Computing the Partial Word Avoidability Indices of Ternary Patterns

By: F. Blanchet-Sadri, Andrew Lohr, Shane Scott

Blanchet-Sadri, F., Lohr, A., Scott, S. (2013). Computing the Partial Word Avoidability Indices of Ternary Patterns. *Journal of Discrete Algorithms, 23,* 119-142. doi: 10.1016/j.jda.2013.06.009

Abstract:

We study pattern avoidance in the context of partial words. The problem of classifying the avoidable binary patterns has been solved, so we move on to ternary and more general patterns. Our results, which are based on morphisms (iterated or not), determine all the ternary patterns' avoidability indices or at least give bounds for them.

Keywords: Combinatorics on words | Partial words | Pattern avoidance | Ternary pattern | Avoidability index

Article:

1. Introduction

Pattern avoidance is a topic of interest in Combinatorics on Words. A *pattern* is a sequence over an alphabet of variables, which are denoted by *A*, *B*, *C*, etc. We obtain an occurrence of the pattern if we replace the variables with arbitrary non-empty words in such a way that we replace each occurrence of the same variable with the same word. A pattern *p* is *avoidable* (resp., *k-avoidable*) if there exists an infinite word (resp., infinite word over a *k*-sized alphabet) that contains no occurrence of *p*; otherwise, *p* is *unavoidable* (resp., *k-unavoidable*). The *avoidability index* of the pattern is the smallest integer *k* for which it is *k*-avoidable; if no such *k* exists, the index is ∞.

The problem of deciding whether a given pattern is avoidable has been solved [1] and [14], but the one of deciding whether it is *k*-avoidable has remained open. An alternative is the problem of classifying all the patterns over a fixed number of variables according to their avoidability indices. This classification has been completed for unary (those over one variable *A*), for binary (those over two variables *A*, *B*), as well as for ternary patterns (those over three variables *A*, *B*, *C*) [7], [11] and [12].

For the lower bounds, we use the so-called backtracking algorithm from [8], while for the upper bounds, we provide *HD0L systems* . For a finite alphabet Σ , a morphism $f:\Sigma^{\square}\rightarrow\Sigma^{\square}$, and $a_0\in\Sigma$, the tuple $(\Sigma,f,a_0)$is called a *D0L system (Deterministic 0-sided Lindenmeyer system)* and the *D0L language* generated by the system is the set $\{f^n(a_0) \mid n\in\mathbb{N}\}$. For example, the Thue–Morse morphism t(a)=ab and t(b)=bagives the D0L system ({a,b},t,a) generating the language {ε,a,ab,abba,abbabaab,abbabaabbaababba,…}.

For a D0L system (Σ,f,a_0), the *fixed point* is $f^{\omega}(a_0)=\lim_{n\to\infty}f^n(a_0)$, provided the limit exists. The Thue–Morse word is $t^{\omega}(a)$. Now, for a morphism $g:\Sigma_1^*\rightarrow\Sigma_2^*$ with alphabets Σ_1,Σ_2 and a D0L system(Σ_1,f,a_0), the tuple $(\Sigma_1,f,a_0,\Sigma_2,g)$ is called an *HD0L system* whose generated language is the set$\{g\circ f^n(a_0)|n\in N\}$.

In [5], we have completed the classification of the avoidability indices of all the binary patterns in partial words (words with holes) that was started in [6]. The algorithms described in Section 5 of this paper have provided us with the morphisms necessary to complete this classification, which is recalled in the following theorem.

Theorem 1.
(See [5].) For partial words, binary patterns fall into three categories:

1.*The binary patterns ε, A, AA, AAB,* AABA, AABAA, *AB, ABA, and their complements, are unavoidable* (*or have avoidability index* ∞).

2.*The binary patterns* AABAB, AABB, ABAB, ABBA, *their reverses, and complements, have avoidability index* 3.

3.*All other binary patterns, and in particular all binary patterns of length six or more, have avoidability index* 2.

In this paper, we investigate the problem of classifying all the avoidable ternary patterns with respect to partial word avoidability. We identify the avoidability indices of almost all of the ternary patterns and show that only four are left in order to complete the classification (for those four we give lower and upper bounds).

The contents of our paper are as follows: In Section 2, we give some background on partial words and patterns (for more information, see [2] and [11]). In Section 3, we discuss the classification of the ternary patterns. In Section 4, we make some observations for general pattern avoidance. In Section 5, we describe an algorithm to search for an HD0L system avoiding a given pattern. Finally in Section 6, we conclude with some remarks. Note that we have put in Appendix A a ternary lexicon which lists the partial word avoidability indices for the ternary patterns, or at least lists bounds for them.

2. Preliminaries

Let Σ be a finite alphabet of letters. A *partial word* over Σ is a sequence of symbols from $\Sigma_\diamond=\Sigma\cup\{\diamond\}$, where Σ is augmented with the "hole" symbol $\diamond$. A *(full) word* is a partial word without holes. The symbol at position i of a partial word u is denoted by $u[i]$, while the *length* of u, i.e., the number of symbols in u, is denoted by $|u|$. The *empty word* ε has length zero. The set of all full words (resp., non-empty full words) over Σ is denoted by Σ^* (resp., Σ^+), while the set of all partial words (resp., non-empty partial words) over Σ is denoted by $\Sigma_\diamond^*$ (resp., $\Sigma_\diamond^+$). The set of all full (resp., partial) words over Σ of length n is denoted by Σ^n(resp., $\Sigma_\diamond^n$).

A partial word u is a *factor* (resp., *prefix*, *suffix*) of a partial word v if there exist x, y such that $v=xuy$(resp., $v=uy$, $v=xu$). The factor, prefix, or suffix u is *proper* if $u\neq\varepsilon$ and $u\neq v$. We denote by $\mathrm{Pref}(v)$(resp., $\mathrm{Suf}(v)$) the set of all prefixes (resp., suffixes) of v. If u and v are two partial words of equal length, then u is *compatible* with v, denoted by $u\uparrow v$, if $u[i]=v[i]$ whenever $u[i],v[i]\in\Sigma$. If u,v are non-empty compatible partial words, then *uv* is a *square*. A full word compatible with a factor of a partial word *v* is a *subword* of *v*.

Let Δ, $\Sigma\cap\Delta=\emptyset$, be an alphabet of pattern variables and denote them by A, B, C, etc. A *pattern* is a word over Δ, e.g., AABAACACCBAACA is a ternary pattern. We denote by $\mathrm{alph}(p)$ the set of distinct variables in pattern p. For a partial word $w\in\Sigma_\diamond^*$ and pattern $p\in\Delta^*$, we say that *w meets p* or *p occurs in w* if there exists some non-erasing morphism $\varphi:\Delta^*\to\Sigma^*$ such that $\varphi(p)$ is compatible with a factor of w; otherwise *w avoids p*. These definitions also apply to infinite partial words over Σ which are functions from$\mathbb{N}$ to $\Sigma_\diamond$. A pattern p is *k-avoidable* if there is a partial word over a k-sized alphabet with infinitely many holes that avoids p. We say that p is *avoidable* if it is k-avoidable for some k. For a given pattern p, the*avoidability index* $\mu(p)$ is the minimal k such that p is k-avoidable. If p is unavoidable, $\mu(p)=\infty$.

For a given pattern p, can we determine $\mu(p)$? A concept useful to answer this question is *division of patterns*. If p occurs in a pattern q, then *p divides q*. For instance, $ABA\underline{C}BAB\underline{C}$ divides $ABA\underline{BC}BAB\underline{BC}$(replacing C by BC gives q from p).

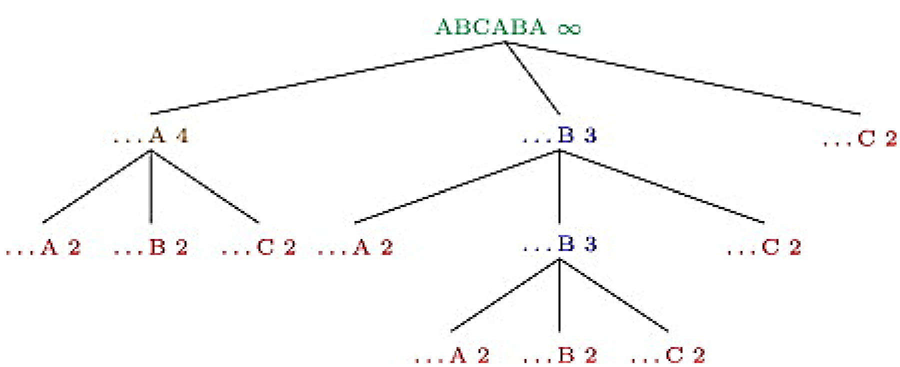
If p divides q and an infinite partial word avoids p then it also avoids q, and so $\mu(q) \leqslant \mu(p)$.

3. Classification of the ternary patterns

In classifying the avoidability indices of the ternary patterns, it is useful to consider the directed tree of patterns T, where the root of T is labeled by ε and each node has children labeled by every canonical pattern formed by appending A, B, C to the parent node's pattern, with all edges directed from parent to child. We have a partial order relation defined on the set of canonical ternary patterns by $q>p$ if there is a path in T from the node labeled by pattern q to the node labeled by pattern p. Because $q>p$ implies $q|p$, we have that $\mu(q) \geqslant \mu(p)$. The classification is complete when every node of T is appended with the avoidability index of the pattern labeling it.

First, we use unavoidability results to rule out known 2-unavoidable patterns, and proceed via a depth-first search to find 2-avoidable patterns which are identified as such using division arguments from the binary patterns and the HD0L finding algorithm described in Section 5. Once a pattern p is known to have avoidability index two, we know its children, grandchildren, etc., also have avoidability index two. We find by exhaustion that every ternary pattern with length twelve or greater is 2-avoidable. This leaves us with finitely many ternary patterns to classify. Next, for any remaining pattern p, we use division arguments and our results to establish bounds on the avoidability index of p. Finally, we try running the algorithms of Section 5 on successively larger outer alphabet sizes, starting at the known lower bound, and going up to one less than the known upper bound in search of an HD0L system which avoids p. Because the algorithm for finding HD0Ls has so many tuning parameters, the implementation used attempted to tweak these parameters, if no HD0L was found.

Here, as an example, is one branch of the tree T, starting with ABCABA:



We end this section by describing how to modify backtracking to improve lower bounds.
We consider AABCBAA first. Since we are looking for avoiding partial words with infinitely many holes, one of the length 11 factors that has a hole in its fifth position must occur infinitely

often. This factor that occurs infinitely often must avoid all of the patterns {AAA,ABBA,AABAA}. If it did not, then we would easily be able to construct a meeting morphism. We are able to show by exhaustive backtracking that all 38 such words that have a hole in their fifth position and simultaneously avoid all these patterns have length less than 11. Similarly, we obtain for AABBCBAAB the set {ABBAA,AAAA}, which only allows 220 words of length at most 34 that have a hole in their tenth position. This technique also gives us corresponding lower bounds for ABBCBBA and ABBACBBAA.
Since it is not always easy to combine the two binary patterns on either side to be a simple binary pattern, we look instead at the equivalent formula for some patterns. Recall that a formula is a set of patterns $\{p_0,\ldots,p_n\}$ often written $p_0 \cdot p_1 \cdot p_2 \cdots p_{n-1} \cdot p_n$, which meet a word *w* only if there is a morphism *h* such that for each p_i in the formula, $h(p_i)$ is a factor of *w*. Also, in order to avoid having to guess how far from the end of the factor the hole must be to make the formula unavoidable, we simply start with a hole in the middle and grow the hole out on either side. This gives us that the following patterns are 2-unavoidable:

Pattern	Formula	Number	Max length
AABACAABB	*AABA .AABB*	313	33
AABBCAABA	*AABA .AABB*	313	33
AABACBABA	*AABA .BABA*	199	20
ABAACABAB	*AABA .BABA*	199	20
AABBCABA	*AABB .ABA*	215	33
AABBCABBA	*AABB .ABBA*	129	18
AABBCBAB	*AABB .BAB*	223	34

4. Observations for general pattern avoidance

The following definitions are useful for our purposes.
Let Σ be an alphabet. For a letter $a \in \Sigma$ and a subset $I \subseteq \mathbb{N}$, we define the function $\mathrm{fill}_I^a : \Sigma_\diamond^* \to \Sigma^*$, where for $w \in \Sigma_\diamond^*$: $\mathrm{fill}_I^a(w)[i] = a$ if $w[i]=\diamond$ and $i \in I$, $\mathrm{fill}_I^a(w)[i] = w[i]$ otherwise. We write $\mathrm{fill}_{\mathbb{N}}^a$ as simplyfill$_a$. For a word $w \in \Sigma^{*}$ and a subset $I \subseteq \mathbb{N}$, we define the function $\mathrm{dig}_I : \Sigma^* \to \Sigma_\diamond^*$, where $\mathrm{dig}_I(w)[i]=\diamond$ if $i \in I$; $\mathrm{dig}_I(w)[i]=w[i]$ otherwise. By dig_j for $j \in \mathbb{N}$, we mean $\mathrm{dig}_{\{j\}}$.

4.1. Depth and shallowness

In this section, we introduce the notions of depth and shallowness of patterns. Shallow patterns, which have small depth, share some properties with full word unavoidable patterns that higher-depth patterns do not have.

A *k* -unavoidable pattern *p* is (h,k)-*deep* if there exists $m \in \mathbb{N}$ such that every partial word *w* over a *k* -sized alphabet meets *p* whenever *w* has at least *h* holes separated pairwise from each other and from the first and final position of the word by factors of length *m* or greater. We call $h:\mathbb{N}\setminus\{0,1\}\to\mathbb{N}$ the *depth function* of an unavoidable pattern p if for all k, p is $(\delta(k),k)$-deep and is not (j,k)-deep for any $j<\delta(k)$. When the depth function of p is bounded, we call its supremum d, the *depth* of p, and say that p is d*-deep*. A pattern p is k*-shallow* if p is $(0,k)$-deep or $(1,k)$-deep. If p is k-shallow for all k, we call p *shallow*. We say that p is k*-non-shallow* if it is not k-shallow.

Every shallow pattern has depth 0 or 1. Naturally, any pattern which is k-unavoidable in the full word case is $(0,k)$-deep and therefore k-shallow. Further, if p is a (h_1,k)-deep pattern and p meets pattern q then q is (h_2,k)-deep for some $h_2\leqslant h_1$. In particular, if $q|p$ and p is k-shallow then q is k-shallow. If a pattern p is (h_1,k_1)-deep, then it is also (h_2,k_1)-deep for all $h_2\geqslant h_1$ and (h_1,k_2)-deep for all $k_2\leqslant k_1$. Hence the depth function is always non-decreasing, and if the depth exists, the depth function is ultimately constant.
The following lemma gives the complete classification of the depths of the 2-unavoidable binary patterns.

Lemma 1.

The 2*-unavoidable binary patterns fall into five categories with respect to depth*:

1.*The patterns ε, A, AB, ABA, and their complements, are shallow with depth* 0.

2.*The patterns AA, AAB, their reverses, and complements, are shallow with depth* 1.

3.*The pattern* AABA*, its reverse, and complements, is* 3*-shallow,* 4*-non-shallow, and has depth* 2.

4.*The pattern* AABAA*, and its complement, is* 2*-shallow and* 3*-non-shallow, and has depth function δ satisfying* $\delta(2)=0$ *and, for all* $k\geqslant 3$, $\delta(k)=k+1$.

5.*The patterns* AABAB, AABB, ABAB, ABBA*, their reverses, and complements, are* 2*-shallow.*

Proof.

For Statement 1, the patterns *ε*, *A*, *AB*, and *ABA* are unavoidable for full words, so they are 0-deep.

For Statement 2, it is known that in the full word case *AA* and *AAB* are 2-unavoidable but 3-avoidable, hence they are $(0,2)$-deep, but not $(0,k)$-deep for any $k\geqslant 3$. They are also $(1,k)$-deep for all *k*, and therefore shallow.

For Statement 3, firstly, to show that AABA is 3-shallow, we show that it is (1,3)-deep. Assume to the contrary that for every $m \in \mathbb{N}$ there is some w, an infinite ternary word with a hole in position m which avoids AABA. Let a be the letter immediately following that hole.

If a occurs again infinitely many times in w, then $w = w' \diamond a w'' a w'''$ for some factors w', w'', w''' of w; but this implies an occurrence of AABA. Otherwise, w has an infinite binary partial word as a suffix. But AABA is 2-unavoidable, so this suffix must have an occurrence of AABA. Secondly, to show that AABA is 4-non-shallow, we show it is not (1,4)-deep. Let w be any ternary full word avoiding squares. Let a be a letter which does not occur in w and consider $w' a \diamond a w''$, where $w = w' w''' w''$ and w''' has length three. Neither w' nor w'' contain squares so any square-compatible factor in $w' a \diamond a w''$ must contain the letter a. But a never occurs in w, so $w' a \diamond a w''$ avoids AABA. Thirdly, if an infinite word w has at least two holes separated by a factor with length at least two, then it may be written as $w = w' \diamond a w'' \diamond w'''$, where a is a letter; then w has a clear occurrence of AABA. We have proved that AABA is (2,k)-deep for all k but not (1,4)-deep. Hence its depth is 2.

For Statement 4, the pattern AABAA is 2-unavoidable for full words, so it is (0,2)-deep and its depth function δ satisfies $\delta(2)=0$. Now, let w be an infinite ternary full word avoiding squares and form the infinite partial word w' from w by replacing the letter in any position with a hole. Every square occurrence of w' must contain the hole, so there are no two non-overlapping square-compatible factors. Hence w' avoids AABAA. So AABAA is (1,2)-deep, but it is not (1,3)-deep. Thus AABAA is 2-shallow and 3-non-shallow.

Let $k \geqslant 3$. To see that $\delta(k) \leqslant k+1$, we first show that AABAA is (k+1,k)-deep. Let w be any infinite word over k letters with at least $k+1$ holes separated by factors of length three or greater. By the pigeonhole principle at least one letter occurs in positions adjacent to two distinct holes. This gives us two occurrences of the same length two square-compatible factor separated by a factor of at least length one, a clear occurrence of the pattern AABAA.
We now show that AABAA is not (k,k)-deep by giving a construction with k holes arbitrarily far apart over k letters that avoids the pattern AABAA. We start with $W = \theta^{\omega}(a)$, where $\theta(a)=abc$, $\theta(b)=ac$ and $\theta(c)=b$. Let m be the minimum spacing that we are requiring to be between holes. Select factors abcacb, abcbacabc and abcbac from W in that order that are at least m positions from each other and from the start of the word. Such factors have to exist because they appear first in $\theta^2(a)$, $\theta^4(a)$, and $\theta^3(a)$ respectively, therefore, occur infinitely often in W. Replace these factors with aa⋄acb, abcc⋄cabc, and ab⋄bac respectively, calling the new word w. We want to prove that any square subword we introduce can only occur once in w. To see this, it is enough to check that the square subwords that the three substitutions introduce are distinct.

We treat the case when the squares are introduced by aa⋄acb (the other two cases are similar).

These squares must include either the second a, the ⋄, or both, because they are the only symbols changed by the substitution. In the case where the squared occurrence is suffixed by the second a, suppose it were length greater than two. We would have it suffixed by aa, which appears nowhere else to the left of aa⋄acb. So, the only square introduced in this case is the trivial occurrence aa.

So consider the case where the squares introduced involve the ⋄. If ⋄ corresponds to an a, we have the squares aa and aaaa. Note that while we introduced two distinct occurrences of the square aa, they have no letters in between, therefore, do not yield an occurrence of AABAA. If ⋄ corresponds to a b, suppose towards a contradiction that ⋄ is not the first or last letter of a squared subword. Then the subword aba appears somewhere else in w for a⋄a to correspond to. This only appears elsewhere in ab⋄bac. Now, suppose ⋄ is the last letter of a squared subword. This means that aab suffixes the squared subword, but aa appears nowhere else in w. Finally, if ⋄ starts w, any squared subword introduced would have to be prefixed by bacb. If ⋄ corresponds to a c, then the square in question must involve the a immediately to the left of the ⋄, otherwise the square would have been there before substituting the factor. If a single square occurrence extends more than one to the left of ⋄ then it contains the subword aac, and therefore cannot appear again. This leaves us only with the possibility that it extends one to the left, so we get the squareacac and possibly one square prefixed by acacb.
We get the following table for the squared subwords introduced:

Substitution	Possible squared subwords				
abcacb → *aa⋄acb*	*a*	*aa*	*ac*	*bacb⋯*	*acacb⋯*
abcbacabc → *abcc⋄cabc*	*c*	*cc*	*ca*	*bcabc⋯*	*cacabc⋯*
abcbac → *ab⋄bac*	*b*	*ab*	*ba*	–	–

Because each of the square subwords introduced by the three substitutions are distinct, w must avoid the pattern AABAA. Then, just take the prefix of w that ends at least m letters after the substituted ab⋄bacto see that AABAA is not (3,3)-deep.

Extending this construction to an avoiding word with k holes over k letters is simple. Start with w, then pick k−3 occurrences of the subword bacab that are each at least m apart after the occurrence ofab⋄bac. For the (i−3)rd of these, substitute it with $ba_i{\diamond}a_ib$ where $\{a_4,\ldots,a_k\}$ are letters distinct from a, b, and c. For each of these substitutions, we see the squares of length greater than two introduced must have an a_i as either the second or second to last position. However, a_i appears nowhere else in the word, so, in a square occurrence, it must correspond to one of the holes that were inserted. Because the letter on the other side of a_i from the hole is a b, the only hole that the a_i could correspond to is the one obtained by replacing abcbac with ab⋄bac. This means that any square occurrence of length greater than two that is introduced by this substitution must only appear once. Each also introduces the trivial squarea_ia_i which must only appear once, because each hole has different letters surrounding it. Then, just trim the infinite

word m positions after the last hole insertion. We then have, for every m, a word over k letters with k holes, each at least m spaces away from each other and from the ends of the word that avoids AABAA. This means that AABAA is not (k,k)-deep for any k.
For Statement 5, the patterns AABAB, AABB, ABAB, ABBA are 2-unavoidable for full words, and therefore (0,2)-deep. They are 3-avoidable for partial words. Hence they are not (h,k)-deep for any $k\geqslant 3$ and any h. □

The following theorem gives a use of shallowness.

Theorem 2.

Let $p_0,p_1,\dots,p_n$ *be k-unavoidable patterns over* Δ *and let* $A_1,\dots,A_n$ *be variables which are not in* Δ*. Then* $p_0A_1p_1\cdots A_np_n$ *is k-unavoidable if any of the following conditions hold*:

1. $\mathrm{alph}(p_i)$ *and* $\mathrm{alph}(p_j)$ *are pairwise disjoint for all* $i\neq j$;

2. *there exists some k-shallow pattern p such that* $p_0,\dots,p_n$ *are factors of p*; *further, if p is* (0,k)*-deep, so is* $p_0A_1p_1\cdots A_np_n$.

Proof.

For Condition 1, let p_0, p_1 be k-unavoidable patterns over Δ and let A_1 be a variable not in Δ. Let Σ be a k-sized alphabet and w be a partial word over Σ with infinitely many holes. Because p_0 and p_1 are k-unavoidable, there must be an infinite number of occurrences of both p_0 and p_1 in w. Then there is an occurrence of p_0 followed by a non-overlapping occurrence of p_1, i.e., there exist non-erasing morphisms $h_0,h_1:\Delta^{□}\to\Sigma^{□}$ and factors w_0, w_1, w', w'', w''' of w such that $h_0(p_0)\uparrow w_0$, $h_1(p_1)\uparrow w_1$ and $w=w'w_0w''w_1w'''$. Consider the non-erasing morphism $f:(\Delta\cup\{A_1\})^{□}\to\Sigma^{□}$ defined by

$$f(B)=\begin{cases} h_0(B), & \text{if } B\in\mathrm{alph}(p_0);\\ h_1(B), & \text{if } B\in\mathrm{alph}(p_1);\\ w'', & \text{if } B=A_1;\\ a, & \text{otherwise,}\end{cases}$$

where $a\in\Sigma$. As $\mathrm{alph}(p_0)$ and $\mathrm{alph}(p_1)$ are disjoint, we are guaranteed that the function f is well-defined. Clearly $f(p_0A_1p_1)\uparrow w_0w''w_1$, so w meets $p_0A_1p_1$. The result then follows by induction on n.

For Condition 2, let $p_0,p_1,\dots,p_n$ be k-unavoidable patterns over Δ, let p be a k-shallow pattern such that $p_0,\dots,p_n$ are factors of p, and let $A_1,A_2,\dots,A_n$ be variables not in Δ. Let Σ be a k-letter alphabet, and let w be a partial word over Σ with infinitely many holes. Let $m\in\mathbb{N}$ be the integer implied by the k -shallowness of p . Write $w=w'_0w_0w'_1w_1\cdots$, where the w_i's are length m factors with at least one hole. There are at most $(k+1)^m$ possible w_i, so at least one must occur infinitely often; call it x . Then $w=y_0xy_1xy_2\cdots xy_{n+1}$ for some y_i's. Because p is k -shallow, we have that x meets pattern p , so there is some non-erasing morphism $h:(\Delta\cup\{A_1,\ldots,A_n\})^{\diamond}\to\Sigma^{\diamond}$ such that h(p) is compatible with a factor of x . Thus, for some x_i, x'_i, x''_i, we may write $x=x_ix'_ix''_i$ where $x'_i\uparrow h(p_i)$, and $w=y_0x_0x'_0x''_0y_1x_1x'_1x''_1y_2\cdots x_nx'_nx''_ny_{n+1}$. This clearly has an occurrence of $q=p_0A_1p_1\cdots A_np_n$, for let $f:(\Delta\cup\{A_1,\ldots,A_n\})^{\diamond}\to\Sigma^{\diamond}$ be the morphism defined by $f(B)=\mathrm{fill}_a(x''_{i-1}y_ix_i)$ if $B=A_i$, and $f(B)=h(B)$ otherwise, where $a\in\Sigma$. Then w has factors compatible with f(q), so w meets q . If p is (0,k)-deep, then the same argument holds with any filling of the holes in w and with w_i any length m factor, and it follows that q is (0,k)-deep. □

Corollary 1.

The sequence of patterns defined recursively by $p_0=A_0A_0$ *and* $p_{n+1}=p_nA_{n+1}p_n$ *is* 2-*unavoidable.*

Proof.

It was shown in Lemma 1 that AA is (0,2)-deep. Then the result follows by induction from Theorem 2. □

Corollary 2.

Let p be a pattern of only distinct variables over Δ and $i<|p|$ *such that* $p_0,p_1,\ldots,p_n\in\Delta^{\diamond}$ *are compatible with factors of* $\mathrm{dig}_i(p)$*. If* $A_1,\ldots,A_n$ *are distinct variables not in Δ, then* $p_0A_1p_1\cdots A_np_n$ *is unavoidable.*

Proof.

Let p be any pattern where no variable occurs more than once in p . Observe that any word of length $|p|$ with a hole in position i meets every pattern compatible with $\mathrm{dig}_i(p)$. It follows that every pattern $p'_0,\ldots,p'_n$ which is a factor of $\mathrm{dig}_i(p)$ is (1,k)-deep for all k . By Theorem 2 we have that $p'_0A_1p'_1\cdots A_np'_n$ is k -unavoidable for all k . Note that we can find an occurrence of $p'_0A_1p'_1\cdots A_np'_n$ with the image of every variable in alph(p) of length 1. Then any completion $p_0A_1p_1\cdots A_np_n$ of $p'_0A_1p'_1\cdots A_np'_n$ with variables from alph(p) (i.e., any filling in of the holes in $p'_0A_1p'_1\cdots A_np'_n$ with variables from alph(p)) has an occurrence whenever $p'_0A_1p'_1\cdots A_np'_n$ does, hence it is also unavoidable. □

Applying Theorem 2 and its corollaries to the patterns in Lemma 1 imply, for instance, that the ternary pattern AABAAC, its reversal, its permutations, and its factors are unavoidable; the pattern AABACAAB (resp.,AABAACAAB), its reversal, its permutations, and its factors are 3-unavoidable (resp., 2-unavoidable). The pattern (AABA)C(AAB) is 3-unavoidable because both AABA and *AAB* are factors of AABAwhich is 3-shallow. There are many patterns that can be classified this way!

4.2. Rules of inference

In this section, we construct partial words avoiding patterns avoidable for full words. Let *p* be a pattern over$\Delta=\{A_1,\ldots,A_n\}$. When we discuss ternary patterns, we write $A=A_1$, $B=A_2$, and $C=A_3$. Suppose that *p* is avoided by *w*, an infinite full word over a *k* -letter alphabet $\Sigma=\{a_1,a_2,\ldots,a_k\}$. There are a finite number of length three factors of *w*, so at least one has infinitely many non-overlapping occurrences. Then there exists an infinite integer sequence $\langle i_m \rangle$ where $|i_m-i_{m'}|\geqslant 3$ and $w[i_m-1..i_m+1]=w[i_{m'}-1..i_{m'}+1]$ for all distinct m,m'. Let $\langle j_m\rangle$ be an infinite subsequence of $\langle i_m\rangle$ such that $j_m>2j_{m-1}+5$, and form the partial word w' from *w* by replacing $w[j_m-1..j_m+1]$ with $a_{k+1}\diamond a_{k+2}$. Then w' is a partial word with infinitely many holes over the alphabet $\Sigma\cup\{a_{k+1},a_{k+2}\}$. It turns out that w' and its reverse, $\mathrm{rev}(w')$, have many useful properties and avoid many patterns between them.

We refer to $A_{i,j}$ as the *j* th occurrence of A_i in *p*, though we drop these subscripts when they are clear from the context. We define a relation on the set of factors of *p*, Fact(p), by $q\lessdot q'$ if *q* is an abelian factor of q' and there are non-overlapping occurrences of *q* and q'. For example, if $p=ABCDCB$ then $B\lessdot B$, $B\lessdot AB$, $BC\lessdot DCB$, and $CB\lessdot BC$.
Assume that for some non-erasing morphisms $h,g:(\Delta\times\{1,\ldots,|p|\})^*\to(\Sigma\cup\{a_{k+1},a_{k+2}\})^*_\diamond$ we have $w'=u_1h(p)v_1$, where $h(A_{i,j})\uparrow h(A_{i,\ell})$ for all $1\leqslant j,\ell\leqslant|p|$, and for some factor w'' of $\mathrm{rev}(w')$ we have$w''=u_2g(p)v_2$. This is equivalent to w' and $\mathrm{rev}(w')$ meeting *p*. If we arrive at a contradiction, after proving that w' or $\mathrm{rev}(w')$ avoid *p*, we have shown that *p* is (k+2)-avoidable.

Write $\overset{\diamond}{q}$ when h(q) is a hole; $\overset{\lrcorner}{q}$ when a_{k+1} suffixes h(q); $\overset{\llcorner}{q}$ when a_{k+2} prefixes h(q); $\overset{\lrcorner\diamond}{q}$ when $a_{k+1}\diamond$suffixes h(q); $\overset{\diamond\llcorner}{q}$ when $\diamond a_{k+2}$ prefixes h(q); $\overset{\sqcup}{q}$ when for some proper factor *u* of w', *u* is a factor ofh(q) and h(q) is a factor of $\diamond a_{k+2}ua_{k+1}\diamond$ (when we have $\overset{\sqcup}{q}$ we say that *q* is *horned*); $\overset{1}{q}$ when h(q)has length one.

Theorem 3.

Let q,q'*be factors of pattern p over* $\{A_1,\ldots,A_n\}$. *Let* q_i*denote an occurrence of q beginning at index i of p. The following rules of inference hold*:

(a)$\overset{\diamond}{A}_{i,j}\Longrightarrow\forall\ell:\ \overset{1}{A}_{i,\ell}$

(b) $\overset{\diamond}{A}_{i,j}A_{\ell,m} \Longrightarrow \overset{\diamond}{A}_{i,j}\overset{\llcorner}{A}_{\ell,m}$

(c) $A_{i,j}\overset{\diamond}{A}_{\ell,m} \Longrightarrow \overset{\lrcorner}{A}_{i,j}\overset{\diamond}{A}_{\ell,m}$

(d) $\neg\overset{\lrcorner}{A}_{i,j}\overset{\llcorner}{A}_{\ell,m}$

(e) $\overset{\lrcorner}{A}_{i,j} \Longrightarrow \forall\ell\colon\ \overset{\lrcorner}{A}_{i,\ell} \vee \overset{\diamond}{A}_{i,\ell}$

(f) $\overset{\llcorner}{A}_{i,j} \Longrightarrow \forall\ell : \overset{\llcorner}{A}_{i,\ell} \vee \overset{\diamond}{A}_{i,\ell}$

(g) $\overset{\llcorner}{q}_i \wedge (\overset{\lrcorner\diamond}{q}_j \vee \overset{\lrcorner}{q}_j) \Longrightarrow \overset{\sqcup}{q}_i$

(h) $(\overset{\llcorner}{q}_i \vee \overset{\diamond\llcorner}{q}_i) \wedge \overset{\lrcorner}{q}_j \Longrightarrow \overset{\sqcup}{q}_i$

(i) $\overset{\lrcorner\diamond}{q}_i \vee \overset{\diamond\llcorner}{q}_i \Longrightarrow \forall\ell\colon\ \overset{\lrcorner\diamond}{q}_\ell \vee \overset{\diamond\llcorner}{q}_\ell$

(j) $\overset{\lrcorner\diamond}{q}_i \wedge \overset{\diamond\llcorner}{q}_j \Longrightarrow \forall\ell\colon\ \overset{\sqcup}{q}_\ell \vee\ h(q_\ell) \in \{\diamond a_{k+2}, a_{k+1}\diamond\}$

(k) $\exists A_{i,j}$: $\overset{\diamond}{A}_{i,j}$, $\overset{\lrcorner\diamond}{A}_{i,j}$, *or* $\overset{\diamond\llcorner}{A}_{i,j}$

(l) $\overset{\sqcup}{q} \Longrightarrow \neg q \lessdot q'$

(m) $A_{i,j}A_{\ell,m}A_{i,j+2} \Longrightarrow \neg\overset{\diamond}{A}_{\ell,m}$

Proof.

For (a), it should be clear that as $h(A_{i,j})\uparrow h(A_{i,\ell})$ for all j,ℓ, if $h(A_{i,j})=\diamond$ then $|h(A_{i,\ell})|=1$.
For (b), (c), (d), by construction $\diamond$, a_{k+1}, and a_{k+2} occur only in the factor $a_{k+1}\diamond a_{k+2}$.
For (e), suppose in the pattern that $A_{i,j}$ satisfies $\overset{\lrcorner}{A}_{i,j}$. Because $h(A_{i,j})\uparrow h(A_{i,\ell})$ for all ℓ, we see that $h(A_{i,\ell})$ must end with a_{k+1} or $\diamond$. But if it ends in $\diamond$, then it can only be that $h(A_{i,\ell})=\diamond$, for if $|h(A_{i,\ell})|\geqslant 2$ then $h(A_{i,j})$ is suffixed by aa_{k+1} for some $a\in\Sigma$ and $a_{k+1}\diamond$ suffixes $h(A_{i,\ell})$. But then $aa_{k+1}\not\uparrow a_{k+1}\diamond$. The proof for (f) is similar.

For (g), (h), if for some factor q_i of the pattern we have $\overset{\llcorner}{q}_i \wedge (\overset{\lrcorner\diamond}{q}_j \vee \overset{\lrcorner}{q}_j)$ or $(\overset{\llcorner}{q}_i \vee \overset{\diamond\llcorner}{q}_i) \wedge \overset{\lrcorner}{q}_j$, then for some proper factor u of w', $h(q_i)$ is a factor of $\diamond a_{k+2}ua_{k+1}\diamond$ but not of $ua_{k+1}\diamond$ nor $\diamond a_{k+2}u$.
Then u must be a factor of $h(q_i)$.

For (i), if for some factor q_i of p we have $\diamond a_{k+2}\in \mathrm{Pref}(h(q_i))$ or $a_{k+1}\diamond\in \mathrm{Suf}(h(q_i))$, then either $h(q_\ell)\in\{\diamond a_{k+2},a_{k+1}\diamond\}$ or $h(q_i)=h(q_\ell)$.

For (j), note that as neither a_{k+1} or a_{k+2} occur in w and w has no holes, the factors $a_{k+1}\diamond$ and $\diamond a_{k+2}$ can only be compatible with factors which overlap $a_{k+1}\diamond a_{k+2}$. It is easy to see that $a_{k+1}\diamond$ or $\diamond a_{k+2}$ are only compatible with themselves and each other. Then if $\overset{\lrcorner\diamond}{q}_i$ and $\overset{\diamond\llcorner}{q}_j$, it must be that for all ℓ either $h(q_\ell)\in\{\diamond a_{k+2},a_{k+1}\diamond\}$ or $h(q_\ell)=\diamond a_{k+2}ua_{k+1}\diamond$ for some factor u of w'.
For (k), assume towards a contradiction that $|h(A_{i,j})|>2$ for all $A_i\in \mathrm{alph}(p)$. Note first that because w avoids p, if w' meets p then for some $A_i\in \mathrm{alph}(p)$, there are two distinct occurrences of A_i in p such that $h(A_{i,j})\uparrow h(A_{i,\ell})$ but $h(A_{i,j})\neq h(A_{i,\ell})$. To see this write $w' = u_1w'_0w'_2\cdots w'_{|p|-1}u_2$, where $w'_i = h(p[i])$. Let the corresponding subwords of w be w_i obtained by the

mapping $a_{k+1} \mapsto w[j_0-1], \diamond \mapsto w[j_0]$, and $a_{k+2} \mapsto w[j_0+1]$, which undoes our original replacement.
If $w'_i = w'_j$, then $w_i = w_j$, so if p occurs in w' without any compatible but unequal variable images then p would also occur in w. So there must be a pair of factors w'_i, w'_j such that $w'_i \neq w'_j$ but $w'_i \uparrow w'_j$. Consider then length 3 factors of w' with holes. They are

$$a_i \quad a_{k+1} \quad \diamond,$$
$$a_{k+1} \quad \diamond \quad a_{k+2},$$
$$\diamond \quad a_{k+2} \quad a_j,$$

where $a_i, a_j \in \Sigma$. We see that none of these are compatible, so the only distinct, compatible factors of w' are $\diamond$ with any letter, and the length two factors $a_{k+1}\diamond$ and $\diamond a_{k+2}$. Then there is some occurrence of some A_i in p, $A_{i,j}$, such that $h(A_{i,j}) \in \{\diamond, \diamond a_{k+2}, a_{k+1}\diamond\}$ and $|h(A_{i,j})| \leqslant 2$, a contradiction.
For (l), assume to the contrary that there are factors q' and q'' of the pattern with non-overlapping occurrences such that $\overset{\sqcup}{q'}$ and $q' \lessdot q''$. Note that $|h(q')| \leqslant |h(q'')|$ and $|g(q')| \leqslant |g(q'')|$. Further since $\overset{\sqcup}{q'}$, it must be the case that $h(q')$ extends from near one hole to near the next hole, or more precisely, $h(q')$ occupies at least every position of w' from position j_n+2 to position $j_{n+1}-2$ for some n. Suppose that $h(q')$ occurs after j_n, the index of the $(n+1)$th hole. Then if q is a factor of the pattern which occurs before q' and does not overlap with q', we see that $|h(q')| \geqslant j_{n+1} - j_n - 3 > j_n + 2 \geqslant |h(q)|$. As $|h(q'')| \geqslant |h(q')|$, it must be that q'' occurs after q'. But we similarly have that $|g(q)| < |g(q')|$ whenever q occurs after q', so $|g(q'')| < |g(q')|$.
But $q' \lessdot q''$ implies $|g(q')| \leqslant |g(q'')|$. This is a contradiction.

For (m), assume to the contrary that we have $A_{i,j} \overset{\diamond}{A}_{\ell,m} A_{i,j+2}$. Then we have $\overset{\lrcorner}{A}_{i,j} \overset{\diamond}{A}_{\ell,m} \overset{\llcorner}{A}_{i,j+2}$. By (g) we have $\overset{\sqcup}{A}_{i,j+2}$, but $A_{i,j+2} \lessdot A_{i,j}$ which is in contradiction with (l). □
Several constructions, nearly identical to the construction from Theorem 3, can avoid many patterns with specific structures occurring in their factors.

Theorem 4.

Let p be a pattern over alphabet Δ with a squared variable factor AA for some $A \in \Delta$. *Then the following hold*:

1. *If there are factors* $Aq'A$ *and* q'' *of p such that* $q' \lessdot q''$, *then either the image of* q' *consists of a single letter or p is* 4-*avoidable.*
2. *If there are factors* q'' *and* $AAq'A$ *or* $Aq'AA$ *of p such that* $q' \lessdot q''$, *then p is* 4-*avoidable.*
3. *If there are factors* q'' *and* $AAq'BB$ *of p such that* $q' \lessdot q''$ *for some* $B \in \Delta$, *then p is* 3-*avoidable.*

Proof.

Let $\Sigma = \{a,b,c\}$. Let $\theta : \Sigma^{*} \rightarrow \Sigma^{*}$ be the generalized Thue–Morse morphism defined by $\theta(a) = abc, \theta(b) = ac$, and $\theta(c) = b$. Define the morphism $\phi : \Sigma^* \rightarrow \Sigma^*_\diamond$ as θ^3 with the factor *bab* of $\theta^3(a)$ changed to $d\diamond d$, i.e.,

$$\phi(a_i) = \begin{cases} abcacd\diamond dcbac, & \text{if } a_i = a; \\ abcacbac, & \text{if } a_i = b; \\ abcb, & \text{if } a_i = c. \end{cases}$$

Let $w=\phi\circ\theta^{\omega}(a)$ and let $\langle i_n\rangle$ be the sequence of indices of holes of w, i.e., $w[i]=\diamond$ if and only if $i\in\langle i_n\rangle$. Let $\langle j_n\rangle$ be any subsequence of $\langle i_n\rangle$ such that $j_{n+1}>2j_n+7$. We form w' from w by replacing $w[i_n-1..i_n+1]$ with $d\diamond d$ if $i_n\in\langle j_n\rangle$ or with *bab* if not. Let f be the identity map on Σ and $f(d)=b$. Note that $f\circ \mathrm{fill}_a(w)=f\circ \mathrm{fill}_a(w')=\theta^{\omega}(a)$ which is known to be square-free [10]. It follows that any square-compatible factor of w' must contain both $\diamond$ and d. We show that the set of square subwords of w' is exactly $\{dd,cdcd,dcdc\}$. Note that any length four or greater factor of w' is always equal whenever it is compatible, as the length four factors of w' containing d or $\diamond$ are

$$\begin{array}{cccc} c & a & c & d, \\ a & c & d & \diamond, \\ c & d & \diamond & d, \\ d & \diamond & d & c, \\ \diamond & d & c & b, \\ d & c & b & a, \end{array}$$

which are all pairwise incompatible. It follows that if there exists any length eight or greater square-compatible factor $s=s_1s_2$ where $s_1\uparrow s_2$, then $s_1=s_2$ which implies $f\circ \mathrm{fill}_a(s_1)=f\circ \mathrm{fill}_a(s_2)$, so $f\circ \mathrm{fill}_a(s)$ is a square factor of $\theta^{\omega}(a)$, a contradiction. Then every square-compatible factor has length six or smaller and must be a factor of $\phi(a)$. It is easy to see from $\phi(a)$ that the only square subwords have length two or four and are dd, $cdcd$, $dcdc$.

Let $p\in\Delta^{\square}$ and $A\in\Delta$. For Statement 1, suppose that AA, $Aq'A$, and q'' are factors of p such that $q'<q''$. Suppose that $h,g:\Delta^{\square}\to\Sigma^{\square}$ are non-erasing morphisms with $h(p)$ compatible with a factor of w', and $g(p)$ compatible with a factor of $\mathrm{rev}(w')$, i.e., both w' and its reverse meet p. Observe that $h(A)$ must contain a d and that $h(q')$ only occurs at position j_n, j_n+1, or j_n+2 for some n. It follows that $h(Aq'A)\uparrow d\diamond d$ or for some n we have $|h(Aq'A)|\geqslant j_{n+1}-j_n-1$ so $|h(q')|\geqslant j_{n+1}-j_n-5>j_n+2\geqslant|h(q)|$, where q is any factor of p non-overlapping with q' and occurring before q'. Note that because $q'<q''$ we have $|h(q')|\leqslant|h(q'')|$, so q'' cannot occur before q'. A similar argument shows that $|g(q')|>|g(q)|$ whenever q is a factor of p non-overlapping with q' occurring after q'. As q'' occurs after q', we have $|g(q')|>|g(q'')|$. But $q'<q''$ implies that $|g(q')|\leqslant|g(q'')|$. This is a contradiction, so $h(Aq'A)\uparrow d\diamond d$ and $|h(q')|=1$. For Statement 2, if the factor $AAq'A$ or the factor $Aq'AA$ occurs in p then $|h(q')|\neq 1$ as $h(A)$ must contain d, $|h(A)|\leqslant 2$, and three d's do not occur in any length seven factors of w'.

For Statement 3, let w' and f be as above and define $w''=f(w')$. We show that the set of square subwords of w'' is $\{bb,cbcb,bcbc\}$. Observe that the length five factors of w'' containing a hole are

$c\ a\ c\ b\ \diamond,$
$a\ c\ b\ \diamond\ b,$
$c\ b\ \diamond\ b\ c,$
$b\ \diamond\ b\ c\ b,$
$\diamond\ b\ c\ b\ a,$

and all of these are pairwise incompatible. This means any square-compatible factor has length eight or less, but it is easy to check that the set of square-compatible factors of $\phi(ax)$ and $\phi(xa)$ for $x\in\{b,c\}$ is exactly $\{bb,bcbc,cbcb\}$. Suppose that *p* has factors $AAq'BB$ and q'' such that $q'\lessdot q''$. Assuming to the contrary that both w'' and $rev(w'')$ meet *p*, we argue as in the previous case. □

Theorem 5.
The pattern AABCBAA *has avoidability index* 4.

Proof.

The pattern AABCBAA is 3-unavoidable by backtracking. We claim that it is 4-avoidable. We proceed similarly as in the proof of Theorem 4. Let $\Sigma=\{a,b,c\}$. Let $\theta:\Sigma^{*}\to\Sigma^{*}$ be the generalized Thue–Morse morphism defined by $\theta(a)=abc$, $\theta(b)=ac$, and $\theta(c)=b$. Define the morphism $\phi:\Sigma^*\to\Sigma_\diamond^*$ as θ^3 with the factor cacba of $\theta^3(a)$ changed to $\diamond dcbd$, i.e.,

$$\phi(a_i)=\begin{cases} ab\diamond dcbdbcbac, & \text{if } a_i=a;\\ abcacbac, & \text{if } a_i=b;\\ abcb, & \text{if } a_i=c.\end{cases}$$

Let $w=\phi\circ\theta^{\omega}(a)$ and let $\langle i_n\rangle$ be the sequence of indices of holes of *w*, i.e., $w[i]=\diamond$ if and only if $i\in\langle i_n\rangle$. Let $\langle j_n\rangle$ be any subsequence of $\langle i_n\rangle$ such that $j_{n+1}>2j_n+8$. We form w' from *w*, for all i_n replacing $w[i_n..i_n+4]$ with $\diamond dcbd$ if $i_n\in\langle j_n\rangle$ or with cacba if not. Let *f* be the identity map on Σ and $f(d)=a$. Note that $f\circ fill_c(w)=f\circ fill_c(w')=\theta^{\omega}(a)$ which is known to be square-free [10]. It follows that any square-compatible factor of w' must contain $\diamond$. We show that the set of square subwords of w' is exactly $\{bb,dd,baba\}$. Note that any length three or greater factors of w' are equal whenever compatible, as the length three factors of w' containing $\diamond$ are
$a\ b\ \diamond,$
$b\ \diamond\ d,$
$\diamond\ d\ c,$

which are all pairwise incompatible. It follows that any length six or greater square-compatible factor $s=s_1s_2$ of w', where $s_1\uparrow s_2$, satisfies $s_1=s_2$. This implies $f\circ fill_c(s_1)=f\circ fill_c(s_2)$, so $f\circ fill_c(s)$ is a square factor of $\theta^{\omega}(a)$, a contradiction. Therefore, every square-compatible factor of w' has length four or smaller and must contain $\diamond$. It is easy to see from $\phi(a)$ that the only square subwords that have length two or four are *bb*, *dd*, baba, the last of which occurring in $\phi(c)\phi(a)$. Assume towards a contradiction that w' meets AABCBAA with meeting morphism *h*.

Case 1 . We have h(A)=d. In this case, note that the h(AA) occurrences happen only in one of the substituted strings, in the positions occupied by $\diamond d$. Therefore, if h(B) were length one or two, then it clearly would not work, because c≠b and cb≠ab. So, we consider h(B) to be length three or greater. Therefore, h(B) must have d in its third position, so, for each h(B), the first letter in h(B) is either one position to the left or two positions to the right of a hole.

Case 2 . We have h(A)=b. Because the word that we are substituting into is square-free except for factors containing $\diamond$, the h(AA) occurrences can happen only in the positions occupied by $b\ \diamond$. Therefore, the first letter of h(B) is a *d*, so, the left end of any image of *B* is either one or four positions to the right of a hole.

Case 3 . We have h(A)=ba. The only occurrence of this is in ϕ(c)ϕ(a). Therefore, the first letter of h(B)is a *d*, so, the left end of any image of *B* is either one or four positions to the right of a hole.

Since *BAA* is a factor of AABCBAA then the right side of the image of the *B* in *BAA* must be either one, two, or four positions to the left of a hole, depending on what h(A) is. So, we have that there are distinct holes within four positions of either side of the second image of *B* . So, the distance between these two holes is then at most 8+|h(B)| but, since it happens after an occurrence of h(B), this gap of length at most8+|h(B)| is starting after index |h(B)|. This contradicts the restriction we placed on the positions of holes by our definition of j_n. So, the word w' avoids AABCBAA. □

A similar argument to that in Theorem 5 shows that ABBCBBA has avoidability index 4.

Theorem 6.

The pattern AABCABA*has avoidability index* 5*.*

Proof.

Let *W* be a 4-letter word with infinitely many holes. We know that the full word avoidability index ofAABCABA is 3 (hence it is 2-unavoidable) [8]. This means that there is a maximal length of avoiding words on two letters, say *n* . Define *w* to be the word obtained from *W* by starting at the beginning and filling in a hole if it is less than n+5 positions away from the most recent hole that has not been filled in. Note that in *w* , all holes are then separated by a distance of at least n+5. Since *W* is then *w* with the possible replacement of some letters with holes, if we can show that *w* meets AABCABA, then *W* will as well. Then, since there are only 16 configurations of the adjacent letters surrounding each hole, at least one of them has to occur infinitely often, say $a\diamond b$. In all cases to follow, we construct *h* a non-erasing morphism taking AABCABA to a subword of *w*.

Case 1 . We have $a=b$. Because $a\diamond a$ occurs infinitely often, it occurs twice separated by at least one letter, so, there exists a finite non-empty partial word w_0 such that $a\diamond aw_0a\diamond a$ is a factor of *w* then, just let $h(A)=a, h(B)=a, h(C)=w_0$.

Case 2 . We have $a\neq b$. Let *c* and *d* denote the remaining two letters in alph(w). Since there are at most 2^n words on $\{c,d\}$ that avoid AABCABA, and infinitely many occurrences of $a\diamond b$, there must be some w_0 such that $a\diamond bw_0$ occurs infinitely often, followed either by *a* or *b*.
First, suppose that $a\diamond bw_0a$ occurs infinitely often. We can find two occurrences whose starting positions are at least $n+5$ positions apart, meaning that for some $w_1\neq\epsilon$, $a\diamond bw_0aw_1a\diamond bw_0a$ is a factor of *w* . So, let $h(A)=a, h(B)=bw_0, h(C)=aw_1a$. Now, suppose that $a\diamond bw_0b$ occurs infinitely often with $w_0=\epsilon$. Then $a\diamond bb$ occurs infinitely often, in particular it occurs twice at least four positions apart, so, for some w_2, *w* has the factor $a\diamond bbw_2a\diamond bb$. Take $h(A)=b, h(B)=b, h(C)=w_2a$.
Next, suppose that $a\diamond bw_0b$ occurs infinitely often with $w_0\neq\epsilon$. Then, in particular, it occurs starting in two positions that are separated by at least $n+5$ positions, so there is some $w_3\neq\epsilon$ by length considerations such that *w* has the factor $a\diamond bw_0bw_3a\diamond bw_0b$, so, let $h(A)=b, h(B)=w_0$, and $h(C)=bw_3aa$.

So, any word with infinitely many holes on an alphabet of size four has to meet the pattern AABCABA.

We now show that AABCABA is 5-avoidable. Our claim is that the word w' from Theorem 4, except with the factor *bab* replaced with $d\diamond e$ instead of $d\diamond d$ avoids AABCABA. We will be making use of the notation and inference rules given in Theorem 3. We first introduce a new rule:

$$(n)\ (\overset{1\llcorner}{A}_{i,j} \vee \overset{\diamond}{A}_{i,j} \vee \overset{1\lrcorner}{A}_{i,j})q(\overset{1\llcorner}{A}_{l,m} \vee \overset{\diamond}{A}_{l,m} \vee \overset{1\lrcorner}{A}_{l,m}) \Longrightarrow \overset{\sqcup}{q}.$$

To see this, note that the image of $A_{i,j}$ and $A_{l,m}$ are each length one and within one of the position of a hole. This means that there is some *u* such that $d\diamond eud\diamond e$ is a factor of w' and $A_{i,j}$ maps to either the *d* , $\diamond$, or *e* on the left and $A_{l,m}$ maps to either the *d* , $\diamond$, or *e* on the right. Then, the image of *q* is a factor of $\diamond eud\diamond$ and must have *u* as a factor, meaning $\overset{\sqcup}{q}$.
Suppose that w' meets AABCABA with meeting morphism *h* . Since the only square occurrences in w' are {dd,ee}, we consider the following cases:

Case 1 . We have $h(A)=d$. Here $\overset{1\lrcorner}{A}\overset{\diamond}{A}\overset{\llcorner}{B}CABA$ which means that, by rule (*e*), the factor of *ABA* will be either $\overset{\diamond}{A}B\overset{\diamond}{A}$, $\overset{\diamond}{A}B\overset{1\lrcorner}{A}$, $\overset{1\lrcorner}{A}B\overset{\diamond}{A}$, or $\overset{1\lrcorner}{A}B\overset{1\lrcorner}{A}$. In any case, by (*n*), we get $\overset{\sqcup}{B}$. Then, since $B<B$, by rule (*l*) we get a contradiction, so w' avoids the pattern.

Case 2 . We have h(A)=e. Here $\overset{\diamond}{A}\overset{1\llcorner}{A}BCABA$ which means that, by rule (*f*), the factor of *ABA* will be either $\overset{\diamond}{A}B\overset{\diamond}{A}$, $\overset{\diamond}{A}B\overset{1\llcorner}{A}$, $\overset{1\llcorner}{A}B\overset{\diamond}{A}$, or $\overset{1\llcorner}{A}B\overset{1\llcorner}{A}$. In any case, by (*n*) we get that $\overset{\sqcup}{B}$. Then, since B⋖B, by rule (*l*) we get a contradiction, so w' avoids the pattern.
Thus, the avoidability index is five. □

Theorem 6 is interesting because in order to get a full word avoidability index of five, the only pattern we know of is far more complicated, using nine variables (not just three) in the pattern:ABVACWBAXBCYCDAZDCD [9]. Also, a similar argument shows that ABACAAB has avoidability index 5.

Theorem 7.

The pattern AABACBAA*has avoidability index* 3.

Proof.
Applying both Theorem 2 and Lemma 1 imply that the pattern AABACBAA is 2-unavoidable becauseAABA and *BAA* are factors of AABAA which is 2-shallow.
To show that it is 3-avoidable, recall that *aba* does not appear as a subword in $\theta^{\omega}(a)$ because any occurrence of *a* is followed by either *c* or *bc* . Let *S* be the set of positions of *a* in $\theta^{\omega}(a)$. Form the sequence $J=\{j_n\}_{n\in N}$ defined as follows: $j_0=1\in J$ and $j_n\in J$ if $j_n\in S$ and $j_n>2j_{n-1}+8$. Then, we claim$\phi\circ\theta^{\omega}(a)$ avoids the pattern AABACBAA, where, in the following definition of ϕ we use the subscript to denote the position of the letter:

$$\phi(a_i)=\begin{cases} abcac\underline{\diamond}abcbacab\underline{aca}bacab\underline{a}babcacbabcbacabcbabcacbac, & \text{if } a_i=a \\ & \text{and } i\in J; \\ abcacbabcbacabcacbacabcbabcacbabcbacabcbabcacbac, & \text{if } a_i=a \\ & \text{and } i\notin J; \\ abcacbabcbacabcacbacabcbabcacbac, & \text{if } a_i=b; \\ abcacbabcbacabcb, & \text{if } a_i=c. \end{cases}$$

Note that $\phi(b)=\theta^5(b)$ and $\phi(c)=\theta^5(c)$. Also note that the positions where $\phi(a)$ differs from $\theta^5(a)$are underlined. Let $w'=\phi(a_i)$ where $a_i=a$ and $i\in J$. We first confirm that any factors of w' of length at least three that are compatible are equal, noticing the following are pairwise incompatible:
$a\ c\ \diamond$,
$c\ \diamond\ a$,
$\diamond\ a\ b$.

Thus the only square occurrences that arise as a result of the ⋄ are {aa,cc,caca}. The only other squares that could have been introduced are a result of the subwords *aca* or *a* that we replaced *cac* and *c*with respectively.
Assume towards a contradiction that $\phi\circ\theta^{\omega}(a)$ meets AABACBAA with meeting morphism *h*.

Case 1 . We have h(A)=a, h(A)=ca, or h(A)=c.

$ab\overline{cac\diamond a}bcbacabacabacabababcacbabcbacabcbabcacbac$

These each only have one square occurrence, occurring to the left of every occurrence of *aba* . Each occurrence of h(A) that occurs to the right of h(AA) but before the subword *aba* would require h(B)to be a subword that cannot appear to the right of h(AA) in ϕ(ua) for any u∈Σ. So, *aba* is a factor ofh(B). Then the left side of *B* is one or two positions to the right of a hole, horning *B*, which gives us a contradiction, because *B* appears twice.

Case 2 . We have h(A)=ab.

$abcac\diamond abcbacabacabac\overline{ababab}cacbabcbacabcbabcacbac$

There are two occurrences of h(AA) in w′, but, we note that the first time that *ab* appears later so that the letters before it are the same as the letters before either occurrence of abab is further than 4 positions, meaning that h(B) has a factor of *aba* on its right end, requiring that any occurrence ofh(AABA) spans two occurrences of w′, horning *B*.

Case 3 . We have h(A)=ba.

$abcac\diamond abcbacabacabaca\overline{baba}bcacbabcbacabcbabcacbac$

There is a single occurrence of h(AA) in w′. Note that the next occurrence of h(A) cannot work because that would require h(B) to be something that is not compatible with a factor appearing immediately to the left of h(AA). This means that |h(B)|>5 so, h(B) has *aba* within four positions of its right end. Since *aba* does not appear to the right of any occurrence of h(AA), h(AABA) has to span two occurrences of w′.

Case 4 . We have that h(A) contains the substituted *a* and extends at least one position on one side and more than one position on the other side. Either abab or baba is a factor of h(A). But these subwords do not have two non-overlapping occurrences in w′, and they appear nowhere else in θ^{ω}(a) because they have *aba* as a subword. So, h(AA) has to span two occurrences of w′, horning *AA*.

Case 5 . We have that h(A) has the substituted *a* as its first position, |h(A)|>2. Here *aba* is a prefix ofh(A). This means that the other h(A) that makes up h(AA) must either correspond to a different occurrence of w′ in which case we are done, or to one of the other two non-overlapping occurrences of *aba* in the same occurrence of w′ which it clearly cannot.

Case 6 . We have that h(A) has the substituted *a* as its last position, |h(A)|>4. Similarly to the previous case, the suffix *aba* of h(A) has to correspond to the first *aba* in the same occurrence of w′, which can be seen not to work because bcbacaba≠cabacaba so, the other occurrence of h(A) has to be in another occurrence of w′.

Case 7 . We have that h(A)=caba, h(A)=baca, or h(A)=acab.

$abcac\diamond abc\overline{bacabacabacaba}babcacbabcbacabcbabcacbac$

There are two overlapping occurrences of h(AA), there is no occurrence of h(A) to the right of either one by at least one position, until the next occurrence of w'. This means that h(AABA) has to span two occurrences of w'.

Case 8 . We have that h(A) contains a letter of the substituted *aca* and |h(A)|>4. This means thath(A) has *aba* as a factor. If, in the other h(A) that makes up h(AA), this does not correspond to another*aba* in the same occurrence of w' then we are done. Note that the *aba* occurrences created by the substituted *aca* are only a single letter apart, so, because |h(A)|>4, they cannot be the two *aba* occurrences we need for h(AA). Because the *aba* occurrences introduced by the substituted *a* are both followed by a *b* instead of a *c* like the *aba* occurrences introduced by the *aca* , we only have to consider when h(A) ends in the first *a* of the substituted *aca* . We check up to length 9, after which point, it cannot work because the two occurrences of h(A) would be overlapping.

Case 9 . We have that h(A)=abac.

$abcac\diamond abcbac\overline{abacabac}abababcacbabcbacabcbabcacbac$

In this case, abac only occurs twice in w', so, h(AABA) has to overlap with two different occurrences of w'. □

Note that a similar argument to that of Theorem 7 shows that the pattern ABBCBBAB has avoidability index 3, and by divisibility, two more patterns, ABAACAABA and AABACABAA have indices of 3.

5. An algorithm to search for an HD0L system avoiding a given pattern

We describe an algorithm to search for an HD0L system $(\Sigma_1,f,a,\Sigma_2,g)$ that avoids a given pattern *p*. The algorithm works as follows:

- It begins by generating a list of D0L systems using Algorithm 1. Algorithm 1 first generates a list of all full words of a given fixed length that avoid *p* using the backtracking algorithm of [8].

Require: *length* is an integer that must be tuned between potentially missing a D0L system and speed, *mesh* is a list of integers, the lengths at which the candidate D0Ls are tested, p is a pattern, *id* is the identity map
Ensure: program prints each D0L system (Σ_1, f, a) it finds that avoids p within the first $\max\{i \in mesh\}$ letters of its fixed point $f^{\omega}(a)$

```
for all w ∈ backtrack(length) do
    for all f ∈ D0L-for-word(w) do
        for all i ∈ mesh do
            if f^ω(a)[0..i) meets p then
                break to a new f
        if HD0L-checker(f, id) then
            print f
```

Algorithm 1. Generating D0L systems to avoid a pattern

Then, for each of these words, say *w*, it calculates all possible morphisms, say *f*, such thatw∈Pref(f^{ω}(a)) using Algorithm 2. It determines *f* by iterating over all legal lengths of images of letters under *f*, for which *w* uniquely defines the morphism. As *w* is only a finite prefix of f^{ω}(a), the algorithm does not consider many D0Ls which do avoid *p*, but have letter images on the order of or larger than *w*. This restriction also means that, so long as the first letter, w[0], appears somewhere in the image of a letter other than as the first letter of its image, then every letter on which *f* is defined appears infinitely often in f^{ω}(a). At this point, the algorithm has found many thousands of D0Ls which avoid *p* for a finite prefix, but may not avoid *p* in general. Though these could be verified by the HD0L system checking algorithm of [8], it would be entirely unfeasible to check each of these individually. However, checking the length *n* prefix of f^{ω}(a) for an occurrence of *p* takes our algorithm O(n^{l+2})time, where *l* is the number of variables. By continuing to check while letting *n* grow very large, there are multiple rounds of elimination, each one considering longer and longer prefixes. This means that for the longest length prefixes that is checked, very few morphisms are left, offsetting the much greater computational cost for each. Typically by length n=1000, only a handful are left due to the length restriction on the word *w* that is used to generate the morphisms. Once only the morphisms whose fixed point avoid *p* for a very long length are left, the algorithm runs the HD0L system checking algorithm of [8] on these remaining D0Ls to ensure that they avoid *p*. Note that for the computationally complex steps of this procedure, there is very little shared data, and none of it is being modified during those steps, so, concurrency is very good.

Require: w is a prefix of a fixed point of the morphisms we are trying to find, h is a partial function from Σ_1 to Σ_1^+ (initially defined for no $a \in \Sigma_1$), i and j are integers (initially 0)
Ensure: program prints each morphism f it finds that can be uniquely defined by w, up to the lengths of the images of letters; the fixed point of f, with prefix w, contains infinitely many of each letter in Σ_1

```
while j < |w| do
  if w[i] ∈ Domain(h) then
    if h(w[i]) = w[j..j + |h(w[i])|] then
      i ← i + 1
      j ← j + |h(w[i])|
    else
      return
  else
    for k = 1..|w| − j do
      D0L-for-word(w, i ← i + 1, j ← j + k, h ← f where f(x) = h(x) for x ∈ Domain(h) and f(w[i]) = w[j..j + k])
    return
if Domain(h) = Σ1 and there exists a ∈ Σ1 \ {w[0]} such that h(a) = u1 w[0] u2 or h(w[0])[1..|h(w[0])|] = u1 w[0] u2 then
  print h
return
```

Algorithm 2. Generating morphisms with given fixed point

To generate an HD0L system avoiding p, it first runs the D0L generation algorithm on an alphabet of a greater size, since the inner morphism must avoid p on its own if there is any hope of the HD0L system avoiding p. It then, using Algorithm 3, separately generates outer morphisms by generating a set of long "seed" words with holes avoiding p using a modification of the backtracking algorithm in which it starts with a hole in the middle and tries to add letters alternating sides. If in this generation phase, it is unable to add any letter to one side, then p is not avoidable by infinitely many holes. Each seed word w is paired with each D0L morphism, say f. By iterating image sizes for the letters of w, an outer morphism g is determined such that w is a finite prefix of $g \circ f^{\omega}(a)$.

Require: *length* is an integer that must be tuned between potentially missing a HD0L system and speed, *mesh* is a list of integers, the lengths at which the candidate HD0Ls are tested, p is a pattern
Ensure: program prints each HD0L system $(\Sigma_1, f, a, \Sigma_2, g)$ it finds that avoids p within the first $\max\{i \in mesh\}$ letters of its fixed point $f^{\omega}(a)$

```
for all w ∈ randomizedBacktrack(length) do
  for all f ∈ D0L-for-pattern(p) do
    for all g ∈ HD0L-for-word(w, f) do
      for all i ∈ mesh do
        if g(f^ω(a))[0..i) meets p then
          break to a new f
      if HD0L-checker(f, g) then
        print f, g
```

Algorithm 3. Generating HD0L systems to avoid a pattern

Then, it applies a refining procedure similar to the D0L case, Algorithm 4, in which a longer and longer prefix of $g \circ f^{\omega}(a)$ is checked for an occurrence of p. After greatly reducing the number of HD0L systems it has generated, it verifies those remaining with the partial word HD0L system checking algorithm described in [3]. Note, in order to assure that the generated HD0L system contains infinitely many holes, it suffices to know that the seed word contains at least (in practice, exactly) one hole, meaning that the image on one of the letters in the inner alphabet Σ_1 contains at least one hole, and that every letter of the underlying D0L system occurs infinitely often.

```
Require: (Σ_1, f, a) is a D0L system, w is a word, h is a partial function from Σ_1 to Σ_2^+, i and j are integers (initially 0)
Ensure: program prints each HD0L system (Σ_1, f, a, Σ_2, g) it finds such that g(f^ω(a)) has prefix w
while j < |w| do
    if f^ω(a)[i] ∈ Domain(h) then
        if h(f^ω(a)[i]) = w[j..j + |h(f^ω(a)[i])|] then
            i ← i + 1
            j ← j + |h(f^ω(a)[i])|
        else
            return
    else
        for k = 1..|w| − j do
            D0L-for-word(w, i ← i + 1, j ← j + k, h ← g where g(x) = h(x) for x ∈ Domain(h) and g(f^ω(a)[i]) = w[j..j + k])
        return
if Domain(h) = Σ_1 then
    print h
return
```
Algorithm 4. Generating HD0L systems to avoid a pattern, given a D0L system and a prefix word

6. Concluding remarks, conjectures, and open problems

Referring to our ternary lexicon in Appendix A, our results have shown that there are 69 unavoidable patterns, 4 patterns with avoidability index 5, 24 patterns with index 4, between 209 and 216 patterns with index 3, and all other patterns have index 2. In particular, all patterns with length greater than 11 have index 2. Since patterns have the same index as their reverse, this reduces to having to find 4 more patterns' avoidability indices: ABACBC, AABCCAB, AABACABBA, and ABBACAABA.

A few comments on the remaining 4 patterns: We know by 1-deep backtracking that any infinite binary word with infinitely many holes that avoids AABACABBA or ABBACAABA must have only finitely many holes that are not surrounded by either

$$baaabbbaabababbbaaabbbaababbbaabbbaabaaabbbaaba-\\ bbbaaabbbaaba\diamond abbbaaabbbaababbbaabbbaabaaabbbaa$$

or

babaaabbbaaabbaba⋄abbaaabbababaaabbbaaabbabab

up to swapping *a* and *b*. For the pattern ABACBC, we similarly get that only finitely many holes cannot be surrounded by aaa⋄bbb up to swapping *a* and *b*.

Note that there are ternary patterns with avoidability index 4 or 5 for partial words (for instance,AABACAAB has index 4 and AABCABA has index 5), while no such ternary pattern exists for full words[8]. Indeed, to our knowledge the only known patterns with an avoidability index of 4 for full words require at least seven variables (for instance, the pattern ABDACEBAFCAGCB from [13]), and the only known patterns with an index of 5 for full words require at least nine variables (for instance, the patternABVACWBAXBCYCDAZDCD from [9]).

Frequently, the lower bound is provided by Theorem 2 from patterns of known depth. The conditions on Theorem 2 can most likely be significantly weakened. We conjecture in particular that if p is k -shallow andp_0 and p_1 are (h_0,k)-deep and (h_1,k)-deep respectively, then p_0Ap_1 is (h_0+h_1,k)-deep. In general, what relation does the depth of p_0Ap_1 have with the depth of p_0 and p_1? Classification of the depths of patterns may give insight.

Every 0-deep pattern that is unavoidable may be seen to be written in the form of Corollary 2. We conjecture that every unavoidable pattern may be written in this form and that Corollary 2 may be implemented into an algorithm which decides the partial word avoidability of a pattern. We believe the sequence of Corollary 1 has maximal length 2-unavoidable pattern p_n with $|p_n|=3\times2^{n-1}-1$. This would mean that any classification of the patterns using k variables by our method would need never explicitly calculate morphisms for any pattern $3\times2^{k-1}$ or longer.

In addition, a World Wide Web server interface at

www.uncg.edu/cmp/research/patterns2

has been established for automated use of our Pattern Avoidance Automated Archive. Given as input a pattern over any alphabet of variables, the Archive attempts to determine the avoidability index or bounds of it, using the algorithms described in our paper. The Archive first checks for unavoidability. If no reason to suspect unavoidability is found, it attempts to generate HD0Ls which avoid it. Note that the HD0L finder is not implemented for patterns with more than three distinct variables. Suggested HD0Ls are also output, and can be verified using our HD0L verification algorithm found there.

Appendix A.

- A note on reading the following classification:
- Patterns are not listed if the canonical form of their reverse is listed, as the two have equivalent avoidability indices;
- Oftentimes, the upper bound is gained implicitly from an upper bound of a prefix/reverse of a prefix;
- Backtracking means that the lower bound was obtained because for smaller alphabet sizes, there were only finitely many words that avoided the pattern;
- The full word case gave us a lower bound for many of the patterns, because introducing infinitely many holes can only cause more occurrences of the pattern;

- In reading the HD0L's used to get an upper bound, f represents the inner morphism and g the outer one, so that $g \circ f^{\omega}(a)$ avoids the pattern (a tuple notation is used, where the image of *a* is the first element of the tuple, image of *b* the second, etc.);

- When a pattern is given as a reason for an upper bound, it means the current pattern is divisible by the given one;
- Any pattern that is neither listed nor has a listed reverse is 2-avoidable, according to division as explained in Section 3;

All indices of ∞ are determined by Corollary 2.

1.A ∞
2.AA ∞
3.AAB ∞
4.AABA ∞
5.AABAA ∞
6.AABAAB 2 Upper: ABAAB
7.AABAAC ∞
8.AABAACA ∞
9.AABAACAA ∞
10.AABAACAAB 3 Lower: Theorem 2 Upper: Theorem 4
11.AABAACAABA 3 Lower: Theorem 2 Upper: Theorem 4
12.AABAACAABAA 3 Lower: Theorem 2 Upper: Theorem 4
13.AABAACAABAAB 2 Upper: AABAAB
14.AABAACAABAAC 2 Upper: ABBCABBC
15.AABAACAABAB 2 Upper: AABAACABAB
16.AABAACAABAC 2 Upper: ABBCABC
17.AABAACAABB 2 Upper: AABAACABB
18.AABAACAABC 2 Upper: ABAACABC
19.AABAACAAC 2 Upper: AABAAB
20.AABAACAB 3 Lower: Theorem 2 Upper: Theorem 4
21.AABAACABA 3 Lower: Theorem 2 Upper: Theorem 4
22.AABAACABAA 3 Lower: Theorem 2 Upper: Theorem 4
23.AABAACABAAB 2 Upper: AABAACBAAB
24.AABAACABAAC 2 Upper: ABBCABBC
25.AABAACABAB 2 Upper: AABAACBAB
26.AABAACABAC 2 Upper: ABBCABC
27.AABAACABB 2 Upper: AABAACBB
28.AABAACABC 2 Upper: ABAACABC
29.AABAACAC 3 Lower: Full word case Upper: ABAB
30.AABAACACA 2 Upper: AABABA
31.AABAACACB 2 Upper: ABAACACB

32.AABAACACC 2 Upper: AABABB
33.AABAACB 3 Lower: Theorem 2 Upper: Theorem 4
34.AABAACBA 3 Lower: Theorem 2 Upper: Theorem 4
35.AABAACBAA 3 Lower: Theorem 2 Upper: Theorem 4
36.AABAACBAAB 2 Upper:g=(baa,aabb⋄,ababab),f=(acb,c,ab)
37.AABAACBAAC 2 Upper: AABACBAC
38.AABAACBAB 2 Upper:g=(aabab,b⋄a,aabb),f=(acb,c,ab)
39.AABAACBAC 2 Upper: ABBCABC
40.AABAACBB 2 Upper:g=(ba,abb⋄aabab),f=(ab,ba)
41.AABAACBC 2 Upper:g=(abaaaa⋄bbaa,bab),f=(ab,ba)
42.AABAACC 3 Lower: Full word case Upper: AABB
43.AABAACCA 2 Upper: AABBA
44.AABAACCB 2 Upper:g=(ab,baba⋄a,bbbaa),f=(acb,c,ab)
45.AABAB 3 Lower: Full word case Upper: ABAB
46.AABABA 2 Upper: AABCBC
47.AABABB 2 Upper:g=(ab,⋄baa),f=(abbb,a)
48.AABABC 3 Lower: Full word case Upper: ABAB
49.AABABCA 3 Lower: Full word case Upper: ABAB
50.AABABCAA 3 Lower: Full word case Upper: ABAB
51.AABABCAAB 3 Lower: Full word case Upper: ABAB
52.AABABCAABA 3 Lower: Full word case Upper: ABAB
53.AABABCAABAA 2 Upper: AABABCABAA
54.AABABCAABAB 3 Lower: Full word case Upper: ABAB
55.AABABCAABABA 2 Upper: AABABA
56.AABABCAABABB 2 Upper: AABABB
57.AABABCAABABC 2 Upper: ABBCABBC
58.AABABCAABAC 2 Upper: ABCAABAC
59.AABABCAABB 2 Upper: AABABCBB
60.AABABCAABC 2 Upper: ABBCABC
61.AABABCAAC 2 Upper: ABABCAAC
62.AABABCAB 3 Lower: Full word case Upper: ABAB
63.AABABCABA 3 Lower: Full word case Upper: ABAB
64.AABABCABAA 2 Upper:g=(aabaaa⋄bb,b,aba),f=(acb,c,ab)
65.AABABCABAB 3 Lower: Full word case Upper: ABAB
66.AABABCABABA 2 Upper: AABABCBABA
67.AABABCABABB 2 Upper: AABACACC
68.AABABCABABC 2 Upper: AABCABC
69.AABABCABAC 2 Upper: ABABCABAC
70.AABABCABB 2 Upper: AABABCBB
71.AABABCABC 2 Upper: ABABCABC

72.AABABCAC 2 Upper: ABABCAC
73.AABABCB 3 Lower: Full word case Upper: ABAB
74.AABABCBA 3 Lower: Full word case Upper: ABAB
75.AABABCBAA 3 Lower: Full word case Upper: ABAB
76.AABABCBAAB 2 Upper: ABABCBAAB
77.AABABCBAAC 2 Upper: ABACABBC
78.AABABCBAB 3 Lower: Full word case Upper: ABAB
79.AABABCBABA 2 Upper: ABABCBABA
80.AABABCBABB 2 Upper: AABACBABB
81.AABABCBABC 2 Upper: AABACBAC
82.AABABCBAC 2 Upper: ABABCBAC
83.AABABCBB 2 Upper:g=(aabaa⋄bb,abb),f=(abbb,a)
84.AABABCBC 2 Upper: ABABCBC
85.AABABCC 2 Upper: ABABCC
86.AABAC ∞
87.AABACA ∞
88.AABACAA ∞
89.AABACAAB 4 Lower: Theorem 2 Upper: Theorem 4
90.AABACAABA 4 Lower: Theorem 2 Upper: Theorem 4
91.AABACAABAA 3 reverse of AABAACABAA
92.AABACAABAAB 2 Upper: AABAAB
93.AABACAABAAC 2 Upper: ABCAABAC
94.AABACAABAB 3 Lower: Full word case Upper: ABAACAC
95.AABACAABABA 2 Upper: AABABA
96.AABACAABABB 2 Upper: AABABB
97.AABACAABABC 2 Upper: ABACABBC
98.AABACAABAC 2 Upper: ABCBBABC
99.AABACAABB 3 Lower: only 313 binary words with hole in middle avoid AABA.AABB of longest length 33 Upper: AABB
100.AABACAABBA 2 Upper: AABBA
101.AABACAABBC 2 Upper: ABCBBAAC
102.AABACAABC 2 Upper: ABACAABC
103.AABACAAC 2 Upper: ABAAB
104.AABACAB 4 Lower: Theorem 2 Upper: Theorem 4
105.AABACABA 4 Lower: Theorem 2 Upper: Theorem 4
106.AABACABAA 3 Lower: Theorem 2 Upper: AABACBAA
107.AABACABAAB 2 Upper: ABAAB
108.AABACABAAC 2 Upper: ABACBAAC
109.AABACABAB 3 Lower: Full word case Upper: AABACAC
110.AABACABABA 2 Upper: AABACACA

111.AABACABABB 2 reverse of AABABCBABB
112.AABACABABC 2 Upper: ABCBABAC
113.AABACABAC 2 Upper: AABCABC
114.AABACABB 3 Lower: Full word case Upper: AABACBB
115.AABACABBA either 2 or 3 Upper: AABACBBA
116.AABACABBAA 2 Upper: ABBAA
117.AABACABBAB 2 Upper: ABBAB
118.AABACABBAC 2 Upper: AABCABBC
119.AABACABBC 2 Upper: ABACABBC
120.AABACABC 2 reverse of ABCACBCC
121.AABACAC 3 Lower: Full word case Upper: ABAB
122.AABACACA 2 Upper: ABABA
123.AABACACB 2 Upper:g=(abbb,aaa⋄bbaba,bba),
f=(acb,c,ab)
124.AABACACC 2 reverse of AABABCBB
125.AABACB 4 Lower: Theorem 2 Upper: Theorem 4
126.AABACBA 4 Lower: Theorem 2 Upper: Theorem 4
127.AABACBAA 3 Lower: Theorem 2 Upper: Special argument
128.AABACBAAB 3 Lower: Full word case Upper: ABBA
129.AABACBAABA 2 Upper: ABBAB
130.AABACBAABB 2 Upper: ABBAA
131.AABACBAABC 2 Upper: ABCABBAC
132.AABACBAAC 2 Upper: ABACBAAC
133.AABACBAB 3 Lower: Full word case Upper: HD0L:g=(bb,a⋄c,cba),f=(acb,c,ab)
134.AABACBABA 3 Lower: only 199 binary words with hole in middle avoid AABA.BABA of longest length 20 Upper: AABACBAB
135.AABACBABAA 2 reverse of AABABCABAA
136.AABACBABAB 2 Upper: ABABA
137.AABACBABAC 2 Upper: AABCBABC
138.AABACBABB 2 HD0L:g=(abbabb⋄a,bbbaa),f=(ab,aa)
139.AABACBABC 2 Upper: ABACBABC
140.AABACBAC 2 Upper: AABCBC
141.AABACBB 3 Lower: Full word case Upper: HD0L:g=(b,a,b⋄aac),f=(acb,c,ab)
142.AABACBBA 3 Lower: Full word case Upper: AABACBB
143.AABACBBAA 2 Upper: Equivalent to AABBCBBAB
144.AABACBBAB 2 HD0L:g=(abb,⋄aabb),f=(ab,baab)
145.AABACBBAC 2 Upper: ABAAB
146.AABACBBC 2 Upper: ABACBBC
147.AABACBC 2 HD0L:g=(aaab,bbb⋄aa,bababa),f=(acb,c,ab)
148.AABACC ∞

149.AABACCA 3 Lower: Full word case Upper: ABBA
150.AABACCAA 2 Upper: ABBAA
151.AABACCAB 2 HD0L:g=(aab,abab⋄baaa,bba),f=(ac,ca,ba)
152.AABACCAC 2 Upper: ABBAB
153.AABACCB 2 HD0L:g=(ababb,aaaabbba,ba⋄bba),f=(ab,ca,ba)
154.AABB 3 Lower: Full word case HD0L:g=(abac,⋄c,babc),f=(acb,c,ab)
155.AABBA 2 Upper: Theorem 1
156.AABBC 3 Lower: Full word case
157.AABBCA 3 Lower: Full word case Upper: AABB
158.AABBCAA 3 Lower: Full word case
159.AABBCAAB 3 Lower: Full word case
160.AABBCAABA 3 Lower: only 313 binary words with hole in middle avoid AABA.AABB of longest length 33
161.AABBCAABAA 2 Upper: AABCCACC
162.AABBCAABAB 2 Upper: AABBCABAB
163.AABBCAABAC 2 Upper: AABCCACB
164.AABBCAABB 3 Lower: Full word case
165.AABBCAABBA 2 Upper: AABBA
166.AABBCAABBC 2 Upper: ABBCABBC
167.AABBCAABC 2 Upper: ABBCABC
168.AABBCAAC 2 HD0L: g=(abba,baa⋄aaabbbab),f=(ab,ba)
169.AABBCAB 3 Lower: Full word case
170.AABBCABA 3 Lower: only 215 binary words with hole in middle avoid AABB.ABA of longest length 33
171.AABBCABAA 2 reverse of AABACBBAA
172.AABBCABAB 2 HD0L: g=(aaab,b⋄abbaab), f=(ab,aa)
173.AABBCABAC 2 Upper: AABCACB
174.AABBCABB 3 Lower: Full word case
175.AABBCABBA 3 Lower: Only 219 binary words simultaneously
avoid {AABBA,AAAA} with hole in ninth position
176.AABBCABBAA 2 Upper: ABBAA
177.AABBCABBAB 2 Upper: ABBAB
178.AABBCABBAC 2 Upper: AABCAACB
179.AABBCABBC 2 Upper: ABBCABBC
180.AABBCABC 2 Upper: ABBCABC
181.AABBCAC 2 HD0L: g=(aabba⋄babbb,aaab,baabab),f=(abc,ac,b)
182.AABBCB 3 Lower: Full word case
183.AABBCBA 3 Lower: Full word case
184.AABBCBAA 3 Lower: Full word case
185.AABBCBAAB 3 Lower: Only 220 binary words simultaneously

avoid {ABBAA,AAAA} with hole in tenth position
186.AABBCBAABA 2 Upper: ABBAB
187.AABBCBAABB 2 Upper: ABBAA
188.AABBCBAABC 2 Upper: AABACCAB
189.AABBCBAAC 2 Upper: ABBCBAAC
190.AABBCBAB 3 Lower: only 223 binary words with hole in middle avoid AABB.BAB of longest length 34
191.AABBCBABA 2 HD0L: g=(bbabb⋄aa,bbab), f=(abb,a)
192.AABBCBABB 3 reverse of AABACAABB
193.AABBCBABBA 2 Upper: ABAAB
194.AABBCBABBC 2 Upper: AABCBABC
195.AABBCBABC 2 Upper: AABCBAC
196.AABBCBAC 2 Upper: ABBCBAC
197.AABBCBB 3 reverse of AABAACC
198.AABBCBBA 3 Lower: Full word case
199.AABBCBBAA 2 HD0L: g=(abaa,ba⋄babbbaa), f=(aba,abb)
200.AABBCBBAB 2 HD0L: g=(aaba,bbabbab, aaba⋄babbabbbab), f=(ab,ca,ba)
201.AABBCBBAC 2 Upper: AABCBAC
202.AABBCBBC 2 Upper: AABCBC
203.AABBCBC 2 HD0L: g=(abab,abbb⋄aaabbb,aba), f=(acb,c,ab)
204.AABBCC 3 Lower: Only 226 avoiding binary words with hole in eighth position
205.AABBCCA 2 reverse of ABBCCAA
206.AABBCCB 2 Upper: AABBA
207.AABC ∞
208.AABCA ∞
209.AABCAA ∞
210.AABCAAB ∞
211.AABCAABA 4 Lower: Theorem 2 Upper: Theorem 4
212.AABCAABAA 3 reverse of AABAACBAA
213.AABCAABAAB 2 Upper: AABAAB
214.AABCAABAAC 2 Upper: ABCABAAC
215.AABCAABAB 3 Lower: Full word case Upper: AABAB
216.AABCAABABA 2 Upper: AABABA
217.AABCAABABB 2 Upper: AABABB
218.AABCAABABC 2 Upper: ABCCACAB
219.AABCAABAC 2 Upper: ABCAABAC
220.AABCAABB 3 reverse of AABBCABB
221.AABCAABBA 2 Upper: AABBA
222.AABCAABBC 2 Upper: ABCAABBC
223.AABCAABC 2 Upper: AABAAB

224.AABCAAC 3 Lower: Full word case Upper: ABBA
225.AABCAACA 2 Upper: ABBAB
226.AABCAACB 2 HD0L:g=(bab,abbb⋄bba,abaa),f=(acb,c,ab)
227.AABCAACC 2 Upper: ABBAA
228.AABCAB ∞
229.AABCABA 5 Lower: special argument Upper: special argument
230.AABCABAA 3 reverse of AABACBAA
231.AABCABAAB 2 Upper: ABAAB
232.AABCABAAC 2 Upper: ABCABAAC
233.AABCABAB 3 Lower: Full word case Upper: AABACAC
234.AABCABABA 2 Upper: AABACACA
235.AABCABABB 2 reverse of AABABCABB
236.AABCABABC 2 Upper: ABCACAB
237.AABCABAC 2 HD0L:g=(aaba,aaa⋄bab,bbba),f=(acb,c,ab)
238.AABCABB ∞
239.AABCABBA 3 Lower: Full word case Upper: AABACCA
240.AABCABBAA 2 Upper: ABBAA
241.AABCABBAB 2 Upper: ABBAB
242.AABCABBAC 2 Upper: ABCABBAC
243.AABCABBC 2 Upper: ABCABBC
244.AABCABC 2 Upper: ABCABC
245.AABCAC ∞
246.AABCACA 3 Lower: Full word case Upper: ABAB
247.AABCACAA 3 reverse of AABABCAA
248.AABCACAAB 2 Upper: ABCACAAB
249.AABCACAAC 2 Upper: ABABBA
250.AABCACAB 2 Upper: ABCACAB
251.AABCACAC 2 Upper: ABABA
252.AABCACB 2 Upper: g=(bbbbabbaaaaa,abab,bbbbbabaaa⋄a),f=(acb,c,ab)
253.AABCACC 3 reverse of AABACBB
254.AABCACCA 2 Upper: ABAAB
255.AABCACCB 2 Upper: ABCACCB
256.AABCB ∞
257.AABCBA ∞
258.AABCBAA 4 Lower: only 94 ternary words with hole in middle
avoid {AAA,AABAA,ABBA} of longest length 10 Upper: special argument, similar to Theorem
4
259.AABCBAAB 3 Lower: Full word case Upper: ABBA
260.AABCBAABA 2 Upper: ABBAB
261.AABCBAABB 2 Upper: ABBAA

262.AABCBAABC 2 Upper: ABCBAABC
263.AABCBAAC 2 Upper: ABCBAAC
264.AABCBAB ∞
265.AABCBABA 3 Lower: Full word case Upper: CBABA
266.AABCBABAA 3 reverse of AABABCBAA
267.AABCBABAAB 2 Upper: ABABBA
268.AABCBABAAC 2 Upper: ABACACCB
269.AABCBABAB 2 Upper: ABABA
270.AABCBABAC 2 Upper: ABCBABAC
271.AABCBABB 3 reverse of AABACABB
272.AABCBABBA 2 Upper: ABAAB
273.AABCBABBC 2 Upper: ABCBABBC
274.AABCBABC 2 Upper: ABCBABC
275.AABCBAC 2 HD0L:g=(abbbb,aaaa⋄abb,bbabaab),f=(acb,c,ab)
276.AABCBB ∞
277.AABCBBA 3 Lower: Full word case HD0L:g=(a,b,bc⋄abac),f=(ac,ca,ba)
278.AABCBBAA 3 reverse of AABBCBAA
279.AABCBBAAB 2 Upper: AABBA
280.AABCBBAAC 2 Upper: ABCBBAAC
281.AABCBBAB 3 Lower: Full word case
282.AABCBBABA 2 Upper: AABCCACA
283.AABCBBABB 2 reverse of AABAACABB
284.AABCBBABC 2 Upper: ABCBBABC
285.AABCBBAC 2 HD0L:g=(ababa,a⋄bb,abbaa),f=(ab,c,ca)
286.AABCBBC 2 Upper: ABAAB
287.AABCBC 2 Upper: ABCBC HD0L:g=(abbbaa,abbabaaa,abbaababb⋄a),f=(acb,c,ab)
288.AABCC ∞
289.AABCCA ∞
290.AABCCAA 3 reverse of AABBCAA
291.AABCCAAB 2 HD0L:g=(abba,abaa⋄bbbb,bbab),f=(acb,c,ab)
292.AABCCAAC 2 Upper: AABBA
293.AABCCAB either 2 or 3 Upper: ABBA
294.AABCCABA 2 Upper: ABCCABA
295.AABCCABB 2 HD0L:g=(ababa⋄bbb,baa),f=(abaa,bbab)
296.AABCCABC 2 Upper: ABBAB
297.AABCCAC 3 Lower: Full word case HD0L:g=(ab,c⋄,bba),f=(acb,c,ab)
298.AABCCACA 2 reverse of ABABBCAA
299.AABCCACB 2 HD0L:g=(aaabab⋄aab,aabb),f=(abb,a)
300.AABCCACC 2 reverse of AABAACBB
301.AABCCB 3 Lower: Full word case Upper: ABBA

302.AABCCBA 2 HD0L:g=(aab,abbb,aba,ab⋄aaabbbbbabba),f=(abc,d,cba,b)
303.AABCCBB 2 Upper: ABBAA
304.AABCCBC 2 Upper: ABBAB
305.AB ∞
306. ABA ∞
307.ABAA ∞
308.ABAAB 2 Theorem 1
309.ABAAC ∞
310.ABAACA ∞
311.ABAACAA ∞
312.ABAACAAB 3 Lower: Theorem 2 Upper: Equivalent to AABCABAA
313.ABAACAABA 3 Lower: Theorem 2 Upper: ABAACAAB
314.ABAACAABAA 3 reverse of AABAACAABA
315.ABAACAABAAB 2 Upper: AABAAB
316.ABAACAABAAC 2 Upper: ABCBBABC
317.ABAACAABAB 2 Upper: HD0L: g=(aabaaa⋄bb,b,aba), f=(acb,c,ab)
318.ABAACAABAC 2 Upper: ABACAABC
319.ABAACAABB 2 reverse of AABBCBBAB
320.ABAACAABC 2 Upper: ABBCBBAC
321.ABAACAAC 2 Upper: AABAAB
322.ABAACAB 4 Lower: Theorem 2 Upper: Theorem 4
323.ABAACABA 4 Lower: Theorem 2 Upper: Theorem 4
324.ABAACABAA 4 reverse of AABACAABA
325.ABAACABAAB 2 Upper: ABAAB
326.ABAACABAAC 2 Upper: ABBCABBC
327.ABAACABAB 3 Lower: only 267 binary words with hole in middle avoid ABAA.ABAB of longest length 19 Upper: ABAB
328.ABAACABABA 2 Upper: AABACACA
329.ABAACABABB 2 Upper: AABACACC
330.ABAACABABC 2 Upper: AABACACB
331.ABAACABAC 2 Upper: ABBCABC
332.ABAACABB 3 reverse of AABCBBAB
333.ABAACABBA either 2 or 3 reverse of ABAACABBA
334.ABAACABBAA 2 Upper: ABBAA
335.ABAACABBAB 2 Upper: ABBAB
336.ABAACABBAC 2 Upper: ABACABBC
337.ABAACABBC 2 Upper: ABBCBAAC
338.ABAACABC 2 Upper: ABBCBAC
339.ABAACAC 3 Lower: Full word case Upper: ABAB
340.ABAACACA 2 Upper: AABABA

341.ABAACACB 2 Upper: g=(aabbb,⋄bab,baa),f=(acb,c,ab)
342.ABAACACC 2 Upper: AABABB
343.ABAACB 4 Lower: Theorem 2 Upper: Theorem 4
344.ABAACBA 4 Lower: Theorem 2 Upper: Theorem 4
345.ABAACBAA 4 reverse of AABCAABA
346.ABAACBAAB 3 Lower: Full word case Upper: ABBA
347.ABAACBAABA 2 Upper: ABBAB
348.ABAACBAABB 2 Upper: ABBAA
349.ABAACBAABC 2 Upper: AABCAACB
350.ABAACBAAC 2 Upper: ABBCABBC
351.ABAACBAB 3 Lower: Full word case HD0L:g=(cabb,cbbb,a⋄a),f=(acb,c,ab)
352.ABAACBABA 3 Lower: Theorem 2
353.ABAACBABAA 3 reverse of AABABCAABA
354.ABAACBABAAB 2 Upper: ABABBA
355.ABAACBABAAC 2 Upper: ABCACAAB
356.ABAACBABAB 2 Upper: ABABA
357.ABAACBABAC 2 Upper: ABACBABC
358.ABAACBABB 2 reverse of AABACBBAB
359.ABAACBABC 2 Upper: AABCACB
360.ABAACBAC 2 Upper: ABBCABC
361.ABAACBB 3 reverse of AABCCAC
362.ABAACBBA 3 Lower: Full word case
363.ABAACBBAA 3 reverse of AABBCAABA
364.ABAACBBAAB 2 Upper: AABBA
365.ABAACBBAAC 2 Upper: AABCCAAB
366.ABAACBBAB 2 HD0L:g=(abbabb⋄a,bbbaa),f=(ab,aa)
367.ABAACBBAC 2 Upper: ABBCAABC
368.ABAACBBC 2 HD0L:g=(aabab⋄,aabbba),f=(ab,baa)
369.ABAACBC 2 HD0L:g=(abb,bb⋄aabab,bbbbaaaa),f=(abc,ac,b)
370.ABAACC 3 reverse of AABBCB
371.ABAACCA 2 Upper: AABBA
372.ABAACCB 2 HD0L:g=(aab⋄babaaabba,baabba,ababba,abbba),f=(ab,cd,ad,cb)
373.ABAACCBA 2 HD0L:g=(bbbb,bba,aa⋄aabaaba),f=(abc,ac,b)
374. ABAACCBB 2 Upper: ABBCCAA
375.ABAACCBC 2 HD0L: g=(abaabba⋄,ababbba), f=(ab,ba)
376.ABAB 3 Lower: Full word case HD0L: g=(aa,b⋄,cca), f=(acb,c,ab)
377.ABABA 2 Upper: ABABC HD0L: g=(abaa,aabba,aababbbb,aabbb⋄a), f=(ab,dc,ac,ad)
378.ABABB 3 reverse of AABAB
379.ABABBA 2 Upper: ABAAB
380.ABABBC 3 Lower: Full word case

381.ABABBCA 3 Lower: Full word case Upper: ABAB
382.ABABBCAA 2 Upper:g=(aaabb,a⋄baabb), f=(ab,aa)
383.ABABBCAB 3 Lower: Full word case
384.ABABBCABA 3 Lower: Full word case
385.ABABBCABAA 2 Upper: ABABBCAA
386.ABABBCABAB 3 Lower: Full word case
387.ABABBCABABA 2 Upper: ABABBCBABA
388.ABABBCABABB 3 reverse of AABABCAABAB
389.ABABBCABABBA 2 Upper: ABABCABBA
390.ABABBCABABBC 2 Upper: ABCACAAB
391.ABABBCABABC 2 Upper: ABBCABC
392.ABABBCABAC 2 Upper: AABCACB
393.ABABBCABB 3 reverse of AABCAABAB
394.ABABBCABBA 2 Upper: ABABCABBA
395.ABABBCABBC 2 Upper: ABBCABBC
396.ABABBCABC 2 Upper: ABBCABC
397.ABABBCAC 2 HD0L:g=(abb,aab⋄babbbaa), f=(ab,bbaa)
398.ABABBCB 3 reverse of ABAACAC
399.ABABBCBA 3 Lower: Full word case
400.ABABBCBAA 2 reverse of AABCBBABA
401.ABABBCBAB 3 Lower: Full word case
402.ABABBCBABA 2 Upper: ABABCBABA
403.ABABBCBABB 3 reverse of AABACAABAB
404.ABABBCBABBA 2 Upper: ABABCABBA
405.ABABBCBABBC 2 Upper: ABABCBABC
406.ABABBCBABC 2 Upper: ABBCABC
407.ABABBCBAC 2 Upper: ABAACABC
408.ABABBCBB 3 reverse of AABAACAC
409.ABABBCBBA 3 Lower: Full word case
410.ABABBCBBAA 2 Upper: ABABBCBAA
411.ABABBCBBAB 2 reverse of ABAACAABAB
412.ABABBCBBAC 2 Upper: ABBCBBAC
413.ABABBCBBC 2 Upper: AABAAB
414.ABABBCBC 2 Upper: AABCBC
415.ABABBCC 2 reverse of AABBCBC
416.ABABC 3 Lower: Full word case
417.ABABCA 3 Lower: Full word case
418.ABABCAA 3 reverse of AABCACA
419.ABABCAAB 3 Lower: Full word case
420.ABABCAABA 3 reverse of ABAACBABA

421.ABABCAABAA 2 Upper: ABACBBABB
422.ABABCAABAB 3 reverse of ABABBCABAB
423.ABABCAABABA 2 Upper: AABABA
424.ABABCAABABB 2 Upper: AABABB
425.ABABCAABABC 2 Upper: ABCCACAB
426.ABABCAABAC 2 Upper: ABCAABAC
427.ABABCAABB 2 reverse of AABBCABAB
428.ABABCAABC 2 Upper: ABAAB
429.ABABCAAC 2 Upper: ABACBBC
430.ABABCAB 3 Lower: Full word case
431.ABABCABA 3 Lower: Full word case
432.ABABCABAA 3 reverse of AABACBABA
433.ABABCABAAB 2 Upper: ABABCBAAB
434.ABABCABAAC 2 Upper: ABCABAAC
435.ABABCABAB 3 Lower: Full word case
436.ABABCABABA 2 Upper: ABABA
437.ABABCABABB 3 reverse of AABABCABAB
438.ABABCABABBA 2 Upper: ABABCABBA
439.ABABCABABBC 2 Upper: ABACBABAAC
440.ABABCABABC 2 Upper: AABAAB
441.ABABCABAC 2 Upper: ABACBABC
442.ABABCABB 3 reverse of AABCABAB
443.ABABCABBA 2 HD0L: g=(aaabba⋄b,aaabbb), f=(abb,a)
444.ABABCABBC 2 Upper: ABACBAAC
445.ABABCABC 2 Upper: ABCABC
446.ABABCAC 2 HD0L: g=(aab,bbbb⋄aa,bbabaa), f=(acb,c,ab)
447.ABABCB 3 Lower: Full word case
448.ABABCBA 3 Lower: Full word case
449.ABABCBAA 3 reverse of AABCBABA
450.ABABCBAAB 2 HD0L: g=(aabbba⋄aba,bbba),f=(abb,a)
451.ABABCBAAC 2 Upper: ABACABBC
452.ABABCBAB 3 Lower: Full word case
453.ABABCBABA 2 HD0L: g=(ba,baaab⋄abb,baab), f=(acb,c,ab)
454.ABABCBABB 3 reverse of AABACABAB
455.ABABCBABBA 2 Upper: ABABCABBA
456.ABABCBABBC 2 Upper: ABACBABC
457.ABABCBABC 2 Upper: ABACABAC
458.ABABCBAC 2 reverse of ABCACBCB
459.ABABCBB 3 reverse of AABACAC
460.ABABCBBA 3 Lower: Full word case

461.ABABCBBAA 2 reverse of AABBCBABA
462.ABABCBBAB 3 reverse of ABAACABAB
463.ABABCBBABA 2 reverse of ABABBCBABA
464.ABABCBBABB 2 reverse of AABAACABAB
465.ABABCBBABC 2 Upper: ABCBBABC
466.ABABCBBAC 2 Upper: ABACAABC
467.ABABCBBC 2 Upper: ABAAB
468.ABABCBC 2 HD0L: g=(ab,aaabb⋄aba,bbaa), f=(acb,c,ab)
469.ABABCC 2 reverse of AABCBC
470.ABAC ∞
471.ABACA ∞
472.ABACAA ∞
473.ABACAAB 5 Lower: special argument Upper: special argument
474.ABACAABA 4 reverse of ABAACABA
475.ABACAABAA 3 reverse of AABAACABA
476.ABACAABAAB 2 Upper: AABAAB
477.ABACAABAAC 2 Upper: ABCAABAC
478.ABACAABAB 3 reverse of ABABBCBAB
479.ABACAABABA 2 Upper: AABABA
480.ABACAABABB 2 Upper: AABABB
481.ABACAABABC 2 Upper: ABAACACB
482.ABACAABAC 2 Upper: ABCBBABC
483.ABACAABB 3 reverse of AABBCBAB
484.ABACAABBA 2 Upper: AABBA
485.ABACAABBC 2 Upper: ABCBBAAC
486.ABACAABC 2 HD0L: g=(aaa,abbb⋄,babab), f=(acb,c,ab)
487.ABACAAC 2 Upper: ABAAB
488.ABACAB ∞
489.ABACABA ∞
490.ABACABAA 4 reverse of AABACABA
491.ABACABAAB 2 Upper: ABAAB
492.ABACABAAC 2 Upper: ABCBABBC
493.ABACABAB 3 reverse of ABABCBAB
494.ABACABABA 2 Upper: ABABA
495.ABACABABB 3 reverse of AABABCBAB
496.ABACABABBA 2 Upper: ABABBA
497.ABACABABBC 2 Upper: ABACACCB
498.ABACABABC 2 Upper: ABCBABAC
499.ABACABAC 2 Upper: ABCBABC
500.ABACABB ∞

501.ABACABBA 3 Lower: Full word case Upper: ABBA
502.ABACABBAA 2 Upper: ABBAA
503.ABACABBAB 2 Upper: ABBAB
504.ABACABBAC 2 Upper: ABCBAABC
505.ABACABBC 2 Upper: ABCBAAC
506.ABACABC 2 HD0L: g=(abba,aaaa,bbb, b⋄abaababb), f=(ab,cd,cb,ad)
507.ABACAC 3 reverse of ABABCB
508.ABACACA 2 Upper: ABABA
509.ABACACB 2 HD0L: g=(aabb,aaabbbab,⋄aababbbbbaaa), f=(abc,ac,b)
510.ABACACC 3 reverse of AABABCB
511.ABACACCA 2 Upper: ABABBA
512.ABACACCB 2 HD0L: g=(baabbaa⋄aba,aabbb),f=(ab,ba)
513.ABACB ∞
514.ABACBA ∞
515.ABACBAA 5 reverse of AABCABA
516.ABACBAAB 3 Lower: Full word case Upper: ABBA
517.ABACBAABA 2 Upper: ABBAB
518.ABACBAABB 2 Upper: ABBAA
519.ABACBAABC 2 Upper: ABCABBAC
520.ABACBAAC 2 Upper: ABCABBC
521.ABACBAB 3 Lower: Full word case HD0L: g=(bb,caabc, aab⋄acbaabc,ac), f=(ad,ab,db,c)
522.ABACBABA 3 reverse of ABABCABA
523.ABACBABAA 3 reverse of AABABCABA
524.ABACBABAAB 2 Upper: ABABBA
525.ABACBABAAC 2 Upper: ABCACAAB
526.ABACBABAB 2 Upper: ABABA
527.ABACBABAC 2 Upper: ABCACAB
528.ABACBABB 3 reverse of AABACBAB
529.ABACBABBA 2 Upper: ABAAB
530.ABACBABBC 2 Upper: ABCABAAC
531.ABACBABC 2 HD0L: g=(baa,ab,aa⋄abbbbbb), f=(ac,ba,b)
532.ABACBAC 2 Upper: ABCABC
533.ABACBB ∞
534.ABACBBA ∞
535.ABACBBAA 3 reverse of AABBCABA
536.ABACBBAAB 2 Upper: AABBA
537.ABACBBAAC 2 Upper: ABCAABBC
538.ABACBBAB 3 reverse of ABAACBAB
539.ABACBBABA 3 reverse of ABABBCABA
540.ABACBBABAA 2 Upper: AABABB

541.ABACBBABAB 2 Upper: AABABA
542.ABACBBABAC 2 Upper: ABCCACAB
543.ABACBBABB 2 reverse of AABAACBAB
544.ABACBBABC 2 Upper: ABCAABAC
545.ABACBBAC 2 Upper: ABAAB
546.ABACBBC 2 reverse of ABBACBC
547. ABACBC either 2 or 3 HD0L: g=(ab,ccbbc⋄ca,cc,bbb), f=(ab,cd,cb,ad)
548.ABACBCA 2 HD0L: g=(aaaabb,baa, bb⋄babb,bbaaba), f=(ab,c,d,da)
549.ABACBCB 2 reverse of ABABCAC
550.ABACBCC 2 reverse of AABACBC
551.ABACC ∞
552.ABACCA 3 Lower: Full word case Upper: ABBA
553.ABACCAA 2 Upper: ABBAA
554.ABACCAB 2 HD0L: g=(aaa⋄bababb,bbbaa,babaaa,aabbb), f=(ab,cd,cb,ad)
555.ABACCAC 2 Upper: ABBAB
556.ABACCB 3 Lower: Full word case HD0L:g=(aba,cb⋄cc,abc),f=(acb,c,ab)
557.ABACCBA 2 HD0L: g=(babbaa,bbba,aab⋄abbab), f=(abc,ac,b)
558.ABACCBB 2 reverse of AABBCAC
559.ABACCBC 2 reverse of ABAACBC
560.ABB ∞
561.ABBA 3 backtracking for lower HD0L:g=(bcc,ca⋄abb,cba),f=(acb,c,ab)
562.ABBAA 2 reverse of AABBA
563.ABBAB 2 reverse of ABAAB
564.ABBAC 3 Lower: Full word case
565.ABBACA 3 reverse of ABACCA
566.ABBACAA 3 reverse of AABACCA
567.ABBACAAB 3 Lower: Full word case
568.ABBACAABA either 2 or 3 reverse of ABAACABBA
569.ABBACAABAA 2 Upper: AABCCACC
570.ABBACAABAB 2 Upper: ABBACABAB
571.ABBACAABAC 2 Upper: ABCBBABC
572.ABBACAABB 3 reverse of AABBCBAAB
573.ABBACAABBA 2 Upper: AABBA
574.ABBACAABBC 2 Upper: AABCBBAAC
575.ABBACAABC 2 Upper: AABCBBAC
576.ABBACAAC 2 Upper: ABAAB
577.ABBACAB 3 Lower: Full word case
578.ABBACABA 3 reverse of ABACABBA
579.ABBACABAA either 2 or 3 reverse of AABACABBA
580.ABBACABAAB 2 Upper: ABAAB

581.ABBACABAAC 2 Upper: AABCBABBC
582.ABBACABAB 2 reverse of ABABCBAAB
583.ABBACABAC 2 Upper: ABBCABC
584.ABBACABB 3 reverse of AABCBAAB
585.ABBACABBA 3 Lower: Full word case
586.ABBACABBAA 2 Upper: ABBAA
587.ABBACABBAB 2 Upper: ABBAB
588.ABBACABBAC 2 Upper: AABCBAABC
589.ABBACABBC 2 Upper: ABCBAAC
590.ABBACABC 2 Upper: AABCBAC
591.ABBACAC 2 reverse of ABABCCB
592.ABBACB 3 Lower: Full word case
593.ABBACBA 3 Lower: Full word case
594.ABBACBAA 3 reverse of AABCABBA
595.ABBACBAAB 2 HD0L:g=(bbaab,⋄babaab), f=(aba,ab)
596.ABBACBAAC 2 Upper: AABCABBC
597.ABBACBAB 3 reverse of ABACBAAB
598.ABBACBABA 2 reverse of ABABCABBA
599.ABBACBABB 3 reverse of AABACBAAB
600.ABBACBABBA 2 Upper: ABAAB
601.ABBACBABBC 2 Upper: ABCABAAC
602.ABBACBABC 2 Upper: AABCABAC
603.ABBACBAC 2 Upper: AABCABC
604.ABBACBB 3 reverse of AABCAAC
605.ABBACBBA 3 Lower: Full word case
606.ABBACBBAA 3 reverse of AABBCABBA
607.ABBACBBAAB 2 Upper: ABBACBAAB
608.ABBACBBAAC 2 Upper: ABACBAAC
609.ABBACBBAB 3 reverse of ABAACBAAB
610.ABBACBBABA 2 reverse of ABABBCABBA
611.ABBACBBABB 2 reverse of AABAACBAAB
612.ABBACBBABC 2 Upper: ABCAABAC
613.ABBACBBAC 2 Upper: AABAAB
614.ABBACBBC 2 HD0L: g=(aaabbb,ab⋄baabaaab), f=(abb,aab)
615.ABBACBC 2 Upper: g=(baaab,bbbb⋄aaaa,bbaba), f=(acb,c,ab)
616.ABBACC 3 reverse of AABCCB
617.ABBACCA 2 HD0L:g=(abbbaab,abb⋄baaaba,abbaa), f=(acb,c,ab)
618.ABBACCB 2 HD0L: g=(bab,baaba,baabba⋄baabbababbbaaaa),f=(abc,ac,b)
619.ABBC ∞
620.ABBCA ∞

621.ABBCAA ∞

622.ABBCAAB ∞

623.ABBCAABA 3 reverse of ABAACBBA

624.ABBCAABAA 2 Upper: AABCCACC

625.ABBCAABAB 2 Upper: AABCCACA

626.ABBCAABAC 2 Upper: AABCCACB

627.ABBCAABB 3 reverse of AABBCAAB

628.ABBCAABBA 2 Upper: AABBA

629.ABBCAABBC 2 Upper: AABCCAAB

630.ABBCAABC 2 Upper: ABBA HD0L: g=(baaaaa,b⋄bb,aabba), f=(acb,c,ab)

631.ABBCAAC 2 reverse of ABBACCB

632.ABBCAB ∞

633.ABBCABA ∞

634.ABBCABAA 3 reverse of AABACBBA

635.ABBCABAAB 2 Upper: ABAAB

636.ABBCABAAC 2 Upper: AABCACCB

637.ABBCABAB 3 reverse of ABABCAAB

638.ABBCABABA 2 Upper: ABABA

639.ABBCABABB 3 reverse of AABABCAAB

640.ABBCABABBA 2 Upper: ABABBA

641.ABBCABABBC 2 Upper: ABCACAAB

642.ABBCABABC 2 Upper: ABCACAB

643.ABBCABAC 2 Upper: AABCACB

644.ABBCABB ∞

645.ABBCABBA 3 reverse of ABBACBBA

646.ABBCABBAA 2 Upper: ABBAA

647.ABBCABBAB 2 Upper: ABBAB

648.ABBCABBAC 2 Upper: AABCAACB

649.ABBCABBC 2 Upper: ABCABC

650.ABBCABC 2 HD0L: g=(aaaab,ab⋄bb,bababa), f=(acb,c,ab)

651.ABBCAC 3 reverse of ABACCB

652.ABBCACA 2 reverse of ABABCCA

653.ABBCACB 2 HD0L: g=(bbaaba,bbbb⋄aaaaa,baabba), f=(acb,c,ab)

654.ABBCACC 2 reverse of AABACCB

655.ABBCB ∞

656.ABBCBA ∞

657.ABBCBAA 3 reverse of AABCBBA

658.ABBCBAAB 3 reverse of ABBACAAB

659.ABBCBAABA 2 Upper: ABBAB

660.ABBCBAABB 2 Upper: ABBAA

661.ABBCBAABC 2 Upper: AABACCAB
662.ABBCBAAC 2 reverse of ABBCACCB
663.ABBCBAB 5 reverse of ABACAAB
664.ABBCBABA 3 reverse of ABABCBBA
665.ABBCBABAA 2 Upper: AABACACC
666.ABBCBABAB 2 Upper: AABACACA
667.ABBCBABAC 2 Upper: AABACACB
668.ABBCBABB 4 reverse of AABACAAB
669.ABBCBABBA 2 Upper: ABAAB
670.ABBCBABBC 2 Upper: ABCBABC
671.ABBCBABC 2 HD0L: g=(ababab,a⋄aaaa,bbb), f=(acb,c,ab)
672.ABBCBAC 2 HD0L: g=(ababbb,aaaaa⋄bbbbbaa,baab), f=(acb,c,ab)
673. ABBCBB ∞
674.ABBCBBA 4 Lower: only 94 ternary words with hole in middle
avoid {AAA,AABAA,ABBA} of longest length 10 Upper: special argument, similar to Theorem 4
675.ABBCBBAA 3 reverse of AABBCBBA
676.ABBCBBAAB 2 Upper: AABBA
677.ABBCBBAAC 2 Upper: AABAACCB
678.ABBCBBAB 3 reverse of ABAACAAB
679.ABBCBBABA 3 reverse of ABABBCBBA
680.ABBCBBABAA 2 Upper: AABABB
681.ABBCBBABAB 2 Upper: AABABA
682.ABBCBBABAC 2 Upper: AABAACACB
683.ABBCBBABB 3 reverse of AABAACAAB
684.ABBCBBABBA 2 Upper: AABAAB
685.ABBCBBABBC 2 Upper: ABCBBABC
686.ABBCBBABC 2 HD0L: g=(abbba⋄,aba,bba), f=(abc,b,aba)
687.ABBCBBAC 2 HD0L: g=(baaa,b,ba⋄bbbaaba), f=(abc,ac,b)
688.ABBCBBC 2 Upper: AABAAB
689.ABBCBC 3 reverse of ABABBC
690.ABBCBCA 2 HD0L: g=(bbaababa,baa⋄b,baaaabbaba), f=(acb,c,ab)
691.ABBCBCB 2 Upper: AABABA
692.ABBCBCC 2 Upper: AABABB
693.ABBCC 3 reverse of AABBC
694.ABBCCA 3 Lower: Full word case
695.ABBCCAA 2 Upper: g=(babaabbb,b⋄baab,aaabbb) f=(acb,c,ab)
696.ABBCCAB 2 HD0L: g=(a⋄aaabbbab,abbaabbab,bbbbaab), f=(abc,ac,b)
697.ABBCCAC 2 reverse of ABAACCB
698.ABBCCACA 2 reverse of ABABBCCA

699.ABBCCACB 2 HD0L: g=(abb,aaba,⋄babbaba), f=(abc,b,cba)
700.ABBCCACC 2 reverse of AABAACCB
701.ABBCCB 2 Upper: AABBA
702.ABC ∞
703.ABCA ∞
704.ABCAA ∞
705.ABCAAB ∞
706.ABCAABA 4 reverse of ABAACBA
707.ABCAABAA 3 reverse of AABAACBA
708.ABCAABAAB 2 Upper: AABAAB
709.ABCAABAAC 2 Upper: ABCACCB
710.ABCAABAB 3 reverse of ABABBCAB
711.ABCAABABA 2 Upper: AABABA
712.ABCAABABB 2 Upper: AABABB
713.ABCAABABC 2 Upper: ABCCACAB
714.ABCAABAC 2 HD0L: g=(babab,⋄bbba,aaaa),f=(acb,c,ab)
715.ABCAABB 3 reverse of AABBCAB
716.ABCAABBA 2 Upper: AABBA
717.ABCAABBC 2 reverse of ABBCCABC
718.ABCAABC 2 Upper: ABAAB
719.ABCAAC 3 reverse of ABBACB
720.ABCAACA 2 Upper: ABBAB
721.ABCAACB 2 HD0L: g=(a⋄aba,bbabbbaaa,baa,bbbbba), f=(abc,d,cba,b)
722.ABCAACC 2 Upper: ABBAA
723.ABCAB ∞
724.ABCABA ∞
725.ABCABAA 4 reverse of AABACBA
726.ABCABAAB 2 Upper: ABAAB
727.ABCABAAC 2 Upper: ABCACCB
728.ABCABAB 3 reverse of ABABCAB
729.ABCABABA 2 Upper: ABABA
730.ABCABABB 3 reverse of AABABCAB
731.ABCABABBA 2 Upper: ABABBA
732.ABCABABBC 2 Upper: ABCBCCA
733.ABCABABC 2 Upper: ABCACAB
734.ABCABAC 2 HD0L: g=(aab,abaa⋄bbb,abb,aaaa), f=(ab,cd,cb,ad)
735.ABCABB ∞
736.ABCABBA 3 reverse of ABBACBA
737.ABCABBAA 2 Upper: ABBAA
738.ABCABBAB 2 Upper: ABBAB

739.ABCABBAC 2 HD0L: g=(abbbaaa,⋄ab,abbbbb), f=(acb,c,ab)
740.ABCABBC 2 reverse of ABBCABC
741.ABCABC 2 Upper: ABAB HD0L: g=(aaabab,⋄bbba,baa), f=(acb,c,ab)
742.ABCAC ∞
743.ABCACA 3 reverse of ABABCA
744.ABCACAA 3 reverse of AABABCA
745.ABCACAAB 2 Upper: ABCBCCA
746.ABCACAAC 2 Upper: ABABBA
747.ABCACAB 2 HD0L: g=(baaaababab,⋄bbbaab,aaabb), f=(acb,c,ab)
748.ABCACAC 2 Upper: ABABA
749.ABCACB 3 Lower: Only 446 avoiding binary words with hole in tenth position HD0L: g=(aa,ccc⋄c,bb,cba), f=(adc,d,abc,b)
750.ABCACBA 2 HD0L: g=(a⋄bb,aaaa,babab,bbba), f=(ab,cd,cb,ad)
751.ABCACBB 2 reverse of AABCBAC
752.ABCACBC 2 reverse of ABACABC
753.ABCACC 4 reverse of AABACB
754.ABCACCA 2 Upper: ABAAB
755.ABCACCB 2 reverse of ABBCBAC
756.ABCB ∞
757.ABCBA ∞
758.ABCBAA ∞
759.ABCBAAB 3 reverse of ABBACAB
760.ABCBAABA 2 Upper: ABBAB
761.ABCBAABB 2 Upper: ABBAA
762. ABCBAABC 2 HD0L: g=(bbaa,abbb⋄ba,baaabab), f=(acb,c,ab)
763.ABCBAAC 2 reverse of ABBCACB
764.ABCBAB ∞
765.ABCBABA 3 reverse of ABABCBA
766.ABCBABAA 3 reverse of AABABCBA
767.ABCBABAAB 2 Upper: ABABBA
768.ABCBABAAC 2 Upper: ABACACCB
769.ABCBABAB 2 Upper: ABABA
770.ABCBABAC 2 HD0L: g=(baaabb,aba⋄bbaabbabaa), f=(abbba,baaab)
771.ABCBABB 4 reverse of AABACAB
772.ABCBABBA 2 Upper: ABAAB
773.ABCBABBC 2 reverse of ABBCBABC
774.ABCBABC 2 HD0L:g=(aabaab,bb⋄b,aaaa,abb), f=(adb,c,ab,d)
775.ABCBAC 3 reverse of ABCACB
776.ABCBACA 2 reverse of ABACBCA
777.ABCBACB 2 reverse of ABCABAC

778.ABCBACC 2 reverse of AABCACB
779.ABCBB ∞
780.ABCBBA ∞
781.ABCBBAA 3 reverse of AABBCBA
782.ABCBBAAB 2 Upper: AABBA
783.ABCBBAAC 2 reverse of ABBCCACB
784.ABCBBAB 4 reverse of ABAACAB
785.ABCBBABA 3 reverse of ABABBCBA
786.ABCBBABAA 2 Upper: AABABB
787.ABCBBABAB 2 Upper: AABABA
788.ABCBBABAC 2 Upper: ABAACACB
789.ABCBBABB 3 reverse of AABAACAB
790.ABCBBABBA 2 Upper: AABAAB
791.ABCBBABBC 2 reverse of ABBCBBABC
792.ABCBBABC 2 Upper: ABBA HD0L:g=(bbaaba,b⋄baaa,abb), f=(acb,c,ab)
793.ABCBBAC 2 HD0L: g=(abbab,baaa⋄aab,bbbbbaaab),f=(abc,ac,bb)
794.ABCBBC 2 Upper: ABAAB
795.ABCBC 3 reverse of ABABC
796.ABCBCA 3 Lower: Full word case Upper: ABAB
797.ABCBCAA 2 Upper: ABBAA
798.ABCBCAB 2 reverse of ABCACAB
799.ABCBCAC 2 reverse of ABACACB
800.ABCBCB 2 Upper: ABABA
801.ABCBCC 3 reverse of AABABC
802.ABCBCCA 2 reverse of ABBCBCA
803.ABCBCCB 2 Upper: ABABBA
804.ABCC ∞
805.ABCCA ∞
806.ABCCAA 3 reverse of AABBCA
807.ABCCAAB 2 reverse of ABBCCAB
808.ABCCAAC 2 Upper: AABBA
809.ABCCAB 3 Lower: Full word case Upper: ABBA
810.ABCCABA 2 reverse of ABACCBA
811.ABCCABB either 2 or 3 reverse of AABCCAB
812.ABCCABBA 2 Upper: ABBCAAC
813.ABCCABBC 2 reverse of ABBCAABC
814.ABCCABC 2 reverse of ABCAABC
815.ABCCAC 4 reverse of ABAACB
816.ABCCACA 3 reverse of ABABBCA
817.ABCCACAA 2 Upper: AABABB

818.ABCCACAB 2 reverse of ABCBCCAB
819.ABCCACAC 2 Upper: AABABA
820.ABCCACB 2 reverse of ABCBBAC
821.ABCCACC 3 reverse of AABAACB
822.ABCCACCA 2 Upper: AABAAB
823.ABCCACCB 2 reverse of ABBCBBAC
824.ABCCB 3 reverse of ABBAC
825.ABCCBA 3 Lower: Full word case Upper: ABBA
826.ABCCBAA 2 reverse of AABCCBA
827.ABCCBAB 2 reverse of ABACCAB
828.ABCCBAC 2 reverse of ABCAACB
829.ABCCBB 2 Upper: ABBAA
830.ABCCBC 2 reverse of ABAABC

References

[1] D.R. Bean, A. Ehrenfeucht, G. McNulty, Avoidable patterns in strings of symbols, Pacific Journal of Mathematics 85 (1979) 261–294.

[2] F. Blanchet-Sadri, Algorithmic Combinatorics on Partial Words, Chapman & Hall/CRC Press, Boca Raton, FL, 2008.

[3] F. Blanchet-Sadri, K. Black, A. Zemke, Unary pattern avoidance in partial words dense with holes, in: A.-H. Dediu, S. Inenaga, C. Martín-Vide (Eds.), LATA 2011, 5th International Conference on Language and Automata Theory and Applications, Tarragona, Spain, in: Lecture Notes in Computer Science, vol. 6638, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 155–166.

[4] F. Blanchet-Sadri, A. Lohr, S. Scott, Computing the partial word avoidability indices of ternary patterns, in: S. Arumugam, B. Smyth (Eds.), IWOCA 2012, 23rd International Workshop on Combinatorial Algorithms, Tamil Nadu, India, in: Lecture Notes in Computer Science, vol. 7643, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 206–218.

[5] F. Blanchet-Sadri, A. Lohr, S. Scott, Computing the partial word avoidability indices of binary patterns, Journal of Discrete Algorithms 23 (2013) 113–118 (in this issue), http://dx.doi.org/10.1016/j.jda.2013.06.007.

[6] F. Blanchet-Sadri, R. Merca ¸s, S. Simmons, E. Weissenstein, Avoidable binary patterns in partial words, Acta Informatica 48 (2011) 25–41.

[7] J. Cassaigne, Unavoidable binary patterns, Acta Informatica 30 (1993) 385–395.

[8] J. Cassaigne, Motifs évitables et régularités dans les mots, PhD thesis, Paris VI, 1994.

[9] R.J. Clark, The existence of a pattern which is 5-avoidable but 4-unavoidable, International Journal of Algebra and Computation 16 (2006) 351–367.

[10] M. Lothaire, Combinatorics on Words, Cambridge University Press, Cambridge, 1997.

[11] M. Lothaire, Algebraic Combinatorics on Words, Cambridge University Press, Cambridge, 2002.

[12] P. Ochem, A generator of morphisms for infinite words, RAIRO – Theoretical Informatics and Applications 40 (2006) 427–441.

[13] P. Ochem, Pattern avoidance and HDOL words, in: P. Ambrož, Š. Holub, Z. Masáková (Eds.), WORDS 2011, 8th International Conference on Words, Prague, Czech Republic, September 12–16, 2011, in: Electronic Proceedings of Theoretical Computer Science, vol. 63, 2011, p. 30.

[14] A.I. Zimin, Blocking sets of terms, Mathematics of the USSR – Sbornik 47 (1984) 353–364.