Non-repetitive strings over alphabet lists

Neerja Mhaskar[1] and Michael Soltys[⋆2]

[1] McMaster University
Dept. of Computing & Software
1280 Main Street West
Hamilton, Ontario L8S 4K1, CANADA
pophlin@mcmaster.ca
[2] California State University Channel Islands
Dept. of Computer Science
One University Drive
Camarillo, CA 93012, USA
michael.soltys@csuci.edu

Abstract. A word is non-repetitive if it does not contain a subword of the form vv. Given a list of alphabets $L = L_1, L_2, \ldots, L_n$, we investigate the question of generating non-repetitive words $w = w_1 w_2 \ldots w_n$, such that the symbol w_i is a letter in the alphabet L_i. This problem has been studied by several authors (e.g., [GKM10], [Sha09]), and it is a natural extension of the original problem posed and solved by A. Thue. While we do not solve the problem in its full generality, we show that such strings exist over many classes of lists. We also suggest techniques for tackling the problem, ranging from online algorithms, to combinatorics over 0-1 matrices, and to proof complexity. Finally, we show some properties of the extension of the problem to abelian squares.

Keywords: Thue words, non-repetitive, square-free, abelian square

1 Introduction

A *string* over a (finite) *alphabet* Σ is an ordered sequence of symbols from the alphabet: let $w = w_1 w_2 \ldots w_n$, where for each i, $w_i \in \Sigma$. In order to emphasize the array structure of w, we sometimes represent it as $w[1..n]$. We say that v is a *subword* of w if $v = w_i w_{i+1} \ldots w_j$, where $i \leq j$. If $i = j$, then v is a single symbol in w; if $i = 1$ and $j = n$, then $v = w$; if $i = 1$, then v is a *prefix* of w and if $j = n$, then v is a *suffix* of w. We can express that v is a subword more succinctly as follows: $v = w[i..j]$, and when the delimiters do not have to be expressed explicitly, we use the notation $v \leq w$. We say that v is a *subsequence* of w if $v = w_{i_1} w_{i_2} \ldots w_{i_k}$, for $i_1 < i_2 < \ldots < i_k$.

We now define the main concept in the paper, namely a string over an *alphabet list.* Let:

$$L = L_1, L_2, \ldots, L_n ,$$

⋆ Research supported in part by an NSERC Discovery Grant.

be an ordered list of (finite) alphabets. We say that w is a string over the list L if $w = w_1 w_2 \dots w_n$ where for all i, $w_i \in L_i$. Note that we impose no conditions on the L_i's: they may be equal, disjoint, or have elements in common. The only condition on w is that the i-th symbol of w must be selected from the i-th alphabet, i.e., $w_i \in L_i$. Let Σ_k denote a fixed generic alphabet of k symbols and let $\Sigma_L = L_1 \cup L_2 \cup \cdots \cup L_n$.

Given a list L of finite alphabets, we can define the set of strings w over L with a regular expression R_L: $R_L := L_1 \cdot L_2 \cdot \ldots \cdot L_n$.

Let $L^+ := L(R_L)$ be the language of all the strings over the list L. For example, if $L_0 = \{\{a, b, c\}, \{c, d, e\}, \{a, 1, 2\}\}$, then

$$R_{L_0} := \{a, b, c\} \cdot \{c, d, e\} \cdot \{a, 1, 2\},$$

and $ac1 \in L_0^+$, but $2ca \notin L_0^+$. Also, in this case $|L_0^+| = 3^3 = 27$. We should point out that $\{a, b, c\}$ is often written as $(a + b + c)$, but we use the curly brackets since it is reminiscent of indeterminate strings, which is yet another way of looking at strings over alphabet lists. See, for example, [Abr87] or [SW09] for a treatment of indeterminates.

We say that w has a *repetition* (or a *square*) if there exists a v such that $vv \leq w$. We say that w is *non-repetitive* (or *square-free*) if no such subword exists. An alphabet list L is *admissible* if L^+ contains a non-repetitive string. Let $\mathcal{L}$ represent a *class* of lists; the intention is for $\mathcal{L}$ to denote lists with a given property. For example, we are going to use $\mathcal{L}_{\Sigma_k}$ to denote the class of all lists $L = L_1, L_2, \dots L_n$, where for each $i \in [n] = \{1, 2, \dots, n\}$, $L_i = \Sigma_k$, and $\mathcal{L}_k$ will denote the class of all lists $L = L_1, L_2, \dots, L_n$, where for each $i \in [n]$, $|L_i| = k$, that is, those lists consisting of alphabets of size k. Note that $\mathcal{L}_{\Sigma_k} \subseteq \mathcal{L}_k$. We say that a class of lists $\mathcal{L}$ is admissible if *every* list $L \in \mathcal{L}$ is admissible. For ease of reference, we include a table summarizing the notation for classes with different properties in a table at the end of the paper.

Since any string of length at least 4 over $\Sigma_2 = \{0, 1\}$ contains a square, it follows that $\mathcal{L}_2$ is not admissible. On the other hand, [Thu06] showed using substitutions that $\mathcal{L}_{\Sigma_3}$ is admissible. Using a probabilistic algorithm, [GKM10] showed that $\mathcal{L}_4$ is admissible; the algorithm works as follows: in its i-th iteration, it selects randomly a symbol from L_i, and continues if the string created thus far is square-free, and otherwise deletes the suffix consisting of the right side of the square it just created, and restarts from the appropriate position.

Our paper is motivated by the following question, already posed in [GKM10]: is the class $\mathcal{L}_3$ admissible? That is, given any list $L = L_1, L_2, \dots, L_n$, where for all $i \in [n]$, $|L_i| = 3$, can we always find a non-repetitive string over such a list? We conjecture with [GKM10] that the answer to this question is affirmative, but we only show that certain (large) subclasses of $\mathcal{L}_3$ are admissible (Theorem 8). In Section 4 we propose different approaches for attacking this conjecture in its full generality.

2 Combinatorial results

Consider the alphabet $\Sigma_3 = \{1, 2, 3\}$, and the following *substitution scheme*, i.e., *morphism*, due to A. Thue, as presented in [GKM10]:

$$S = \begin{cases} 1 \mapsto 12312 \\ 2 \mapsto 131232 \\ 3 \mapsto 1323132 \end{cases} \tag{1}$$

Given a string $w \in \Sigma_3^*$, we let $S(w)$ denote w with every symbol replaced by its corresponding substitution: $S(w) = S(w_1 w_2 \dots w_n) = S(w_1)S(w_2)\dots S(w_n)$.

Lemma 1. *If $w \in \Sigma_3^*$ is a square-free string, then so is $S(w)$.*

Thue's substitution (1) is not the only one; for example, [Lee57] proposes a different substitution[3]. [Ber95, Theorem 3.2], which is a translation of Thue's work on repetitions in words, gives a characterization of the properties of such substitutions (called therein *iterated morphism*). It requires the morphism to be square free for any w of length 3 over Σ_3. Our proof does not require this assumption.

Corollary 2 (A. Thue). *$\mathcal{L}_{\Sigma_3}$ is admissible.*

We are interested in the question whether $\mathcal{L}_3$ is admissible, i.e., whether every list $L = L_1, L_2, \dots, L_n$, with $|L_i| = 3$, is admissible. Experimental data, with lists of length 20, seems to confirm it. Since we are not able to answer this question in its full generality, we examine different sub-classes of $\mathcal{L}_3$ for which it is true. The goal of this approach is to eventually show that $\mathcal{L}_3$ is admissible.

Recall that a System of Distinct Representatives (SDR) of a collection of sets $\{L_1, L_2, \dots, L_n\}$ is a selection of n distinct elements $\{a_1, a_2, \dots, a_n\}$, $a_i \in L_i$.

Claim 3. *If L has an SDR, then L is admissible.*

Proof. Simply let $w = a_1 a_2 \dots a_n$ be the string consisting of the distinct representatives; as all symbols are distinct, w is necessarily square-free. $\square$

It is a celebrated result of P. Hall ([Hal87]) that a necessary and sufficient condition for a collection of sets to have an SDR is that they have the *union property*: for any sub-collection $\{L_{i_1}, \dots, L_{i_k}\}$, $1 \leq k \leq n$, $|L_{i_1} \cup \dots \cup L_{i_k}| \geq k$.

Corollary 4. *If L has the union property, then L is admissible.*

Given a list L, we say that the mapping $\Phi : L \longrightarrow \Sigma_3$, $\Phi = \langle \phi_i \rangle$, is *consistent* if for all i, $\phi_i : L_i \longrightarrow \Sigma_3$ is a bijection, and for all $i \neq j$, if $a \in L_i \cap L_j$, then $\phi_i(a) = \phi_j(a)$. In other words, Φ maps all the alphabets to the single alphabet Σ_3, in such a way that the same symbol is always mapped to the same unique symbol in $\Sigma_3 = \{1, 2, 3\}$.

[3] Leech's substitutions are longer than Thue's, and they are defined as follows (see [Tom10]): $1 \mapsto 1232132312321$; $2 \mapsto 2313213123132$; $3 \mapsto 3121321231213$.

Lemma 5. *If L has a consistent mapping, then L is admissible.*

Proof. Suppose that L has a consistent mapping $\Phi = \langle \phi_i \rangle$. By Corollary 2 we pick a non-repetitive $w = w_1 w_2 \dots w_n$ of length n. Let

$$w' = \phi_1^{-1}(w_1)\phi_2^{-1}(w_2)\dots\phi_n^{-1}(w_n),$$

then w' is a string over L, and it is also non-repetitive. If it were the case that $vv \leq w'$, then the subword vv of w' under Φ would be a square in w, which is a contradiction. □

Let CMP $= \{\langle L \rangle : L$ has a consistent mapping$\}$ be the "Consistent Mapping Problem," i.e., the language of lists $L = L_1, L_2, \dots, L_n$ which have a consistent mapping. We show in Lemma 6 that this problem is **NP**-complete. It is clearly in **NP** as a given mapping can be verified efficiently for consistency.

Lemma 6. CMP *is* **NP***-hard.*

Proof. A graph $G = (V, E)$ is 3-colorable if there exists an assignment of three colors to its vertices such that no two vertices with the same color have an edge between them. The problem 3-color is **NP**-hard, and by [GJS76] it remains **NP**-hard even if the graph is restricted to be planar.

We show that CMP is **NP**-hard by reducing the 3-colorability of planar graphs to CMP. Given a planar graph $P = (V, E)$, we first find all its triangles, that is, all cliques of size 3. There are at most $\binom{n}{3} \approx O(n^3)$ such triangles, and note that two different triangles may have 0, 1, or 2 vertices in common. If the search yields no triangles in P, then by [Grö59] such a P is 3-colorable, and so we map P to a fixed list with a consistent mapping, say $L = L_1 = \{a, b, c\}$. (In fact, by [DKT11] it is known that triangle-free planar graphs can be colored in linear time.)

Otherwise, denote each triangle by its vertices, and let $T_1, T_2, \dots, T_k$ be the list of all the triangles, each $T_i = \{v_1^i, v_2^i, v_3^i\}$; note that triangles may overlap. We say that an edge $e = (v_1, v_2)$ is *inside* a triangle if both v_1, v_2 are in some T_i. For every edge $e = (v_1, v_2)$ *not* inside a triangle, let $E = \{e, v_1, v_2\}$. Let $E_1, E_2, \dots, E_\ell$ be all such triples, and the resulting list is:

$$L_P = T_1, T_2, \dots, T_k, E_1, E_2, \dots, E_\ell.$$

See example given in Figure 1.

We show that L_P has a consistent mapping if and only if P is 3-colorable.

Suppose that P is 3-colorable. Let the colors be labeled with $\Sigma_3 = \{1, 2, 3\}$; each vertex in P can be labeled with one of Σ_3 so that no edge has end-points labeled with the same color. This clearly induces a consistent mapping as each triangle $T_i = \{v_1^i, v_2^i, v_3^i\}$ gets 3 colors, and each $E = \{e, v_1, v_2\}$ gets two colors for v_1, v_2, and we give e the third remaining color.

Suppose, on the other hand, that L_P has a consistent mapping. This induces a 3-coloring in the obvious way: each vertex inside a triangle gets mapped to one of the three colors in Σ_3, and each vertex not in a triangle is either a singleton, in which case it can be colored arbitrarily, or the end-point of an edge not inside a triangle, in which case it gets labeled consistently with one of Σ_3. □

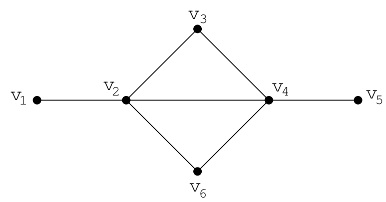

Fig. 1. In this case the list L_P is composed as follows: there are two triangles, $\{v_2, v_3, v_4\}, \{v_2, v_6, v_4\}$, and there are two edges not inside a triangle giving rise to $\{v_1, v_2, (v_1, v_2)\}, \{v_4, v_5, (v_4, v_5)\}$. Note that this planar graph is 3-colorable: $v_1 \mapsto 1$, $v_2 \mapsto 2$, $v_3 \mapsto 3$, $v_6 \mapsto 3$, $v_4 \mapsto 1$, and $v_5 \mapsto 2$. And the same assignment can also be interpreted as a consistent mapping of the list L_P.

We say that a collection of sets $\{L_1, L_2, \ldots, L_n\}$ is a *partition* if for all i, j, $L_i = L_j$ or $L_i \cap L_j = \emptyset$.

Corollary 7. *If L is a partition, then L is admissible.*

Proof. We show that when L is a partition, we can construct a consistent Φ, and so, by Lemma 5, L is admissible. For each i in $[n]$ in increasing order, if L_i is new i.e., there is no $j < i$, such that $L_i = L_j$, then let $\phi_i : L_i \longrightarrow \Sigma_3$ be any bijection. If, on the other hand, L_i is not new, there is a $j < i$, such that $L_i = L_j$, then let $\phi_i = \phi_j$. Clearly $\Phi = \langle \phi_i \rangle$ is a consistent mapping. □

Note that by Lemma 5, the existence of a consistent mapping guarantees the existence of a square-free string. The inverse relation does not hold: a list L may not have a consistent mapping, and still be admissible. For example, consider $L = \{\{a, b, c\}, \{a, b, e\}, \{c, e, f\}\}$. Then, in order to have consistency, we must have $\phi_1(a) = \phi_2(a)$ and $\phi_1(b) = \phi_2(b)$. In turn, by bijectivity, this implies that $\phi_1(c) = \phi_2(e)$. Again, by consistency:

$$\phi_3(c) = \phi_1(c) = \phi_2(e) = \phi_3(e),$$

and so $\phi_3(c) = \phi_3(e)$, which violates bijectivity. Hence L does not have a consistent mapping, but $w = abc \in L^+$, and w is square-free.

Let $\mathcal{L}_{\mathrm{SDR}}, \mathcal{L}_{\mathrm{Union}}, \mathcal{L}_{\mathrm{Consist}}$, and $\mathcal{L}_{\mathrm{Part}}$, be classes consisting of lists with: an SDR, the union property, a consistent mapping, and the partition property, respectively. Summarizing the results in the above lemmas we obtain the following theorem.

Theorem 8. $\mathcal{L}_{\mathrm{SDR}}, \mathcal{L}_{\mathrm{Union}}, \mathcal{L}_{\mathrm{Consist}}$, *and* $\mathcal{L}_{\mathrm{Part}}$ *are all admissible.*

A natural way to construct a non-repetitive string over L is as follows: pick any $w_1 \in L_1$, and for $i + 1$, assuming that $w = w_1 w_2 \ldots w_i$ is non-repetitive, pick an $a \in L_{i+1}$, and if wa is non-repetitive, then let $w_{i+1} = a$. If, on the other hand, wa has a square vv, then vv must be a suffix (as w is non-repetitive by assumption). Delete the right copy of v from w, and restart.

The above paragraph describes the gist of the algorithm for computing a non-repetitive string over $\mathcal{L}_4$, presented in [GKM10]. The correctness of the algorithm relies on a beautiful probabilistic argument that we present partially in the proof of Lemma 11. For the full version of this result the reader is directed to the source [GKM10]. On the other hand, the correctness of the algorithm in [GKM10] also relies on Lemma 9 shown below, which was assumed but not shown [GKM10, line 7 of Algorithm 1, on page 2].

Incidentally, suppose that there is an $L \in \mathcal{L}_4$ with the following property: there exists an $L_i = \{a, b, c, d\}$ such that if w is a non-repetitive string in L^+, then $w_i = a$. That is, all non-repetitive strings in L^+ must select a from L_i. Then $\mathcal{L}_3$ would be inadmissible, since we could construct an inadmissible $L' \in \mathcal{L}_3$ as follows: $L'_i = \{b, c, d\}$, and for $j \neq i$, L'_j any 3-element subset of L_j.

Lemma 9. *If w is non-repetitive, then for any symbol a, either $w' = wa$ is still non-repetitive, or w' has a unique square (consisting of a suffix of w').*

Proof. Suppose that $w' = wa$ has a square; denote this square $v_\ell v_r$, where $v_\ell = v_r$, and $v_\ell v_r$ is a suffix of w'. Suppose that there is another square $v'_\ell v'_r$. We examine the following cases:

1. If $|v'_r| \leq \lfloor \frac{|v_r|}{2} \rfloor$, then $v'_\ell v'_r$ is a suffix of v_r, and hence $v'_\ell v'_r$ is also a suffix of v_ℓ, and hence w has a square — contradiction.
2. If $\lfloor \frac{|v_r|}{2} \rfloor < |v'_r| < |v_r|$, then let x be the (unique) suffix of v'_ℓ that corresponds to a prefix of v_r. Note that the case $|v'_r| = |v_r|$ is superfluous, as it means that $v'_r = v_r$, and since $|v'_r| < |v_r|$, $|x| > 0$. Since x is a suffix of v'_ℓ, it also must be a suffix of v'_r, and so x is also a suffix of v_r, and hence a suffix of v_ℓ. Thus, we must have xx straddling $v_\ell v_r$, and thus we have a square in w — contradiction.
3. The case $|v_r| < |v'_r| < |v_\ell v_r|$ is symmetric to the previous case, with the roles of v_r, v'_r and v_ℓ, v'_ℓ reversed.
4. Finally, $|v'_r| \geq |v_\ell v_r|$ means that $v_\ell v_r$ is also a subword of v'_ℓ, giving us a repetition $v'_\ell \leq w$, and hence a contradiction.

Thus, the only possible case is $v_\ell = v'_\ell, v_r = v'_r$, and this means that w' must have a unique repetition, if it has one at all. □

An open question is how to de-randomize [GKM10, Algorithm 1]. The naïve way to de-randomize it is to employ an exhaustive search algorithm: given an L in $\mathcal{L}_4$, examine every $w \in L^+$ in lexicographic order until a non-repetitive is found, which by [GKM10, Theorem 1] must happen. In that sense, the correctness of the probabilistic algorithm implies the correctness of the deterministic exhaustive search algorithm. However, such an exhaustive search algorithm takes $4^{|L|}$ steps in the worst case; is it possible to de-randomize it to a deterministic polytime algorithm? Also, what is the expected running time of the probabilistic algorithm?

3 Abelian Squares

There are generalizations of the notion of a square in a string. For example, while a *square* in w is a subword $vv \leq w$, an *overlap* is a subword of the form $avava$, where a is a single symbol, and v is an arbitrary word (see [Sha09, pg. 37], and the excellent [Ram07]). The point is that the string $avava$ can be seen as two overlapping occurrences of the word ava. While there are no arbitrarily long square-free words over $\Sigma_2 = \{0, 1\}$, there are arbitrarily long overlap-free words over Σ_2 (see [Sha09, Theorem 2.5.1, pg. 38]).

An *abelian square* is a word of the form ww' where $|w| = |w'|$, and where w' is a permutation of w. That is, if $w = w_1 w_2 \ldots w_n$, then $w' = w_{\pi(1)} w_{\pi(2)} \ldots w_{\pi(n)}$, where $\pi : [n] \longrightarrow [n]$ is a bijection. A word w is abelian-square-free if there is no $vv' \leq w$ such that vv' is an abelian square. While there are arbitrarily long square-free words over Σ_3, the question was posed in [Sha09, Section 2.9, Problem 1(a), pg. 47] whether there are infinite abelian-square-free words (where aa is not counted as an abelian square, that is, abelian-square-of-size-at-least-2-free words). We show in Lemma 10 that there are no abelian-square-free words of size 8 or bigger; but allowing abelian squares of size 1 makes the problem more difficult. Here is a word of size 25, with no abelian-square-free but allowing abelian squares of size 1: $aaabaaacaaabbbaaacaa$.

Lemma 10. *If w is a word over Σ_3 such that $|w| \geq 8$, then w must have an abelian square.*

Proof. We show that if $w \in \Sigma_3^{\geq 8}$, i.e., w is a word over Σ_3 of size at least 8, then w necessarily has an abelian square.

Let $\tau : \Sigma_3 \longrightarrow \Sigma_3$ be a bijection, that is, τ is a permutation of $\{a, b, c\}$. (Note that this is not the same as the π above, which is a permutation of a string w.) It is easy to see that for each of the six possible τ's, w is an abelian square if and only if $\tau(w)$ is an abelian square. Therefore, if we show that for any w of the form $w = abx$, where $x \in \Sigma_3^*$, w has an abelian square, it will follow that every w has an abelian square. (If $w = aax, bbx, ccx$ then w has a square, which is also an abelian square, and for the six cases that arise from two distinct initial characters we apply a τ to reduce it to the $w = abx$ case.)

Consider Figure 2 which represents with a tree the prefixes of all the strings over Σ_3. Think of the labels on the nodes on any branch starting at the root (ε) as spelling out such a prefix. Note that all the branches starting with ab end in a $\times$-leaf, which denotes that adding any symbol in $\Sigma_3 = \{a, b, c\}$ would yield an abelian square. This proves the Lemma, as the other prefixes (starting with one of $\{ba, bc, ca, cb\}$) would also eventually yield an abelian square. $\square$

Adapting the method of [GKM10] we can also show that there are infinite abelian-square-free words over lists of size 4.

Lemma 11. *Let L be any list where for all i, $|L_i| = 4$. Then, there is an abelian-square-free word over L.*

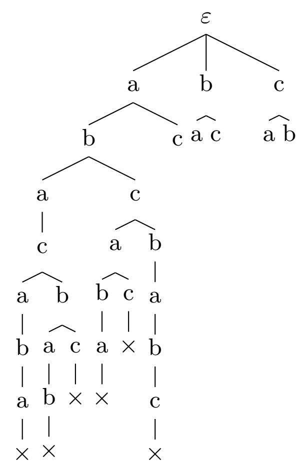

Fig. 2. No abelian squares of length greater than 8.

Proof. Fix an ordering inside each L_i, and let $r = r_1, r_2, \ldots, r_m$ be a sequence over $\{1, 2, 3, 4\}$. We use r to build an abelian-square-free word as follows: starting with $w = \varepsilon$, in the i-th step, add to the right end of w the symbol in position r_i in $L_{|w|+1}$. If the resulting w' is abelian-square-free, continue. Otherwise, there is an abelian square (which, unlike in the case of regular squares, does not have to be unique — see Lemma 9). Let vv' be the longest abelian square so that $w' = xvv'$. Delete v' and restart the process. Let (D, s) be a log of the procedure, where D is a sequence of integers keeping track of the differences in size of the w's from one step to the next; let s be the final string after the entire r has been processed. Following the same technique as in [GKM10], we show that given (D, s) there is a unique r corresponding to it. By assuming that the total number of s's are less than a given n_0, we get a contradiction by letting r be sufficiently large, and bounding the number of logs with Catalan numbers [Sta99]. $\square$

The authors have written a short Python program for checking abelian squares; you may find it on the second author's web page.

4 Future directions

4.1 Online algorithms and games

In the online version of the problem, L is presented piecemeal, one alphabet at a time, and we select the next symbol without knowing the future, and once selected, we cannot change it later. More precisely, the L_i's are presented one at a time, starting with L_1, and when L_i is presented, we must select $w_i \in L_i$, without knowing $L_{i+1}, L_{i+2}, \ldots$, and without being able to change the selections already committed to in $L_1, L_2, \ldots, L_{i-1}$.

We present the online problem in a game-theoretic context. Given a class of lists $\mathcal{L}$, and a positive integer n, the players take turns, starting with the adversary. In the i-th round, the *adversary* presents a set L_i, and the *player* selects a $w_i \in L_i$; the first i rounds can be represented as:

$$G = L_1, w_1, L_2, w_2, \ldots, L_i, w_i.$$

The condition imposed on the adversary is that $L = L_1, L_2, \ldots, L_n$ must be a member of $\mathcal{L}$.

The player has a *winning strategy for* $\mathcal{L}$, if $\forall L_1 \exists w_1 \forall L_2 \exists w_2 \ldots \forall L_n \exists w_n$, such that $L = L_1, L_2, \ldots, L_n \in \mathcal{L}$ and $w = w_1 w_2 \ldots w_n$ is square-free. For example, the player does not have a winning strategy for $\mathcal{L}_1$ and $\mathcal{L}_2$; see Figure 3. On the other hand, the player has a winning strategy for $\mathcal{L}_{\Sigma_3}$: simply pre-compute a square-free w, and select w_i from L_i. However, this is not a bona fide online problem, as all future L_i's are known beforehand. In a true online problem we expect the adversary to have the ability to "adjust" the selection of the L_i's based on the history of the game.

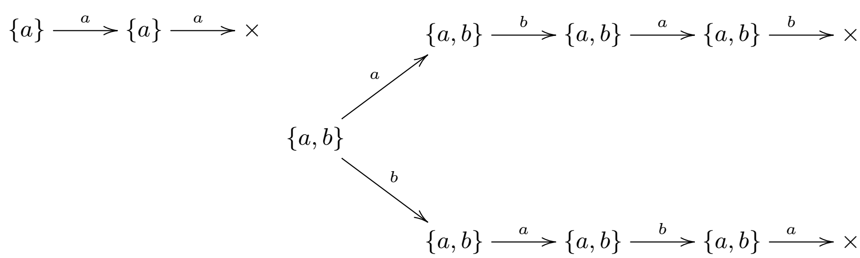

Fig. 3. Player loses if adversary is allowed subsets of size less than 3: the moves of the adversary are represented with subsets $\{a\}$ and $\{a, b\}$ and the moves of the player are represented with labeled arrows, where the label represents the selection from a subset.

We present another class of lists for which the player has a winning strategy. Let $\text{size}_L(i) = |L_1 \cup \ldots \cup L_i|$. We say that L has the *growth* property if for all $1 \leq i < n = |L|$, $\text{size}_L(i) < \text{size}_L(i+1)$. We denote the class of lists with the growth property as $\mathcal{L}_{\text{Grow}}$.

Lemma 12. *The player has a winning strategy for* $\mathcal{L}_{\text{Grow}}$.

Proof. In the i-th iteration, select w_i that has not been selected previously; the existence of such a w_i is guaranteed by the growth property. □

The growth property places a rather strong restriction on L, as it allows the construction of square-free strings where all the symbols are different, and hence they are trivially square-free. Note that the growth property implies the union property discussed in Corollary 4. To see this note that the growth property implies the existence of an SDR (discussed in Claim 3), in the stronger sense of every L_i containing an a_i such that for all $j \neq i$, $a_i \notin L_j$.

It would be interesting to study the relationship between admissible $\mathcal{L}$ in the original sense, and those $\mathcal{L}$ for which the player has a winning strategy in the online game sense. Clearly, if there exists a winning strategy for $\mathcal{L}$, then $\mathcal{L}$ is admissible; what about the converse?

4.2 Boolean matrices

Instead of considering alphabets, we consider sets of natural numbers, i.e., each $L_i \subseteq \mathbb{N}$, and $L = L_1, L_2, \ldots L_n$, and $\mathcal{L}$ is a class of lists as before. We say that $w \in L^+$ if $w = j_1, j_2, \ldots, j_n$, i.e., w is a sequence of numbers, such that for all $i \in [n]$, $j_i \in L_i$. The definition of repetitive (square) is analogous to the alphabet of symbols case.

Note that any $L = L_1, L_2, \ldots, L_n$ can be *normalized* to be $\hat{L}$, where each L_i is replaced with $\hat{L}_i \subseteq [3n]$. This can be accomplished by mapping all integers in $\cup L$, at most $3n$ many of them, in an order preserving way, to $[3n]$. Clearly, L is admissible iff $\hat{L}$ is admissible, and given a list L, it can be normalized in polynomial time. This allows us to restate the game theoretic approach given in the previous section with bounded quantification; this in turn places the problem in the polytime hierarchy, and hence in **PSPACE**. This is not surprising as many two-player zero-sum games are in this class (see [Pap94, Chapter 19]).

The integer restatement suggests an approach based on 0-1 matrices. Given a normalized list $L = \{L_1, L_2, \ldots, L_n\}$, we define the 0-1 $n \times 3n$ matrix A_L where row i of A_L is the incidence vector of L_i: $A_L(i,j) = 1 \iff j \in L_i$.

The attraction of this setting is that it may potentially allow us to use the machinery of combinatorial matrix theory to show that $\mathcal{L}_3$ is admissible.

It is easy to see that L is admissible iff there is a selection S that picks a single 1 in each row in such a way that there are no i consecutive rows equal to the next i consecutive rows. More precisely, L is admissible iff there does not exist i, j, such that $1 \leq i \leq j \leq \lfloor \frac{n}{2} \rfloor$, and such that the submatrix of A_L consisting of rows i through j is equal to the submatrix of A_L consisting of rows $j+1$ through $2(j+1)-i$.

Suppose that Σ_L is re-ordered bijectively by Γ, and let

$$L_\Gamma = \{\Gamma(L_1), \Gamma(L_2), \ldots, \Gamma(L_n)\}.$$

Then L is admissible iff L_Γ is admissible. Note that a bijective re-ordering of Σ_L is represented by a permutation of the columns of A_L. Thus, permuting the columns of A_L does not really change the problem; the same is not true of permuting the rows, which actually re-orders the list L, changing the constraints, and therefore changing the problem.

Consider the matrices $S = A_L A_L^t$ and $T = A_L^t A_L$. The element $[s_{ij}]$ record the number of elements common in the sets L_i and L_j, where $1 \leq i, j \leq n$. The diagonal elements $[s_{ii}]$ record the cardinality of the set L_i, which is 3. The element $[t_{ij}]$ record the number of times the numbers i and j, where $1 \leq i, j \leq 3n$, occur together in the sets of L. The diagonal elements of T display the total number of times each number in $[3n]$ appears in L. The properties of these matrices are studied to possibly use them in the construction of Φ(consistent mapping).

4.3 Proof complexity

By restating the generalized Thue problem in the language of 0-1 matrices, as we did in Section 4.2, we can more easily formalize the relevant concepts in the language of first order logic, and use its machinery to attack the problem.

We are going to adopt the logical theory $\mathbf{V}^0$ as presented in [CN10], whose language is $\mathcal{L}_A^2 = [0, 1, +, \cdot, ||; =_1, =_2, \leq, \in]$ (see [CN10, Definition IV.2.2, pg. 76]). Without going into all the details, this language allows the indexing of a 0-1 string X; on the other hand, a 0-1 matrix A_L can be represented as a string X_L with the definition: $X_L(3n(i-1)+j) = A_L(i,j)$. Hence, $\mathcal{L}_A^2$ is eminently suitable for expressing properties of strings.

Define the following auxiliary predicates:

- Let $\mathrm{Three}(X_L)$ be a predicate which states that the matrix A_L corresponding to X_L has exactly three 1s per row.
- Let $\mathrm{Sel}(Y_L, X_L)$ be a predicate which states that Y_L is a selection of X_L, in the sense that Y_L corresponds to the 0-1 matrix which selects a single 1 in each row of A_L.
- Let $\mathrm{SF}(Y_L)$ be a predicate which states that Y_L is square-free (i.e., non-repetitive).

Lemma 13. *All three predicates* $\mathrm{Three}, \mathrm{SF}, \mathrm{Sel}$ *are* Σ_0^B.

Our conjecture can be stated as a Σ_1^B formula over $\mathcal{L}_A^2$ as follows:

$$\alpha(X_L) := \exists Y_L \leq |X_L|(\mathrm{Three}(X_L) \wedge \mathrm{Sel}(Y_L, X_L) \wedge \mathrm{SF}(Y_L)).$$

Suppose we can prove that $\mathbf{V}^0 \vdash \alpha(X_L)$; then, we would be able to conclude that given any L, we can compute a non-repetitive string over L in $\mathbf{AC}^0$. Likewise, if $\mathbf{V}^1 \vdash \alpha(X_L)$, then we would be able to conclude that the non-repetitive string can be computed in polynomial time.

References

[Abr87] Karl R. Abrahamson. Generalized string matching. *SIAM J. Comput.*, 16(6):1039–1051, 1987.

[Ber95] J. Berstel. Axel Thue's papers on repetitions in words: a translation. Technical report, Université du Québec a Montréal, 1995.

[CN10] Stephen A. Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity.* Cambridge Univeristy Press, 2010.

[DKT11] Zdeněk Dvořák, Ken-Ichi Kawarabayashi, and Robin Thomas. Three-coloring triangle-free planar graphs in linear time. *ACM Trans. Algorithms*, 7(4):41:1–41:14, September 2011.

[GJS76] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science*, 1(3):237 – 267, 1976.

[GKM10] Jarosław Grytczuk, Jakub Kozik, and Pitor Micek. A new approach to nonrepetitive sequences. arXiv:1103.3809, December 2010.

[Grö59] Herbert Grötzsch. Ein dreifarbensatz für dreikreisfreie netze auf der kugel. 8:109–120, 1959.

[Hal87] P. Hall. On representatives of subsets. In Ira Gessel and Gian-Carlo Rota, editors, *Classic Papers in Combinatorics*, Modern Birkhäuser Classics, pages 58–62. Birkhäuser Boston, 1987.

[Lee57] John Leech. A problem on strings of beads. *Mathematical Gazette*, page 277, December 1957.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Ram07] Narad Rampersad. *Overlap-free words and generalizations*. PhD thesis, Waterloo University, 2007.

[Sha09] Jeffrey Shallit. *A second course in formal languages and automata theory*. Cambridge Univeristy Press, 2009.

[Sta99] Richard P. Stanley. Exercises on catalan and related numbers. *Enumerative Combinatorics*, 2, 1999.

[SW09] William F. Smyth and Shu Wang. An adaptive hybrid pattern-matching algorithm on indeterminate strings. *Int. J. Found. Comput. Sci.*, 20(6):985–1004, 2009.

[Thu06] Axel Thue. *Über unendliche zeichenreichen*. Skrifter: Matematisk-Naturvidenskapelig Klasse. Dybwad, 1906.

[Tom10] C. Robinson Tompkins. The morphisms with unstackable image words. *CoRR*, abs/1006.1273, 2010.

Summary of classes of lists

$\mathcal{L}$ denotes a class of lists		
$L = L_1, L_2, \ldots, L_n$ denotes a (finite) list of alphabets		
L_i denotes a finite alphabet		
Class name	**Description**	**Admissible**
$\mathcal{L}_{\Sigma_k}$	for all $i \in [n]$, $L_i = \Sigma_k$	for Σ_k, yes for $k \geq 3$; no for $k < 3$
$\mathcal{L}_k$	for all $i \in [n]$, $\lvert L_i\rvert = k$	yes for $k \geq 4$; no for $k \leq 2$; for $k = 3$?
$\mathcal{L}_{\text{SDR}}$	L has an SDR	yes
$\mathcal{L}_{\text{Union}}$	L has the union property	yes
$\mathcal{L}_{\text{Consist}}$	L has a consistent mapping	yes
$\mathcal{L}_{\text{Part}}$	L is a partition	yes
$\mathcal{L}_{\text{Grow}}$	for all i, $\lvert \cup_{j=1}^{i} L_j\rvert < \lvert \cup_{j=1}^{i+1} L_j\rvert$	yes, even for online games