-block, then in X, this primitive ↑-block is extended to the left by x₁' and possibly a lower nick letter y₁. Whether y₁ = ∅ or y₁ = λ depends on the first component of ν⁺(X₂). It is, however, not important for the number of primitive ↑-blocks. The primitive ↑-blocks of ν⁺(X₁) and the other primitive ↑-blocks of ν⁺(X₂) simply reappear as primitive ↑-blocks of X. Hence,

\[ B_1(X) = B_1(ν⁺(X₁)) + B_1(ν⁺(X₂)). \]
We obtain, of course, the same equality, if $\nu^+(X_1)$ ends with a primitive $\uparrow$-block and $\nu^+(X_2)$ does not start with a primitive $\uparrow$-block.

- If both $\nu^+(X_1)$ ends with a primitive $\uparrow$-block and $\nu^+(X_2)$ starts with a primitive $\uparrow$-block, then these two primitive $\uparrow$-blocks form one primitive $\uparrow$-block in $X$. Again, it does not matter if $y_1 = \triangle$ or $y_1 = \lambda$. A lower nick letter, which is an $\uparrow$-component, would fit perfectly into the combined primitive $\uparrow$-block. The other primitive $\uparrow$-blocks of $\nu^+(X_1)$ and $\nu^+(X_2)$ are not affected. In this case, we lose one primitive $\uparrow$-block:

$$B_{\uparrow}(X) = B_{\uparrow}(\nu^+(X_1)) + B_{\uparrow}(\nu^+(X_2)) - 1.$$ 

We have summarized the possibilities in Table 6.1. In all cases, $B_{\uparrow}(X)$ satisfies the inequalities in Claims 1 and 2.

Table 6.1: Number of primitive $\uparrow$-blocks of a formal DNA molecule $X = S(\langle \uparrow \epsilon_1 \epsilon_2 \rangle) = \nu^+(X_1)y_1\nu^+(X_2)$ for all possible combinations of $\nu^+(X_1)$ and $\nu^+(X_2)$. The formal DNA molecule $\nu^+(X_1)$ either is double-complete, or ends with a primitive $\downarrow$-block or a primitive $\uparrow$-block. Similarly, $\nu^+(X_2)$ either is double-complete, or starts with a primitive $\downarrow$-block or a primitive $\uparrow$-block (see the proof of Lemma 6.26(1) and (2)).

<table>
<thead>
<tr>
<th>end $\nu^+(X_1)$</th>
<th>start $\nu^+(X_2)$</th>
<th>$y_1$</th>
<th>$B_{\uparrow}(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>double-complete</td>
<td>double-complete</td>
<td>$\triangle$</td>
<td>$B_{\uparrow}(\nu^+(X_1)) + B_{\uparrow}(\nu^+(X_2)) + 1 = 1$</td>
</tr>
<tr>
<td>double-complete</td>
<td>prim. $\downarrow$-block</td>
<td>$\triangle$</td>
<td>$B_{\uparrow}(\nu^+(X_1)) + B_{\uparrow}(\nu^+(X_2)) + 1 = B_{\uparrow}(\nu^+(X_2)) + 1$</td>
</tr>
<tr>
<td>double-complete</td>
<td>prim. $\uparrow$-block</td>
<td>$\triangle / \lambda$</td>
<td>$B_{\uparrow}(\nu^+(X_1)) + B_{\uparrow}(\nu^+(X_2)) = B_{\uparrow}(\nu^+(X_2))$</td>
</tr>
<tr>
<td>prim. $\downarrow$-block</td>
<td>double-complete</td>
<td>$\triangle$</td>
<td>$B_{\uparrow}(\nu^+(X_1)) + B_{\uparrow}(\nu^+(X_2)) + 1 = B_{\uparrow}(\nu^+(X_1)) + 1$</td>
</tr>
<tr>
<td>prim. $\downarrow$-block</td>
<td>prim. $\downarrow$-block</td>
<td>$\triangle$</td>
<td>$B_{\uparrow}(\nu^+(X_1)) + B_{\uparrow}(\nu^+(X_2)) + 1$</td>
</tr>
<tr>
<td>prim. $\uparrow$-block</td>
<td>prim. $\uparrow$-block</td>
<td>$\triangle / \lambda$</td>
<td>$B_{\uparrow}(\nu^+(X_1)) + B_{\uparrow}(\nu^+(X_2)) = B_{\uparrow}(\nu^+(X_1))$</td>
</tr>
<tr>
<td>prim. $\uparrow$-block</td>
<td>double-complete</td>
<td>$\triangle / \lambda$</td>
<td>$B_{\uparrow}(\nu^+(X_1)) + B_{\uparrow}(\nu^+(X_2))$</td>
</tr>
<tr>
<td>prim. $\uparrow$-block</td>
<td>prim. $\downarrow$-block</td>
<td>$\triangle / \lambda$</td>
<td>$B_{\uparrow}(\nu^+(X_1)) + B_{\uparrow}(\nu^+(X_2)) - 1$</td>
</tr>
</tbody>
</table>

3. In principle, the concatenation of two formal DNA molecules may cause a decrease of the total number of primitive $\downarrow$-blocks. When we concatenate a formal DNA molecule ending with a primitive $\downarrow$-block and a formal DNA molecule starting with a primitive $\downarrow$-block, these two primitive $\downarrow$-blocks merge into one.

However, if both $\nu^+(X_1)$ ends with a primitive $\downarrow$-block $X_{11}$ and $\nu^+(X_2)$ starts with a primitive $\downarrow$-block $X_{21}$, then both the last double component of $\nu^+(X_1)$ and the first double component of $\nu^+(X_2)$ are double components. Hence, $R(X_1) \in A_{\pm}$ and $L(X_2) \in A_{\pm}$. Then the primitive $\downarrow$-blocks $X_{11}$ and $X_{21}$ are separated in $X$ by the lower nick letter $y_1$, which is an $\uparrow$-component. They do not merge into one and there is a 1–1 correspondence between the primitive $\downarrow$-blocks of $\nu^+(X_1)$ and $\nu^+(X_2)$ and the (same) primitive $\downarrow$-blocks of $X$:

$$B_{\downarrow}(X) = B_{\downarrow}(\nu^+(X_1)) + B_{\downarrow}(\nu^+(X_2)).$$

It is easily verified that this 1–1 correspondence certainly exists, if either $\nu^+(X_1)$ does not end with a primitive $\downarrow$-block, or $\nu^+(X_2)$ does not start with a primitive $\downarrow$-block (or both).
4. If both the last component of \( \nu^+(X_1) \) and the first component of \( \nu^+(X_2) \) are double components, then \( R(X_1) \in \mathcal{A}_\pm \) and \( L(X_2) \in \mathcal{A}_\pm \). By definition, the two double components are separated in \( X \) by the lower nick letter \( y_1 \). Hence, there is a 1–1 correspondence between the double components of \( \nu^+(X_1) \) and \( \nu^+(X_2) \) and the (same) double components of \( X \):

\[
n_{\uparrow}(X) = n_{\uparrow}(\nu^+(X_1)) + n_{\uparrow}(\nu^+(X_2)).
\]

It is easily verified that this 1–1 correspondence certainly exists, if either the last component of \( \nu^+(X_1) \), or the first component of \( \nu^+(X_2) \) (or both) is not a double component.

We now consider arbitrary DNA expressions.

**Lemma 6.27** Let \( E \) be a DNA expression, and let \( X = S(E) \).

1. If \( E = (\uparrow \varepsilon_1 \ldots \varepsilon_n) \), where \( n \geq 1 \) and the arguments \( \varepsilon_1, \ldots, \varepsilon_n \) are \( N \)-words and DNA expressions, then let, for \( i = 1, \ldots, n \), \( X_i = S^+(\varepsilon_i) \).

   \[
   B_{\uparrow}(X) \leq B_{\uparrow}(X_1) + \cdots + B_{\uparrow}(X_n) + \#_\triangle(X) - \#_\triangle(X_1) - \cdots - \#_\triangle(X_n), \quad (6.2)
   
   B_{\downarrow}(X) \geq B_{\downarrow}(X_1) + \cdots + B_{\downarrow}(X_n) - (n-1) - \#_\vee(X_1) - \cdots - \#_\vee(X_n), \quad (6.3)
   
   B_{\uparrow}(X) \leq B_{\uparrow}(X_1) + \cdots + B_{\uparrow}(X_n), \quad (6.4)
   
   B_{\downarrow}(X) \geq B_{\downarrow}(X_1) + \cdots + B_{\downarrow}(X_n) - \#_\vee(X_1) - \cdots - \#_\vee(X_n) \quad \text{and} \quad (6.5)
   
   n_{\uparrow}(X) = n_{\uparrow}(X_1) + \cdots + n_{\uparrow}(X_n) - \#_\vee(X_1) - \cdots - \#_\vee(X_n). \quad (6.6)
   
2. If \( E = (\downarrow \varepsilon_1 \ldots \varepsilon_n) \), where \( n \geq 1 \) and the arguments \( \varepsilon_1, \ldots, \varepsilon_n \) are \( N \)-words and DNA expressions, then let, for \( i = 1, \ldots, n \), \( X_i = S^-(\varepsilon_i) \).

   \[
   B_{\downarrow}(X) \leq B_{\downarrow}(X_1) + \cdots + B_{\downarrow}(X_n) + \#_\vee(X) - \#_\vee(X_1) - \cdots - \#_\vee(X_n), \quad (6.7)
   
   B_{\uparrow}(X) \geq B_{\uparrow}(X_1) + \cdots + B_{\uparrow}(X_n) - (n-1) - \#_\triangle(X_1) - \cdots - \#_\triangle(X_n), \quad (6.8)
   
   B_{\downarrow}(X) \leq B_{\downarrow}(X_1) + \cdots + B_{\downarrow}(X_n), \quad (6.9)
   
   B_{\uparrow}(X) \geq B_{\uparrow}(X_1) + \cdots + B_{\uparrow}(X_n) - \#_\triangle(X_1) - \cdots - \#_\triangle(X_n) \quad \text{and} \quad (6.10)
   
   n_{\downarrow}(X) = n_{\downarrow}(X_1) + \cdots + n_{\downarrow}(X_n) - \#_\triangle(X_1) - \cdots - \#_\triangle(X_n). \quad (6.11)
   
3. If \( E = (\downarrow E_1) \) for a DNA expression \( E_1 \), then let \( X_1 = S(E_1) \) and let \( r \geq 0 \) be the number of single-stranded components of \( X_1 \).

   \[
   B_{\uparrow}(X) \leq B_{\uparrow}(X_1), \quad (6.7)
   
   B_{\downarrow}(X) \geq B_{\downarrow}(X_1) - r, \quad (6.8)
   
   B_{\uparrow}(X) \leq B_{\uparrow}(X_1), \quad (6.9)
   
   B_{\downarrow}(X) \geq B_{\downarrow}(X_1) - r, \quad (6.10)
   
   n_{\uparrow}(X) \leq n_{\uparrow}(X_1) + 2 - r, \quad (6.11)
   
   n_{\downarrow}(X) \leq n_{\downarrow}(X_1) + 1 \quad \text{and} \quad (6.12)
   
   n_{\downarrow}(X) \geq n_{\downarrow}(X_1) - r. \quad (6.13)
   \]
Figure 6.8: Pictorial representation of the $\uparrow$-expression $E$ from Example 6.28 for which the values of the three counting functions are calculated. The primitive $\uparrow$-blocks and the primitive $\downarrow$-blocks of the formal DNA molecules involved are also indicated.

Example 6.28 As an illustration of Claim 1 consider

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

Here $n = 4$, the arguments $\varepsilon_1, \ldots, \varepsilon_4$ have been indicated using additional white space, and

\[
\begin{align*}
X_1 &= (c(a_1)) (c(a_2)) (c(a_3)) (c(a_4)) (c(a_5)) (c(a_6)); \\
X_2 &= (c(a_7)) (c(a_8)) (c(a_9)); \\
X_3 &= (c(a_{10})) (c(a_{11})) (c(a_{12})) (c(a_{13})) (c(a_{14})); \\
X_4 &= (c(a_{15})) (c(a_{16})) (c(a_{17})).
\end{align*}
\]

Then

\[
X = S(E) = (c(a_1a_2)) (c(a_3)) (c(a_4)) (c(a_5)) (c(a_6)) (c(a_7)) (c(a_8));
\]

and

\[
\begin{align*}
B_\uparrow(X) &= 4 \prec (1 + 0 + 1 + 1) + 3 - (0 + 0 + 0 + 1), \\
B_\downarrow(X) &= 4 \succ (1 + 0 + 1 + 1) - 3 - (1 + 0 + 1 + 0), \\
B_i(X) &= 3 \prec 2 + 1 + 1 + 0, \\
B_j(X) &= 3 \succ 2 + 1 + 1 + 0 - (1 + 0 + 1 + 0), \\
n_\uparrow(X) &= 9 = (4 + 2 + 3 + 2) - (1 + 0 + 1 + 0).
\end{align*}
\]

This example is depicted in Figure 6.8.

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

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

does not hold in general. Lower nick letters added between the arguments of $\uparrow$ may introduce new primitive $\uparrow$-blocks. Indeed, for the $\uparrow$-expression we considered above, $B_\uparrow(X) = 4$, whereas $B_\uparrow(X_1) + B_\uparrow(X_2) + B_\uparrow(X_3) + B_\uparrow(X_4) = 3$. The difference $\#_\uparrow(X) - \#_\downarrow(X_1) - \cdots - \#_\downarrow(X_n)$ in inequality (6.2) accounts for the lower nick letters added.

Note also that in Claim 4 we do not consider $\downarrow$-expressions of the form $\langle \downarrow \alpha_1 \rangle$ for an $N$-word $\alpha_1$. For such $\downarrow$-expressions, however, the values of $B_\uparrow$, $B_i$ and $n_\uparrow$ are trivial: if
6.3 Lower bounds for the length of a DNA expression

\[ X = S(\langle \dagger \alpha_1 \rangle) = \left( \frac{\alpha_1}{\epsilon(\alpha_1)} \right), \]
then \( B_\dagger(X) = B_\ddagger(X) = 0 \) and \( n_\ddagger(X) = 1 \).

**Proof of Lemma 6.27:**

1. Let \( E \) be an \( \dagger \)-expression as described in the claim. We prove the five equations by induction on \( n \).

- If \( n = 1 \), then \( E = \langle \dagger \varepsilon_1 \rangle \) for an \( \mathcal{N} \)-word or a DNA expression \( \varepsilon_1 \) and the five equations we have to prove reduce to:

  \[
  \begin{align*}
  B_\dagger(X) & \leq B_\dagger(X_1) + \#_\Delta(X) - \#_\Delta(X_1), \\
  B_\dagger(X) & \geq B_\dagger(X_1) - 0 - \#_\varphi(X_1), \\
  B_\ddagger(X) & \leq B_\ddagger(X_1), \\
  B_\ddagger(X) & \geq B_\ddagger(X_1) - \#_\varphi(X_1) \quad \text{and} \\
  n_\ddagger(X) & = n_\ddagger(X_1) - \#_\varphi(X_1).
  \end{align*}
  \]

  If \( \varepsilon_1 \) is an \( \mathcal{N} \)-word \( \alpha_1 \), then \( E = \langle \dagger \alpha_1 \rangle \) and by definition, \( X = X_1 = \langle \alpha_1 \rangle \).
  Hence, \( \#_\varphi(X_1) = 0 \) and the five equations are trivially valid, with \( B_\dagger(X) = 1, \)
  \( \#_\Delta(X) = 0, B_\ddagger(X) = 0 \) and \( n_\ddagger(X) = 0 \).

  If, on the other hand, \( \varepsilon_1 \) is a DNA expression \( E_1 \), then \( E = \langle \dagger E_1 \rangle \) and by definition, \( X = \nu^+(X_1) \). Because the function \( \nu^+ \) neither introduces nor removes lower nick letters (hence, \( \#_\Delta(X) = \#_\Delta(X_1) \)), the five equations are just special cases of the ones in Lemma 6.16. Whereas, in Lemma 6.16 we considered an arbitrary formal DNA molecule \( X \), we have an expressible formal DNA molecule \( X_1 \) here.

- Let \( n \geq 1 \) and suppose that equations (6.2)–(6.6) hold for all \( \dagger \)-expressions with \( n \) arguments (induction hypothesis). Now let \( E \) be an arbitrary \( \dagger \)-expression with \( n + 1 \) arguments: \( E = \langle \dagger \varepsilon_1 \ldots \varepsilon_n \varepsilon_{n+1} \rangle \) for \( \mathcal{N} \)-words and DNA expressions \( \varepsilon_1, \ldots, \varepsilon_n, \varepsilon_{n+1} \).

  By Lemma 5.10

  \[ E \equiv \langle \dagger \langle \dagger \varepsilon_1 \ldots \varepsilon_n \rangle \varepsilon_{n+1} \rangle, \]

  i.e., \( X = S(E) = S(\langle \langle \dagger \varepsilon_1 \ldots \varepsilon_n \rangle \varepsilon_{n+1} \rangle) \). Let \( X' = S(\langle \langle \dagger \varepsilon_1 \ldots \varepsilon_n \rangle \rangle) \). By definition,

  \[ X = \nu^+(X') y_1 \nu^+(X_{n+1}), \]

  where \( y_1 = \Delta \) if \( R(X'), L(X_{n+1}) \in \mathcal{A}_\pm \) and \( y_1 = \lambda \) otherwise. By Lemma 5.11(1), the semantics of an \( \uparrow \)-expression does not contain upper nick letters. Hence \( \nu^+(X') = X' \). Because the function \( \nu^+ \) neither introduces, nor removes lower nick letters,

  \[ \#_\Delta(X) = \#_\Delta(X') + |y_1| + \#_\Delta(X_{n+1}), \]

  which can be rewritten as

  \[ |y_1| = \#_\Delta(X) - \#_\Delta(X') - \#_\Delta(X_{n+1}). \]

  We can now make the following derivation:

  \[
  \begin{align*}
  B_\dagger(X) & \leq B_\dagger(\nu^+(X')) + B_\dagger(\nu^+(X_{n+1})) + |y_1| \\
  & = B_\dagger(X') + B_\dagger(\nu^+(X_{n+1})) + \#_\Delta(X) - \#_\Delta(X') - \#_\Delta(X_{n+1})
  \end{align*}
  \]
\[ \begin{align*}
\leq & \ B_t(X') + B_t(X_{n+1}) + \#_\triangle(X) - \#_\triangle(X') - \#_\triangle(X_{n+1}) \\
\leq & \ B_t(X_1) + \cdots + B_t(X_n) + \#_\triangle(X') - \#_\triangle(X_1) - \cdots - \#_\triangle(X_n) \\
& \quad + B_t(X_{n+1}) + \#_\triangle(X) - \#_\triangle(X') - \#_\triangle(X_{n+1}) \\
= & \ B_t(X_1) + \cdots + B_t(X_n) + B_t(X_{n+1}) \\
& \quad + \#_\triangle(X) - \#_\triangle(X_1) - \cdots - \#_\triangle(X_n) - \#_\triangle(X_{n+1}).
\end{align*} \]

Here, the three inequalities follow from Lemma 6.26(1), Lemma 6.16(1) and the induction hypothesis (in particular, inequality (6.2) for \( X = S((\uparrow \varepsilon_1 \ldots \varepsilon_n)) \)), respectively. We have thus obtained inequality (6.3) for \( X = S(E) \).

In a similar way, using the other claims from Lemma 6.26 and Lemma 6.16 and the other equations from the induction hypothesis, we find inequality (6.4) for \( X = S(E) \):

\[ B_\downarrow(X) \geq B_\downarrow(\nu^+(X')) + B_\downarrow(\nu^+(X_{n+1})) - 1 \]

\[ \begin{align*}
\geq & \ B_\downarrow(X') + B_\downarrow(\nu^+(X_{n+1})) - \#_\tri(\nu) - 1 \\
\geq & \ B_\downarrow(X_1) + \cdots + B_\downarrow(X_n) - (n - 1) - \#_\tri(X_1) - \cdots - \#_\tri(X_n) \\
& \quad + B_\downarrow(X_{n+1}) - \#_\tri(X_{n+1}) - 1 \\
= & \ B_\downarrow(X_1) + \cdots + B_\downarrow(X_n) + B_\downarrow(X_{n+1}) - n \\
& \quad - \#_\tri(X_1) - \cdots - \#_\tri(X_n) - \#_\tri(X_{n+1}),
\end{align*} \]

inequality (6.5) for \( X = S(E) \):

\[ B_\downarrow(X) = B_\downarrow(\nu^+(X')) + B_\downarrow(\nu^+(X_{n+1})) \\
= B_\downarrow(X') + B_\downarrow(\nu^+(X_{n+1})) \\
\leq B_\downarrow(X') + B_\downarrow(X_{n+1}) \\
\leq B_\downarrow(X_1) + \cdots + B_\downarrow(X_n) + B_\downarrow(X_{n+1}), \]

and equality (6.6) for \( X = S(E) \):

\[ n_\uparrow(X) = n_\uparrow(\nu^+(X')) + n_\uparrow(\nu^+(X_{n+1})) \\
= n_\uparrow(X') + n_\uparrow(\nu^+(X_{n+1})) \\
= n_\uparrow(X_1) + \cdots + n_\uparrow(X_n) - \#_\tri(X_1) - \cdots - \#_\tri(X_n) \\
& \quad + n_\uparrow(X_{n+1}) - \#_\tri(X_{n+1}) \\
= n_\uparrow(X_1) + \cdots + n_\uparrow(X_n) + n_\uparrow(X_{n+1}) \\
& \quad - \#_\tri(X_1) - \cdots - \#_\tri(X_n) - \#_\tri(X_{n+1}). \]
We conclude that the five equations are also valid for the ↑-expression $E$ with $n + 1$ arguments.

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

3. Let $E$ be an ↑-expression $\langle \uparrow E \rangle$ for a DNA expression $E_1$, let $X_1 = S(E_1)$ and let $r \geq 0$ be the number of single-stranded components of $X_1$. By definition, $X = \kappa(X_1)$.

Now, all equations are just special cases of the ones in Lemma 6.19.

In the proof of Lemma 6.27[1], we observed that if $n = 1$ and $\varepsilon_1$ is a DNA expression $E_1$, equations (6.2)–(6.6) are in fact special cases of the claims from Lemma 6.16. Likewise, the proof of Lemma 6.27[3] consisted mainly of the observation that equations (6.7)–(6.13) are special cases of the claims from Lemma 6.19.

For expressible formal DNA molecules $X$, we can also walk the other direction. We can consider the claims from Lemma 6.16 as special cases of equations (6.2)–(6.6) from Lemma 6.27[1]. If we take $n = 1$ and let $\varepsilon_1$ be a DNA expression denoting $X_1$, then Lemma 6.16 follows from the observation that $\nu^+(X) = \nu^+(S(\varepsilon_1)) = S(\langle \uparrow \varepsilon_1 \rangle)$.

Similarly, if $X$ is an expressible formal DNA molecule and $E_1$ is a DNA expression denoting $X$, then Lemma 6.19 follows from Lemma 6.27[3] and the observation that $\kappa(X) = \kappa(S(E_1)) = S(\langle \downarrow E_1 \rangle)$.

By inequalities (6.7) and (6.9) from Lemma 6.27[3], the values of the functions $B_{\uparrow}$ and $B_{\downarrow}$ do not increase when we apply the operator $\downarrow$ to a DNA expression $E_1$. There exists, however, a much stronger result concerning $B_{\uparrow}$ and $B_{\downarrow}$ for $\downarrow$-expressions:

**Lemma 6.29** Let $E$ be an $\downarrow$-expression, and let $X = S(E)$. Then $B_{\uparrow}(X) + B_{\downarrow}(X) \leq 1$.

**Proof:** Let $E = \langle \downarrow \varepsilon_1 \rangle$, where $\varepsilon_1$ is an $N$-word or a DNA expression. By the definition of the semantics of an $\downarrow$-expression, $X = \kappa(S^+(\varepsilon_1))$.

Hence, $X$ neither contains upper components, nor lower components. Each $\uparrow$-component of $X$ has to be a lower nick letter and each $\downarrow$-component of $X$ has to be an upper nick letter. By Theorem 5.4, $X$ does not both contain lower nick letters and upper nick letters. This implies that $X$ either does not contain any $\uparrow$-component or does not contain any $\downarrow$-component (or both). Then by definition, $B_{\uparrow}(X) = 0$, or $B_{\downarrow}(X) = 0$ (or both). Now, the claim follows from Lemma 6.12[1] and (2).

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 $\downarrow$ in a DNA expression.

**Theorem 6.30** Let $E$ be a DNA expression, and let $X = S(E)$.

1. If $E$ is an $\uparrow$-expression, then
   \[
   \#_{\uparrow}(E) \geq 1 + B_{\downarrow}(X) \quad \text{and} \quad \#_{\uparrow}(E) \geq n_{\uparrow}(X).
   \]

2. If $E$ is a $\downarrow$-expression, then
   \[
   \#_{\uparrow}(E) \geq 1 + B_{\downarrow}(X) \quad \text{and} \quad \#_{\uparrow}(E) \geq n_{\uparrow}(X).
   \]
3. If \( E \) is an \( \downarrow \)-expression, then
\[
\begin{align*}
\#_{\uparrow \downarrow}(E) & \geq B_{\uparrow}(X), \\
\#_{\uparrow \downarrow}(E) & \geq B_{\downarrow}(X) \quad \text{and} \quad \#_{\uparrow}(E) \geq n_{\uparrow}(X).
\end{align*}
\] (6.14) (6.15)

**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 \( \langle \uparrow \downarrow \alpha_1 \rangle \) for an \( \mathcal{N} \)-word \( \alpha_1 \).

  If \( E = \langle \uparrow \alpha_1 \rangle \), then \( \#_{\uparrow \downarrow}(E) = 1 \), \( \#_{\uparrow}(E) = 0 \), \( X = \left( \alpha_1 \right) \) and \( B_{\uparrow}(X) = n_{\uparrow}(X) = 0 \). 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 \( E = \langle \uparrow \downarrow \alpha_1 \rangle \), then \( \#_{\uparrow \downarrow}(E) = 0 \), \( \#_{\uparrow}(E) = 1 \), \( X = \left( \frac{\alpha_1}{c(\alpha_1)} \right) \), \( B_{\uparrow}(X) = B_{\downarrow}(X) = 0 \) and \( n_{\uparrow}(X) = 1 \). Indeed, these values satisfy the inequalities in Claim 3.

- Let \( p \geq 1 \), and suppose that the lower bounds 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 an \( \uparrow \)-expression, or a \( \downarrow \)-expression or an \( \uparrow \downarrow \)-expression. We consider each of these cases separately.

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

    By definition, if an argument \( \varepsilon_i \) is an \( \mathcal{N} \)-word \( \alpha \), then \( \#_{\uparrow \downarrow}(\varepsilon_i) = \#_{\uparrow}(\varepsilon_i) = 0 \), \( X_i = S^+(\varepsilon_i) = \left( \alpha \right) \) and \( B_{\downarrow}(X_i) = n_{\uparrow}(X_i) = 0 \). If, on the other hand, an argument \( \varepsilon_i \) is a DNA expression, then \( X_i = S^+(\varepsilon_i) = S(\varepsilon_i) \). Because such an argument contains at most \( p \) operators, the induction hypothesis provides us with lower bounds for \( \#_{\uparrow \downarrow}(\varepsilon_i) \) and \( \#_{\uparrow}(\varepsilon_i) \). For \( \downarrow \)-expressions \( \varepsilon_i \), we use lower bound (6.15) for \( \#_{\uparrow \downarrow}(\varepsilon_i) \).

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

    \[
    \#_{\uparrow \downarrow}(E) = 1 + \sum_{i=1}^{n} \#_{\uparrow \downarrow}(\varepsilon_i)
    \]

    \[
    = 1 + \sum_{\mathcal{N}-\text{words } \varepsilon_i} \#_{\uparrow \downarrow}(\varepsilon_i) + \sum_{\uparrow \text{-expr. } \varepsilon_i} \#_{\uparrow \downarrow}(\varepsilon_i)
    \]

    \[
    + \sum_{\downarrow \text{-expr. } \varepsilon_i} \#_{\uparrow \downarrow}(\varepsilon_i) + \sum_{\uparrow \downarrow \text{-expr. } \varepsilon_i} \#_{\uparrow \downarrow}(\varepsilon_i)
    \]

    \[
    \geq 1 + \sum_{\mathcal{N}-\text{words } \varepsilon_i} B_{\downarrow}(X_i) + \sum_{\uparrow \text{-expr. } \varepsilon_i} (1 + B_{\downarrow}(X_i))
    \]

    \[
    + \sum_{\downarrow \text{-expr. } \varepsilon_i} (1 + B_{\uparrow}(X_i)) + \sum_{\uparrow \downarrow \text{-expr. } \varepsilon_i} B_{\downarrow}(X_i).
    \]

    Obviously, the term \( 1 + B_{\downarrow}(X_i) \) for an \( \uparrow \)-expression \( \varepsilon_i \) is greater than \( B_{\downarrow}(X_i) \). For the \( \downarrow \)-expressions \( \varepsilon_i \), we use Lemma 6.12 to replace the terms \( 1 + B_{\uparrow}(X_i) \) by \( B_{\downarrow}(X_i) \).
When subsequently, we apply inequality (6.4) from Lemma 6.27(1), we obtain the desired lower bound for $\#_{\uparrow,\downarrow}(E)$:

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

The lower bound for $\#_{\downarrow}(E)$ is easy to calculate. First, we observe that for each argument $\varepsilon_i$, either by definition (if $\varepsilon_i$ is an $N$-word), or by the induction hypothesis (if $\varepsilon_i$ is a DNA expression), $\#_{\downarrow}(\varepsilon_i) \geq n_{\downarrow}(X_i)$. Next, we apply equality (6.6) from Lemma 6.27(1):

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

If $E$ is a $\downarrow$-expression, then the proof is analogous.

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

If $E_1$ is an $\uparrow$-expression, then we additionally apply inequality (6.9) from Lemma 6.27(3) and Lemma 6.12(1):

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

Note that by this series of inequalities, we have proved both inequality (6.14) and inequality (6.15) for the case that $E_1$ is a $\uparrow$-expression. Analogously, if $E_1$ is a $\downarrow$-expression, then

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

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

$$\#_{\uparrow,\downarrow}(E) = \#_{\uparrow,\downarrow}(E_1) \geq B_{\uparrow}(X_1) = B_{\uparrow}(X) \quad \text{and} \quad \#_{\uparrow,\downarrow}(E) = \#_{\uparrow,\downarrow}(E_1) \geq B_{\downarrow}(X_1) = B_{\downarrow}(X).$$

To calculate a lower bound for $\#_{\downarrow}(E)$, 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 (6.12) from Lemma 6.27(3),

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

It is only a small step from the number of operators occurring in a DNA expression to the length of that DNA expression:
Theorem 6.31  Let $E$ be a DNA expression, and let $X = S(E)$.
1. If $E$ is an $\uparrow$-expression, then $|E| \geq 3 + 3 \cdot B_1(X) + 3 \cdot n_\uparrow(X) + |X|_A$.
2. If $E$ is a $\downarrow$-expression, then $|E| \geq 3 + 3 \cdot B_1(X) + 3 \cdot n_\downarrow(X) + |X|_A$.
3. If $E$ is an $\updownarrow$-expression, then
   \begin{align*}
   |E| & \geq 3 \cdot B_1(X) + 3 \cdot n_\uparrow(X) + |X|_A \quad \text{and} \quad \\
   |E| & \geq 3 \cdot B_1(X) + 3 \cdot n_\downarrow(X) + |X|_A. 
   \end{align*}
4. If $E = (\uparrow \alpha_1)$ for an $N$-word $\alpha_1$, then $|E| = 3 \cdot n_\uparrow(X) + |X|_A$.
5. If $E = (\downarrow E_1)$ for a DNA expression $E_1$, then $|E| \geq 3 + 3 \cdot n_\downarrow(X) + |X|_A$.
6. Unless $E = (\updownarrow \alpha_1)$ for an $N$-word $\alpha_1$, $|E| \geq 3 + 3 \cdot n_\uparrow(X) + |X|_A$.

Note that the starting point for this result is a DNA expression $E$. Some formal DNA molecules $X$ can only be denoted by certain types of DNA expressions. If $X$ contains single-stranded components, then $E$ cannot be an $\updownarrow$-expression. If $X$ contains upper (or lower) nick letters, then $E$ cannot be an $\uparrow$-expression ($\downarrow$-expression, respectively). Hence, given a formal DNA molecule $X$, some of the claims from this result may not be applicable.

In Theorem 7.42 and Theorem 7.46, we will see that the lower bounds from Claims 1 and 2 are tight. They are achieved by the shortest $\uparrow$-expressions and $\downarrow$-expressions for a given formal DNA molecule. As we will observe after the statement of Theorem 7.5, the lower bounds from Claim 3 are tight for nick free formal DNA molecules, where $B_1(X) = B_\downarrow(X) = 0$ and $n_\updownarrow(X) = 1$. However, in Theorem 8.19 we will find that these two lower bounds are not tight for expressible formal DNA molecules containing nick letters: there do not exist $\updownarrow$-expressions denoting such molecules that achieve these lower bounds.

**Proof:** Claims 1, 3 follow immediately from Lemma 6.11 and Theorem 6.30.

4. If $E = (\uparrow \alpha_1)$ for an $N$-word $\alpha_1$, then $X = \left(\frac{\alpha_1}{c_1(\alpha_1)}\right)$ and $n_\uparrow(X) = 1$. Hence, both sides of the equality in the claim evaluate to $3 + |\alpha_1|$.

5. Assume that $E = (\downarrow E_1)$ for a DNA expression $E_1$, and let $X_1 = S(E_1)$. By definition, $X = \kappa(X_1)$ and hence $|X|_A = |X_1|_A$. We distinguish three cases.

   If $E_1$ is an $\uparrow$-expression, then by Claim 1 and inequality (6.12) from Lemma 6.27[3],

   \begin{align*}
   |E| & = 3 + |E_1| \\
   & \geq 3 + 3 + 3 \cdot B_1(X_1) + 3 \cdot n_\uparrow(X_1) + |X_1|_A \\
   & \geq 3 + 3 + 0 + 3 \cdot n_\uparrow(X_1) + |X_1|_A \\
   & \geq 3 + 3 \cdot n_\uparrow(X) + |X|_A.
   \end{align*}

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

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

   \begin{align*}
   |E| = 3 + |E_1| \geq 3 + 0 + 3 \cdot n_\uparrow(X_1) + |X_1|_A = 3 + 3 \cdot n_\uparrow(X) + |X|_A.
   \end{align*}

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.