THE COMPLEXITY OF UNAVOIDABLE WORD PATTERNS

PAUL SAUER

University of South Africa
e-mail address: paul.v.sauer@gmail.com

Abstract. The avoidability, or unavoidability of patterns in words over finite alphabets has been studied extensively. The word \( \alpha \) over a finite set \( A \) is said to be unavoidable for an infinite set \( B^+ \) of nonempty words over a finite set \( B \) if, for all but finitely many elements \( w \) of \( B^+ \), there exists a morphism \( \phi : A^+ \to B^+ \) such that \( \phi(\alpha) \) is a factor of \( w \). We present various complexity-related properties of unavoidable words. For words that are unavoidable, we provide an upper bound to the length \( s \) of words that avoid them. In particular, for a pattern \( \alpha \) of length \( n \) over an alphabet of size \( r \), we give a concrete function \( N(n, r) \) such that no word of length \( N(n, r) \) over the alphabet of size \( r \) avoids \( \alpha \).

A natural subsequent question is how many unavoidable words there are. We show that the fraction of words that are unavoidable drops exponentially fast in the length of the word. This allows us to calculate an upper bound on the number of unavoidable patterns for any given finite alphabet.

Subsequently, we investigate computational aspects of unavoidable words. In particular, we exhibit concrete algorithms for determining whether a word is unavoidable. We also prove results on the computational complexity of the problem of determining whether a given word is unavoidable.

1. Introduction

Let \( \mathbb{N} \) denote the nonnegative integers. If \( A \) is a finite set, we write \( A^* \) for the set \( \{a_1a_2\ldots a_n \mid a_i \in A \text{ and } n \in \mathbb{N} \} \) of words over \( A \), while \( A^+ \) is the subset of all nonempty words in \( A^* \). For \( n \in \mathbb{N} \) we symbolize the set of words of length \( n \) over \( A \) by \( A^n \). Here the length of a word is defined in the conventional sense: if \( w \in A^* \) and \( w = a_1a_2\ldots a_n \) with each \( a_i \in A \), then the length \( |w| \) of \( w \) is \( n \). The set \( A \) above is sometimes called an alphabet and its members are called letters. We say that the word \( v = a_1a_2\ldots a_m \) is a factor of the word \( w = b_1b_2\ldots b_n \) if there is an \( i \) such that, for \( 1 \leq j \leq m \), we have \( a_j = b_{i+j} \).

For a word \( w \) and letters \( x_1, x_2, \ldots, x_k \), we denote by \( w^{x_1\cdot x_2\cdot \ldots \cdot x_k} \) the word derived from \( w \) by deleting all occurrences of each of the \( x_i \).

We say that a word \( w \) over a finite alphabet \( B \) reflects a word \( \alpha \) (or a pattern \( \alpha \), for the sake of clarity) over a finite alphabet \( A \) whenever there is a semigroup morphism \( \phi : A^+ \to B^+ \) such that \( \phi(\alpha) \) is a factor of \( w \). The pattern \( \alpha \) is called unavoidable for a set \( X \) of words over a finite alphabet if all but finitely many \( w \in X \) reflect \( \alpha \). The pattern \( \alpha \) is simply called unavoidable if the preceding statement holds for every set over a finite alphabet. Otherwise \( \alpha \) is called avoidable.

Key words and phrases: regularity, avoidability, unavoidability.
The study of combinatorial patterns is one of the most repeated themes in Mathematics [4], [7]. Among these studies, the unavoidability of patterns in words over finite alphabets has been explored extensively. Over the last century, this theme has resurfaced repeatedly [12], [8], [1], [13], [9], [11]. In the last decade, there has been a resurgence in the investigation of unavoidability [10]. Thue [12] proved that \(xxx\) is avoidable on the binary alphabet and \(xx\) is avoidable on the alphabet of size 3. Bean et al. [1] conducted an extensive investigation into the avoidability of patterns. One central discovery of this investigation is the notion of a letter that is free for a pattern.

\[\text{Definition 1.1.} \] Let \(A\) be a finite alphabet and let \(w \in A^+\). A letter \(x \in A\) is free for \(w\) if \(x\) occurs in \(w\) and there is no \(n \in \mathbb{N}\) and \(a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n\) such that
\[xa_1 \quad b_1a_1 \quad b_1a_2 \quad b_2a_2 \quad \ldots \quad b_nx\]
are all factors of \(w\).

Free letters are connected to the phenomenon of unavoidability by the following lemma, whose proof appears in [1].

\[\text{Lemma 1.2.} \] Suppose \(\alpha\) is a pattern with a free letter \(x\). If \(\alpha x\) is unavoidable, then so \(\alpha\).

A surprising, complete characterization of unavoidable patterns follows from Lemma 1.2.

\[\text{Theorem 1.3.} \] A pattern \(\alpha\) is unavoidable if and only if it is reducible to the empty word by iteratively performing one of the following operations on the pattern:

1. deleting every occurrence of a free letter, or
2. replacing all occurrences of some letter \(x\) occurring in \(\alpha\) by a different letter \(y\), also occurring in \(\alpha\).

We refer to the second operation as the identification of letters. The proof of Theorem 1.3, presented in [1], is not constructive. Therefore it gives no indication, for any given pattern, what the longest word avoiding that pattern might be. Subsequent to [1], one constructive unavoidability result was established, pertaining to the subset of patterns that represent permutations. We now discuss this result briefly.

Let \([n]\) denote the set \(\{1, 2, \ldots, n\}\) and let \(S_n\) be the set of all permutations of \([n]\). We use one-line notation to express a permutation \(\pi \in S_n\) – that is we write \(x_1x_2\ldots x_n\) when \(\pi(i) = x_i\) for \(i \in [n]\). The write \(\langle \pi \rangle\) for the word \(12\ldots nx_0x_1\ldots x_n\), where \(x_0\) is a symbol not in \([n]\). Fouché [5] discovered the following

\[\text{Theorem 1.4.} \] For \(n, r \in \mathbb{N}\) there is an \(N(n, r) \in \mathbb{N}\) such that every \(w \in [r]^N\) reflects every \(\langle \pi \rangle\), where \(\pi \in S_n\). Specifically, the numbers \(N(n, r)\) are inductively bounded from above by
\[N = N(n + 1, r + 1) \leq 2(n + 1)N(n + 1, r)N(n, (2n + 2)^2r^{N(n+1,r)})\]
In the sequel, we show that a similar bound holds for all unavoidable patterns. The proof of the Main Theorem 2.4 follows Fouché’s reasoning. Subsequent sections are organized as follows:

In Section 3, we investigate the density of unavoidable patterns in the space of all patterns. We establish that this density drops quite fast as the length of the pattern increases. This fact then provides a way to calculate an upper bound for the number of unavoidable patterns as function of the size of the underlying alphabet.

Section 4 is devoted to the algorithmic decision problem of whether a letter appearing in a given pattern is free. We present a concrete algorithm running in polynomial time. In Section 5, we show that there is a simple reduction from boolean formulas to patterns that maps satisfiable formulas to unavoidable patterns and unsatisfiable formulas to avoidable patterns. The final substantial part of the paper is Section 6, where we prove that the the problem of deciding whether a pattern is unavoidable is NP-complete.

2. General Bounds for Unavoidable Patterns

The main result of this section is Theorem 2.4, which provides an upper bound on the length of words that can avoid a given, unavoidable pattern. In order to establish Theorem 2.4, we first need to establish a few facts. Lemma 2.1 below gives us a method for building morphisms as the size of our alphabet increases, provided that there is a free letter in the pattern. This lemma, stated here without proof, is proved in [1].

Lemma 2.1. Let $A$ and $B$ be a finite alphabets and let $w$ be a word over $A$. Suppose $x$ is free for $w$. If there is a morphism $\phi : w^x \mapsto v$, where $v \in B^+$ is of the form $a_{i1}X_1a_{i2}X_2\ldots a_{it}X_ta_{i+1}$, each $X_i$ being a word over $B \setminus \{a\}$, then there is a morphism $\psi : w \mapsto v$.

Next, we strengthen Theorem 1.3 slightly. This makes the proof of the subsequent lemma easier. In addition, the theorem simplifies the characterization of unavoidable patterns in [1] in that eliminates the need for identifying letters as a required operation. It may therefore be of independent interest. See also [13]

Theorem 2.2. A pattern is unavoidable if and only if it is reducible to the empty word by iteratively deleting free letters.

Proof. Let $n > 0$ and suppose $\alpha \in [r]^n$ is reducible to the empty word by iteratively deleting free letters. Assume without loss of generality that every letter in $[r]$ appears in $\alpha$. We have that there is an ordering $a_1, a_2, \ldots, a_r$ of $[r]$ such that $a_1$ is free for $\alpha$ and, for each $i < r$, we have that $a_{i+1}$ is free for $\alpha_{a_1,a_2,\ldots,a_i}$. Since $\alpha_{a_1,a_2,\ldots,a_r}$ is the empty word, we know that $\alpha_{a_1,a_2,\ldots,a_i}$ is unavoidable, by Theorem 1.3. Now we can proceed inductively: If $\alpha_{a_1,a_2,\ldots,a_{i+1}}$ is free for some $i > 1$, then Lemma 1.2 yields that $\alpha_{a_1,a_2,\ldots,a_i}$ is unavoidable. Consequently $\alpha_{a_1}$ is unavoidable. Applying Lemma 1.2 one more time, we get that $\alpha$ itself is unavoidable, as desired.

Now suppose $\alpha \in [r]^n$ is unavoidable. By Theorem 1.3, there is a sequence of operations $f_1, f_2, \ldots, f_k$ such that each $f_i : [r]^* \rightarrow [r]^*$ and either

1. deletes every occurrence some free letter, or
2. replaces all occurrences of some letter $x$ occurring in $\alpha$ by a different letter $y$, also occurring in $\alpha$ and, furthermore,
the composition of operations $F_1 = f_1 \circ f_2 \circ \cdots \circ f_k$ applied to $\alpha$ is the empty string. Now let $f_{i_1}, f_{i_2}, \ldots, f_{i_t}$ be the subsequence of operations of type (1) above. We claim that $f_{i_1} \circ f_{i_2} \circ \cdots \circ f_{i_t}(\alpha)$ is also the empty string. For if we suppose otherwise, then there is an $i_m$ such that $F_2(\alpha) = f_{i_1} \circ f_{i_2} \circ \cdots \circ f_{i_m}(\alpha)$ has no free letter. Now consider the $F_3 = f_1 \circ f_2 \circ \cdots \circ f_{i_m} \circ f_{i_m+1} \circ f_{i_m+2} \circ \cdots \circ f_{i_m+j}$, the first $i_m+j$ operations of $F_1$, where each operation among $f_{i_m+1}, f_{i_m+2}, \ldots, f_{i_m+j}$ is of type (2) above and $f_{i_m+j+1}$ is of type (1). We know $F_3(\alpha)$ has a free letter $x$, otherwise the operation $f_{i_m+j+1}$ of $F_1$ would be impossible. Now, since $x$ is not free for $F_2(\alpha)$, we can infer that there is an $s \in \mathbb{N}$ and $a_1, a_2, \ldots, a_s, b_1, b_2, \ldots, b_s \in [r]$ such that

$$xa_1$$
$$b_1a_1$$
$$b_1a_2$$
$$b_2a_2$$
$$\vdots$$
$$b_sx$$

are all factors of $F_2(\alpha)$. But since $F_3$ differs from $F_2$ only in the addition of operations that identify letters, we know that $F_3(\alpha)$ is identical to $F_2(\alpha)$, except for the fact that certain letters that are distinct in $F_2(\alpha)$ may be identical in $F_3(\alpha)$. Therefore there is also a sequence of factors $a_1', a_2', \ldots, a_s', b_1', b_2', \ldots, b_s' \in [r]$ such that

$$xa_1'$$
$$b_1'a_1'$$
$$b_1'a_2'$$
$$b_2'a_2'$$
$$\vdots$$
$$b_s'x$$

are all factors of $F_3(\alpha)$. But then we have that $x$ is not free for $F_3(\alpha)$, a contradiction. This completes the proof.

The following Corollary follows immediately from Theorem 2.2. It shows that an unavoidable pattern always has a free letter. This will be used in conjunction with Lemma 2.1 to build morphisms in the proof of the main result.

**Corollary 2.3.** Every unavoidable pattern has a free letter.

We are now ready to prove our main result. The construction of the proof closely follows [5].

**Main Theorem 2.4.** For $n, r \in \mathbb{N}$ there is an $N(n, r) \in \mathbb{N}$ such that every $w \in [r]^N$ reflects every unavoidable pattern of length $n$ over $[r]$. The minimal values for the numbers $N(n, r)$ are bounded from above by

$$N(n + 1, r + 1) \leq (n + 1)N(n + 1, r)N(n, (n + 1)^2r^{N(n+1,r)})$$
A natural subsequent question is how many unavoidable words there are. We start by claim that every element of a morphism \( L \) as well as for \( n \) and all \( r \), as well as for \( n + 1 \) and some \( r \geq 1 \).

Let \( w \) be a word of length \((n + 1)KL\) over an alphabet \( A \) of size \( r + 1 \), where \( K = N(n + 1, r) \) and \( L = N(n, (n + 1)^2rN(n + 1, r)) \). We may assume that every factor of length \( K \) in \( w \) contains every letter in \( A \), for otherwise \( w \) reflects every unavoidable pattern of length \( n + 1 \), by our inductive hypothesis. Consequently, the word \( w \) is of the form \( a^iX_1a^{i_2}X_2\ldots a^{i_L}X_La^{i_l+1} \), where each \( X_i \in \{ A \setminus \{ a \} \}^+ \) satisfies \( |X_i| < K \). We may assume that \( 1 \leq a_{ij} \leq n \), for otherwise the morphism \( f(x) = a \) that sends every letter to \( a \) shows that every pattern of length \( n + 1 \) is reflected by \( w \).

We immediately have

\[
(n + 1)KL = |w| \leq (K - 1)t + (t + 1)(n + 1) \\
= (K + n)t + n + 1 \\
\leq (n + 1)Kn + 1
\]

since \( K > 2 \) is readily available from the definition of \( K \). Therefore we have \( t > L \) and hence \( w \) has a factor \( v = a^iX_1a^{i_2}X_2\ldots a^{i_L}X_La^{i_l+1} \), where each \( X_i \in \{ A \setminus \{ a \} \}^+ \) satisfies \( |X_i| < K \).

Define the alphabet \( B \) as the set of words of the form \( a^iX \), with \( 1 \leq i \leq n \) and \( X \in \{ A \setminus \{ a \} \}^+ \) satisfies \( |X| < K \).

\[
|B| = n(r + r^2 + \cdots + r^{K-1}) \\
\leq (n + 1)^2r^{K-1}
\]

since \( K \geq n + 1 \) for every \( n \) and \( r \).

We have \( v \) is a word of length \( L \) over \( B \). Suppose that \( \alpha \) is any unavoidable pattern of length \( n + 1 \) over \( A \). Using Corollary 2.3 there is a letter \( x \in A \) that is free for \( \alpha \). We remind ourselves that \( L = N(n, (n + 1)^2rN(n + 1, r)) \) and note, by our inductive hypothesis, that there is thus a morphism \( f : \alpha^x \mapsto v \). Consequently, Lemma 2.1 yields that there is a morphism \( \psi : \alpha \mapsto v \) and the proof is complete.

\[ \square \]

3. Density and Counting Unavoidable Patterns

A natural subsequent question is how many unavoidable words there are. We start by showing that, for alphabets of 3 or more letters, the fraction of words that are unavoidable drops exponentially fast in the length of the word.

**Lemma 3.1.** Let \( r > 2 \) and \( n > 0 \). Let \( p_{r,n} \) be the probability that a pattern of length \( n \) is unavoidable over \([r]\). We have \( p_{r,n} \leq \left( \frac{r - 1}{r} \right)^{n-1} \).

**Proof.** Let \( w \) be a word of length \( n \) over \( r \). If \( n = 1 \) then \( w \) is unavoidable, so that our claim holds with \( p_{r,1} = \left( \frac{r - 1}{r} \right)^0 = 1 \). Now suppose \( n > 1 \). We will use the fact that \( xx \) is avoidable, established in [12]. Let \( V = \{ w \in [r]^* : x \in [r] \text{ and } xx \text{ is a factor of } w \} \). First we claim that every element of \( V \) is avoidable. To prove our claim, we start by noting that \( x \) is not free for any \( v \in [r]^* \) that has \( xx \) as a factor. Hence any sequence of deletions
of free letters applied to $w$ results in a word that has $xx$ as a factor. Using Theorem 1.3, our claim is proved. Let $U_{n,r}$ be the set of all unavoidable words of length $n$ over $r$. By our claim above, we have $U \subseteq \bar{V} = [r]^n \setminus V$. Now we count the elements of $\bar{V}$. Let $w = w_1w_2\ldots w_n$ be an abstract word of length $n$ over $r$. For $w_1$ we can choose any one of the $r$ letters in $[r]$. For each subsequent $w_i$, we can choose any letter from $[r]$, other than our choice of $w_{i-1}$. Hence $|\bar{V}| = r(r-1)^{n-1}$. It follows that $|U| \leq r(r-1)^{n-1}$ and therefore $pr,n \leq \frac{r(r-1)^{n-1}}{r^n} = \left(\frac{r-1}{r}\right)^{n-1}.$

We also know from [1] that all patterns over $[r]$ have length less than $2^n$. Combined with Lemma 3.1 above, we can now obtain an upper bound on the number of unavoidable patterns over $[r]$.

**Proposition 3.2.** The number of unavoidable patterns over $[r]$ is at most $r \left(\frac{(r-1)^{2^r-1} - 1}{r-2}\right)$.

**Proof.** The number of patterns of length $n$ is bounded from above by

$$pr,n r^n \leq \left(\frac{r-1}{r}\right)^{n-1} r^n = r(r-1)^{n-1}.$$ 

Since there are no unavoidable patterns of length greater than $2^n - 1$ we have the total number of unavoidable patterns is at most

$$\sum_{i=1}^{2^r-1} r(r-1)^{i-1} = \sum_{i=0}^{2^r-2} (r-1)^i = r \left(\frac{(r-1)^{2^r-1} - 1}{r-2}\right)$$

and the proof is complete.

---

4. **Free Letters and Computation**

We now proceed to investigate the computational aspects of unavoidability, assuming a basic familiarity with algorithms and computational complexity, for which Hopcroft and Ullman [6] and [2] provide authoritative references.

For a pattern $\alpha$ we construct a directed bipartite graph $G_\alpha$, which we call the graph of $\alpha$. The vertex set $V(G_\alpha)$ of $G_\alpha$ has two nodes $0ab$ and $1ab$ for each 2-factor $ab$ of $\alpha$. The pair of 2-factors $(0ab,1cd)$ of $\alpha$ is an edge of $G_\alpha$ whenever $b = d$. Similarly, the pair $(1ab,0cd)$ of $\alpha$ is an edge of $G_\alpha$ whenever $a = c$. The reason why we create two vertices for each 2-factor is to prevent paths of the form $xa, xb, xc$.

**Lemma 4.1.** Let $\alpha$ be a pattern. A letter $x$ of $\alpha$ is not free if and only if there is a path in $G_\alpha$ from a node having $x$ as its first component to a node having $x$ as its second component.
Proof. If $x$ is not free for $\alpha$, then there is an $n \in \mathbb{N}$ and $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ such that

\begin{align*}
xa_1 \\
b_1a_1 \\
b_1a_2 \\
b_2a_2 \\
\vdots \\
b_na
\end{align*}

are all factors of $w$. It is clear from the definition of $G_\alpha$ that the edges

\begin{align*}
(0xa_1, 1b_1a_1) \\
(1b_1a_1, 0b_1a_2) \\
(0b_1a_2, 1b_2a_2) \\
(1b_2a_2, 0b_2a_3) \\
\vdots \\
(0b_na_{n-1}, 1b_nx)
\end{align*}

all exist in $G_\alpha$. Therefore a path from $0xa_1$ to $1b_nx$ exists in $G_\alpha$, as desired.

Proving the converse is essentially the same as reading the construction above in reverse.

Given Lemma 4.1, we can easily construct an efficient algorithm that decides, given a pattern $\alpha$ and a letter $x$ appearing in $\alpha$, whether $x$ is free for $\alpha$.

Firstly, the construction of the adjacency matrix of $G_\alpha$ from $\alpha$ can be defined as follows:

\begin{verbatim}
BUILD_G(n, alpha[])
    initialize G[0:n-1][0:n-1] = 0
    initialize V[0:n-1][0:1][0:1] = 0
    for i = 0 to n - 1
        V[i][0] = V[i][1] = alpha[i:i+1]
    for i = 0 to n - 1
        for j = 0 to n - 1
            if V[i][0][0] = V[j][1][0]
                G[j][i] = 1
            if V[i][0][1] = V[j][1][1]
                G[i][j] = 1
    return (V, G)
\end{verbatim}
We notice that the runtime is dominated by the nested for loop and therefore requires $O(n^2)$ computational steps.

Now, given a graph $G = G_\alpha$ and a letter $x$, we can use a standard depth-first search algorithm to detect if $x$ is not free.

For simplicity we write a standard depth-first search subroutine.

```python
DFS(G[], V[], i, p, x, is_seen[])
   is_seen[i][p] = True
   if p = 0:
       q = 1
   else:
       q = 0
   for j = 0 to n - 1:
       if not is_seen[j][q] and G[i][j] = 1:
           if V[j][q][1] = x:
               return True
           else:
               return DFS(G, V, j, q, x, is_seen)
   return False
```

We are now ready to write the subroutine determining if $x$ is free for $\alpha$.

```python
IS_FREE(n, alpha[], x)
   (V, G) = BUILD_G(n, alpha[])
   initialize is_seen[0:n-1][0:1] = False
   for i = 1 to n - 1:
       if V[i][0] = x:
           if not is_seen[i][0]:
               return DFS(G, V, i, 0, x, is_seen)
   return False
```

The subroutine IS_FREE requires $O(n^2)$ computational steps, where $n = |\alpha|$: We already know that BUILD_G is $O(n^2)$. In subsequent steps, DFS is called at most $n$ times since every vertex is marked as seen subsequent to the invocation of DFS. At every invocation of DFS, at most $n$ neighbors of a vertex are examined.

Let us pause for a moment to remember where we started and what we have seen along the way. Our initial definition of unavoidability sounds distinctly non-finitary: A pattern must be reflected by all but finitely many elements for every set over any finite alphabet. Theorem 2.2 then gives us a finitary characterization of unavoidability in that we only need to look for a sequence of deletions of free letters. Most recently we have seen, in addition, that the problem of deciding whether a letter is free falls in Polynomial Time. It is hence starting to look as though the problem of determining whether a pattern is free might fall
in \( NP \): We can nondeterministically guess the sequence of deletions and verify the validity of the guess (each deletion being of a free letter) in polynomial time. We may also ask how hard this problem is, relative to other problems in \( NP \). In the following two sections, we explore this.

5. Unavoidability and Logic

We work to establish a natural correspondence between boolean formulas and patterns. In particular, we show that given a boolean formula, we can construct a word whose unavoidability coincides with the satisfiability of the formula. We will restrict our construction to 3-CNF boolean formulas, as the correspondence between this subset of boolean formulas and the set of all boolean formulas is well-understood (see [6]).

Let \( \phi \) be any 3-CNF boolean formula. We construct \( \alpha_\phi \), the word of \( \phi \), as follows:

1. The set \( X_{\alpha_\phi} = \{ x_i, \overline{x}_i : i \leq n \} \)
2. The set \( Y_{\alpha_\phi} = \{ a_j, b_j, c_j : j < m \} \)
3. The letter \( d \)
4. The set \( Z = \{ z_i : i \leq M \} \). We choose \( M \) to be sufficiently large so that every element of this set will appear exactly once in \( \alpha_\phi \).

The elements of \( Z \) above are used as "separator" letters to prevent unfortunate 2-factors from occurring. We adopt the convention that we will use each letter in \( Z \) once and denote each occurrence of a letter from \( Z \) in \( \alpha_\phi \) by \( z_+ \). We denote the set of letters itemized above by \( A_\alpha \).

For each variable \( x_i \), we create the factors

\[
d x_i \overline{x}_i d z_+
\]

For each clause \( C_j \) in \( \phi \) we construct a factor \( \delta_j \) as the concatenation of the following factors.

If \( x_i \) is the first literal in \( C_j \), we add the following factors to \( \alpha \):

\[
\begin{align*}
  a_j x_i z_+ \\
  b_j x_i z_+ \\
  b_j a_j z_+ \\
  a_j d z_+
\end{align*}
\]
If \( x_i \) is the first literal in \( C_j \), we add the following factors to \( \alpha \):
\[
\begin{align*}
  a_j b_j z_+ \\
  x_i b_j z_+ \\
  x_i a_j z_+ \\
  d a_j z_+
\end{align*}
\]

If \( x_i \) is the second literal in \( C_j \), we add the following factors to \( \alpha \):
\[
\begin{align*}
  b_j x_i z_+ \\
  c_j x_i z_+ \\
  c_j b_j z_+
\end{align*}
\]

If \( \overline{x}_i \) is the second literal in \( C_j \), we add the following factors to \( \alpha \):
\[
\begin{align*}
  b_j c_j z_+ \\
  \overline{x}_j c_j z_+ \\
  \overline{x}_j b_j z_+
\end{align*}
\]

If \( x_i \) is the third literal in \( C_j \), we add the following factors to \( \alpha \):
\[
\begin{align*}
  c_j x_i z_+ \\
  a_j x_i z_+ \\
  a_j c_j z_+
\end{align*}
\]

If \( \overline{x}_i \) is the third literal in \( C_j \), we add the following factors to \( \alpha \):
\[
\begin{align*}
  c_j a_j z_+ \\
  \overline{x}_j a_j z_+ \\
  \overline{x}_j c_j z_+
\end{align*}
\]

We define the word \( \alpha_{\phi} \) of \( \phi \) as the culmination of the above construction.

**Lemma 5.1.** Let \( \phi = C_1 \land C_2 \land \cdots \land C_m \) be a 3-CNF boolean formula. Let \( B \subset A_\alpha \) be such that, if \( p_i \) is a literal in \( C_j \), then the letter \( p_i \) is not in \( B \). No letter in \( \{a_j, b_j, c_j\} \) is free for \( \alpha_{\phi}^B \).

**Proof.** We know that every clause \( C_j \) contains three literals. If \( x_i \) is the first literal in \( C_j \), then the path
\[
\begin{align*}
  a_j x_i \\
  b_j x_i \\
  b_j a_j
\end{align*}
\]
shows that $a_j$ is not free. Similarly, if $\pi_i$ is the first literal in $C_j$, then the path
\[
\begin{align*}
& a_j b_j \\
& \pi_i b_j \\
& \pi_i a_j
\end{align*}
\]
makes $a_j$ not free. The argument showing that $b_j$ and $c_j$ are not free is substantially similar.

Lemma 5.2. Let $B \subseteq A_\alpha$.
(1) If there are $i$ and $j$ such that neither $x_i$ nor $a_j$ is in $B$ and $x_i$ appears as the first literal in the clause $C_j$, then the letter $d$ is not free for $\alpha_B^\phi$.
(2) If there are $i$ and $j$ such that neither $x_i$ nor $a_j$ is in $B$ and $x_i$ appears as the first literal in the clause $C_j$, then the letter $d$ is not free for $\alpha_B^\phi$.

Proof. If $x_i$ is in $C_j$, then the path
\[
\begin{align*}
& dx_i \\
& a_j x_i \\
& a_j d
\end{align*}
\]
ensures that $d$ is not free. Similarly, if $\pi_i$ is in $C_j$, then the path
\[
\begin{align*}
& da_j \\
& \pi_i a_j \\
& \pi_i d
\end{align*}
\]
makes $d$ is not free.

Lemma 5.3. Let $B \subseteq A_\alpha \setminus \{d\}$. If there is an $i$ such that both $x_i$ and $\pi_i$ are in $B$, then $\alpha_B^\phi$ is avoidable.

Proof. If $x_i$ and $\pi_i$ are both in $B$, then $dx_i \pi_i d_B = dd$ is a factor of $\alpha_B^\phi$.

Lemma 5.4. Let $w$ be a word of the form $z_+ z_+ \ldots z_+$. Every letter in $w$ is free.

Proof. Each of the letters $z_+$ appears at most once in $w$.

Lemma 5.5. Fix $k < n$. Let $S_k = \{p_1, p_2, \ldots, p_k\}$, where for each $i$, either $p_i = x_i$ or $p_i = \pi_i$. Both $x_{k+1}$ and $\pi_{k+1}$ are free for $\alpha_{S_k}^\phi$.

Proof. We proceed by induction on $k$. For $k = 0$, we have $S_k = \emptyset$. Hence, the only 2-factor (excluding those containing $z_+$) that contains $x_1$ as the first letter is $x_1 \pi_1$ and the only 2-factor that contains $\pi_1$ as the second letter is $x_1 \pi_1$. So the only path starting at a 2-factor having $x_1$ as the first first letter is the one-cycle from $x_1 \pi_1$ to itself. Our base case has thus been established.

Now suppose the lemma holds for some $k$. Again, the only 2-factor containing $x_k$ as the first letter is $x_k \pi_k$ and the only 2-factor that contains $\pi_k$ as the second letter is $x_k \pi_k$. The lemma immediately follows.
Lemma 5.6. Let $\phi$ be a 3-CNF boolean formula and let $\alpha = \alpha_{\phi}$ be the word of $\phi$. If $\alpha$ is unavoidable, then there is a deletion sequence of free letters that starts by deleting either $x_i$ or $\overline{x}_i$, for $i \leq n$.

Proof. Suppose $\alpha$ is unavoidable. By Theorem 2.2 there is a deletion sequence reducing $\alpha$ to the empty word. We know from Lemma 5.5 that it is possible to delete $x_i$ or $\overline{x}_i$ as the $i$th letter in a deletion sequence of free letters. We need to establish that there is a continuation of this sequence that reduces $\alpha$ to the empty word.

It suffices to show that we can always invert the deletion order when an $x \in X_\alpha$ is deleted immediately after some letter that is not in $X_\alpha$.

Let $B_0 = \emptyset$ and, for $i \leq |A_\alpha|$, let $B_i$ be the first $i$ letters that are deleted. Suppose $x_i$ is the $(k+1)$th letter to be deleted and, furthermore, $\overline{x}_i$ is deleted later on in the deletion sequence. Suppose $a_j$ is the $k$th letter that is deleted. Since $a_j$ is free for $\alpha^{B_{k-1}}$, we have that $b_j \in B_{k-1}$ or $p_j \in B_{k-1}$ where $p_j \in X_\alpha$ corresponds to the first literal in $C_j$, for otherwise there would be a path

$$a_jx_i$$

$$b_jx_i$$

$$b_ia_j$$

or a path

$$a_jb_j$$

$$\overline{x}_ib_j$$

$$\overline{x}_ia_j$$

that with contradict the fact that $a_j$ is free.

Case 1. $p_j = x_i \in B_{k-1}$ for some $x_i \in X_\alpha$. The only possible two-factors containing $a_j$ are $a_jz_+`b_ia_j$ and $a_jd$. Since every occurrence of $z_+$ in $\alpha$ is unique, there is no path from $a_j$ to itself containing with $a_jz_+$. So any path from $a_j$ to itself must start with $a_jd$ and end in $b_ia_j$. Now let us suppose that $a_j$ is not free for $\alpha^{B_{k-1}x_i}$, the word derived by deleting $x_i$ from $\alpha^{B_{k-1}}$. Since the only new 2-factor created by the deletion of $x_i$ (other than two-factors containing a $z_+$) is $d\overline{x}_i$, we conclude that the path from $a_jd$ to $b_ia_j$ goes through $d\overline{x}_i$. But this is the only factor having $\overline{x}_i$ as its second component. So the path from $d\overline{x}_i$ to $b_ia_j$ has $dx$ as the second vertex, for some $x \in A$. But then the vertex immediately preceding $d\overline{x}_i$ has $\overline{x}_i$ as its second component. Consequently this preceding vertex has no predecessor a contradiction. So we have that $a_j$ is free for $\alpha^{B_{k-1}x_i}$ and we can interchange the deletion order of $a_j$ and $x_i$, as desired.

Case 2. $p_j = \overline{x}_i \in B_{k-1}$ for some $x_i \in X_\alpha$. This is symmetric to Case 1.

Case 3. $b_j \in B_{k-1}$.

The only possible two-factors containing $a_j$ (excluding factors with $z_+$ as a component) are $a_jx_j$ and $a_jd$. Since $a_j$ appears as the first component in both cases, we immediately have that $a_j$ is free. Once again, we can interchange the deletion order of $a_j$ and $x_i$.

Repeating cases (1) – (3) with $a_j$ replaced by $b_j$ or $c_j$ is significantly simpler due to the absence of the 2-factors $b_jd$ and $cjd$.

An essentially similar argument holds for $\overline{x}_i$ being the $(k+1)$th letter to be deleted, instead of $x_i$. 

Next, let us suppose, as before, that $x_i$ is the $(k + 1)$th letter to be deleted and $\overline{x}_i$ is deleted later on in the deletion sequence. But now, instead of $a_j$, let us consider the possibility that $d$ is the $k$th letter that is deleted. If $x_i$ cannot be deleted before $d$, it means that the deletion of $x_i$ makes $d$ not free. Since the only new 2-factor (again ignoring the $z_+$) created by the deletion of $x_i$ is $d\overline{x}_i$ and this is the only 2-factor that has $\overline{x}_i$ as its second component, we again reach the same conclusion as Case 1 above: The deletion order of $d$ and $x_i$ are interchangeable.

The same argument carries through symmetrically for $d$ and $\overline{x}_i$.

Finally, we have to show that it is safe to delete the letters $z_+$ last. By Lemma 5.4, we are done.

**Proposition 5.7.** If $\phi$ is a 3-CNF boolean formula and $\alpha = \alpha_\phi$ is the word of $\phi$, then $\phi$ is satisfiable if and only of $\alpha$ is unavoidable.

**Proof.** Suppose $\phi$ with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$ is satisfiable. Let $x_1 = e_1, x_2 = e_2, \ldots, x_n = e_n$, with each $e_i \in \{0, 1\}$, be a satisfying assignment for $\phi$. We show that $\alpha_\phi$ will reduce to the empty set by deleting all its letters in the following stages:

1. For $i \leq n$, delete $x_i$ if $e_i = 1$, otherwise delete $\overline{x}_i$.
2. Next, for $j \leq m$, delete $a_j$, $b_j$ and $c_j$. For each $j$ the order in which $a_j$, $b_j$ and $c_j$ is deleted may differ.
3. Delete the letter $d$.
4. Delete the remaining $x_i$ and $\overline{x}_i$.
5. Delete the remaining characters $z_+$ in any order.

Furthermore, every letter that is deleted will be free at the stage when the deletion happens.

Lemma 5.5 guarantees that every deletion in Stage (1) above is of a free letter. Since $\phi$ is satisfiable, every clause $C_j = (p_1 \lor p_2 \lor p_3)$ has at least one literal that is set to 1. If $p_1 = x_i = 1$, then $x_i$ is deleted in Stage (1). Consequently $a_j$ is free after Stage (1) and can be deleted in Stage (2). The deletion of $a_j$, in turn, causes $c_j$ to be free and the deletion of $c_j$ frees $b_j$. The remaining cases among $p_k = x_i = 1$ and $p_k = \overline{x}_i = 0$ lead to $a_j$, $b_j$ and $c_j$ being deleted in a similar fashion. We can therefore successfully complete the deletions in Stage (2).

After the completion of Stage (2) the only 2-factors (once again ignoring the $z_+$) containing $d$, are of the form $dp_i$ and $p_id$, where for each $i$ we have either $p_i = x_i$ or $p_i = \overline{x}_i$. Furthermore, for each $i$ the same 2-factors are the only ones containing $p_i$. Therefore $d$ is free and consequently Stage (3) can be completed.

After the completion of Stage (3), there are no 2-factors left that do not contain one of the $z_+$. Since every letter $z_+$ is unique, we can safely complete Stage (4). Now all that remains is letters of the form $z_+$ and hence, using Lemma 5.4, we can delete the remaining letters. It follows, by Theorem 2.2, that $\alpha_\phi$ is unavoidable, as desired.

Now suppose $\phi$ is unsatisfiable. For contradiction, suppose $\alpha_\phi$ is unavoidable. Using Lemma 5.6, we may assume that the first $n$ deletions are $p_1, p_2, \ldots, p_n$ with, for every $i$, either $p_i = x_i$ or $p_i = \overline{x}_i$. Define the following assignment on $\phi$: If $p_i = x_i$, then set the variable $x_i$ to 1, otherwise set $x_i$ to 0. Since $\phi$ is not satisfiable, we know that there is some clause $C_j$ that is not satisfied by our chosen assignment. But this means that none of the $p_i$ in the first $n$ deletions appear in $C_j$ and consequently none of the letters $a_j$, $b_j$, $c_j$ are free after the first $n$ deletions, by Lemma 5.1. In addition, by Lemma 5.2, we have that $d$ is not free. In order to free any of these letters, we have to delete at least one letter $x_i$ or $\overline{x}_i$.
which has, thus far not been deleted. But this means, for some \( i \), both \( x_i \) and \( \overline{x}_i \) has been deleted. Using Lemma 5.3, we have a contradiction.

6. UNAVOIDABILITY AND COMPUTATIONAL COMPLEXITY

We define the Word Unavoidability Problem as follows: Given a pattern \( \alpha \) over a finite alphabet, determine if \( \alpha \) is unavoidable. We refer to the set of unavoidable patterns as \( WU \).

**Theorem 6.1.** The Word Unavoidability Problem is \( NP \)-complete.

**Proof.** We note that, given a 3-CNF boolean formula \( \phi \), the construction of the word \( \alpha_\phi \) of \( \phi \) requires a number of computational steps that is linear in the length of \( \phi \): For every variable \( x_i \), we need to add a factor \( dx_i \overline{x}_i d \). For every clause we need to add a constant number of factors that are derived purely from the literals in that clause.

Proposition 5.7 therefore leaves us very little work to do. All that remains is to prove \( WU \in NP \).

Using Theorem 2.2 and the algorithm \( IS\_FREE \) above, we write the following test for unavoidability:

```python
IS_UNAVOIDABLE(alpha):
    A[] <- determine the distinct letters in alpha
    n = |A|
    m = n
    nondeterministically guess the permutation \( \pi \) on \([n]\)
    for \( i = 1 \) to \( n \):
        if IS_FREE(m, alpha, A[\pi(i)]):
            alpha = delete A[\pi(i)] from alpha
        else:
            nondeterministic guess dies
    return True
return False
```

The subroutine \( IS\_FREE \) returns \( True \) when the branch of nondeterminism does not die and thus completes \( |A_\alpha| \) deletions. That is equivalent to saying that the subroutine has found a deletion sequence of free letters. Since \( IS\_FREE \) runs in polynomial time, so does each branch of \( IS\_UNAVOIDABLE \). The number of branches of nondeterminism is bounded from above by the number of permutations on \( n \) distinct letters, which in turn is bounded from above by the number of permutations on the input \( \alpha \).
7. Conclusion

Many interesting questions remain regarding the complexity of unavoidable patterns [3]. The bounds established in Theorem 2.4 above are not primitive recursive. We do not know if there is a primitive recursive upper bound, nor do we know what lower bounds exist, for any significantly general subset of patterns.

8. Acknowledgements

This article has been written in partial fulfillment of the requirements for the degree Doctor of Philosophy in Operations Research at the University of South Africa. Special and sincere thanks go to Willem Fouché and Petrus Potgieter.

References

[1] D R Bean, A Ehrenfeucht, G F McNulty, http://projecteuclid.org/euclid.pjm/1102783913 Avoidable patterns in strings of symbols, Pacific Journal of Mathematics 85 (1979) 261–294
[2] T H Cormen, C E Leiserson, R L Rivest, Introduction to Algorithms, The MIT Press and McGraw-Hill Book Company (1989)
[3] J Currie, http://www.jstor.org/stable/2324790 Open Problems in Pattern Avoidance, The American Mathematical Monthly 100 (1993) 790–793
[4] K Devlin, https://books.google.com/books?id=HW26uSG2yVgC Mathematics: The Science of Patterns: The Search for Order in Life, Mind and the Universe, A Scientific American Library paperback, Henry Holt and Company (1996)
[5] W L Fouché, Unavoidable regularities and factor permutations of words, from: “Proc. Royal Society Edinburgh”, volume 125A, Cambridge University Press (1995) 519–524
[6] J Hopcroft, J Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, Reading, Massachusetts (1979)
[7] F R Madelaine, http://dx.doi.org/10.2168/LMCS-5(2:13)2009 Universal Structures and the logic of Forbidden Patterns, Logical Methods in Computer Science Volume 5, Issue 2 (2009)
[8] M Morse, G A Hedlund, http://dx.doi.org/10.1215/S0012-7094-44-01101-4 Unending chess, symbolic dynamics and a problem in semigroups, Duke Math. J. 11 (1944) 1–7
[9] P Roth, http://dx.doi.org/10.1007/BF01178567 Every binary pattern of length six is avoidable on the two-letter alphabet, Acta Informatica 29 (1992) 95–107
[10] M Sapir, V Guba, M Volkov, https://books.google.com/books?id=HYWBDQEACAAJ Combinatorial Algebra: Syntax and Semantics, Springer Monographs in Mathematics, Springer International Publishing (2016)
[11] U Schmidt, http://dx.doi.org/10.1007/BF00292112 Long unavoidable patterns, Acta Informatica 24 (1987) 433–445
[12] A Thue, https://books.google.com/books?id=-gwpGwAAACAJ Über unendliche Zeichenreihen, Skrifter utgivne af Videnskabsselskabet i Christiania: Mathematis- naturlidenskabelig Klasse (1906)
[13] A I Zimin, http://stacks.iop.org/0025-5734/47/i=2/a=A05 BLOCKING SETS OF TERMS, Mathematics of the USSR-Sbornik 47 (1984) 353