# On shuffled-square-free words

Laurent Bulteau, Vincent Jugé, Stéphane Vialette

# On shuffled-square-free words

Laurent Bulteau[1], Vincent Jugé[1], and Stéphane Vialette[1]

Univ Gustave Eiffel, CNRS, LIGM, F-77454 Marne-la-Vallée, France
{laurent.bulteau,vincent.juge,stephane.vialette}@univ-eiffel.fr

**Abstract** A word $u$ is a shuffle of words $v$ and $w$, which we denote by $u \in v \sqcup\!\sqcup w$, if $u$ can be obtained by mixing the letters from $v$ and $w$ in a way that preserves the left-to-right ordering of the letters from $v$ and $w$. In case $u \in v \sqcup\!\sqcup v$ for some word $v$, the word $u$ is called a shuffled-square. A word $u$ is shuffled-square-free if it does not have a non-empty factor (*i.e*, non-empty sequence of adjacent letters) that is a shuffled-square. Our contribution in this context is two-fold. First, we prove that there exist arbitrarily long shuffled-square-free words in any alphabet with six letters or more, thereby improving on a previous result of Guégan and Ochem. Furthermore, we show that recognizing shuffled-square-free words on arbitrary alphabets is NP-complete.

## 1 Introduction

A *square* is a non-empty word of the form $uu$, and a word is square-free if none of its factors are squares [6]. For example, $aa$, $abab$ and $abbabbabbab$ are square words, whereas $bcabacb$ and $abacabcacb$ are square-free words. There exist several algorithms that can verify the square-freeness of a word of length $n$ in $O(n \log n)$ time [1,7,16]. Over a binary alphabet $\Sigma = \{a, b\}$, it is easy to check that the only square-free words are the empty word $\varepsilon$, $a$, $b$, $ab$, $ba$, $aba$ and $bab$. However, there exist arbitrarily long square-free words in any alphabet with three or more letters, as proved by Thue [22,21] (see also [2]). The numbers $s(n)$ of ternary square-free words of length $n = 1, 2, \dots$ form the sequence $1, 3, 6, 12, 18, 30, 42, 60, \dots$ (OEIS A006156), and each number $s(n)$ is subject to the inequalities $6 \cdot 1.032^n \leqslant s(n) \leqslant 6 \cdot 1.379^n$ [3]. Fraenkel and Simpson proved that the number of distinct squares in a word of length $n$ is bounded by $2n$ [10]. In [13], Ilie improved this bound to $2n - \theta(\log n)$. Let us also mention that extremal square-free words (*i.e.*, those square-free words that cannot be extended to a new square-free word by inserting a single letter in an arbitrary position) have been recently considered [11,17], and that a simple algorithm for generating square-free words from a random source is given in [20].

The *shuffle* product $\sqcup\!\sqcup$ applied to a pair of strings $u$ and $v$ returns the set of all possible interleavings of the symbols in $u$ and $v$ [15]. For instance, $ab \sqcup\!\sqcup ba = \{abba, abba, abab, baba, baab, baab\}$. A *shuffled-square* is a word $u$ that satisfies $u \in v \sqcup\!\sqcup v$ for some non-empty word $v$, and a *shuffled-square-free word* is a word that does not contain any factor that is a shuffled-square. For example, $abaabacc$ and $ababccaa$ are shuffled-squares, since $abaabacc \in abac \sqcup\!\sqcup abac$ and $ababccaa \in abca \sqcup\!\sqcup abca$, whereas $abacbabca$, $abcacbac$ and $abcbacbcab$ are not shuffled-square-free words. Note that $abbacc$ and $acbcba$ are neither shuffled-squares nor shuffled-square-free words.

Deciding whether a word is a shuffled-square has been proved to be NP-complete independently by Buss and Soltys [5] and Rizzi and Vialette [19], the former proving the result for alphabets as small as 9 letters. Recently, it has been proved that deciding whether a binary word is a shuffled-square is NP-complete [4]. Note that, given words $u$, $v$ and $w$, it is a standard textbook exercise to show that deciding if $u$ is a shuffle of $v$ and $w$ is polynomial-time solvable.

In Dagstuhl Seminar 14111 (*Combinatorics and Algorithmics of Strings*), Karhumäki and Rao asked about the existence of an infinite shuffled-square-free word over some finite alphabet $\Sigma$ [8]. It is clear that there exist only six shuffled-square-free words on the binary alphabet $\{a, b\}$: $a$, $ab$, $aba$, $b$, $ba$ and $bab$. Furthermore, a computer-aided search shows that there exist 237 shuffled-square-free words on the ternary alphabet $\{a, b, c\}$ (see Figure 1 for a complete list), so that $|\Sigma| \geqslant 4$ is a clear lower bound on the smallest possible size of $\Sigma$. As for the upper bound, using the Lovász local lemma, Currie proved there exist infinite shuffled-square-free words over a finite but large alphabet [9]. Soon afterward, Müller lowered the size of the alphabet to 10 [18].

The best result obtained so far is by Guégan and Ochem that used entropy compression to show that there exist at least $5.59^n$ shuffled-square-free words of length $n$ over a 7-letter alphabet. It is worth noting that this latter result is non-constructive.

Our contribution in this paper is two-fold. First, we prove that there exist arbitrarily long shuffled-square-free words in any alphabet with 6 letters or more, thereby improving on the previous result of Guégan and Ochem [12]. Furthermore, complementing [5], [19] and [4], we show that recognizing shuffled-square-free words is NP-complete on arbitrary alphabets.

## 2 Definitions

We follow standard terminology as used in [15,6]. Let $\Sigma$ be a non-empty, finite set, called *alphabet*. A *word* over $\Sigma$ is a sequence of letters from $\Sigma$. We denote the length of a word $u$ (*i.e.*, the number of letters it consists of) by $|u|$. The *empty word* $\varepsilon$ is the unique word of length 0. The set of all finite words over $\Sigma$ is denoted by $\Sigma^*$ and the set of all non-empty finite words over $\Sigma$ is denoted by $\Sigma^+$, so that $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$. For any integer $n \geqslant 0$, we denote by $\Sigma^n$ the set of all finite words over $\Sigma$ of length $n$. For any word $u \in \Sigma^n$ and any integer $i$ such that $1 \leqslant i \leqslant n$, we denote by $u_i$ the $i$-th letter of $u$.

For two words $u$ and $v$, we say that $u$ is a *prefix* (resp., a *suffix* or a *factor*) of $u$ if there exists a word $x$ (or if there exist words $x$ and $y$) such that $v = ux$ (resp., $v = xu$ or $v = xuy$). A *square* is a word $u$ of the form $u = vv$, where $v$ is a non-empty word. A *square-free word* is a word that does not contain any factor that is a square.

The *shuffle operator* on words, denoted by $\sqcup\!\sqcup$, is the set of words defined recursively as follows:

$$\begin{aligned} \forall u \in \Sigma^*, \qquad & u \sqcup\!\sqcup \varepsilon = \varepsilon \sqcup\!\sqcup u = \{u\} \\ \forall u, v \in \Sigma^*,\ \forall a, b \in \Sigma, \qquad & ua \sqcup\!\sqcup vb = (u \sqcup\!\sqcup vb)a \cup (ua \sqcup\!\sqcup v)b. \end{aligned}$$

A *shuffled-square* is a word $u$ that satisfies $u \in v \sqcup\!\sqcup v$ for some non-empty word $v$. A *shuffled-square-free word* is a word that does not contain any factor that is a shuffled-square.

## 3 Shuffled-square-free words on small alphabets

This section is devoted to studying the existence of arbitrarily (or even infinitely) long shuffled-square-free words on a small alphabet. Guégan and Ochem have shown that there exist infinite shuffled-square-free words over an alphabet of size 7 [12]. We improve this bound to 6.

**Theorem 1.** *For all integers $n \geqslant 0$, there exist at least $4.56^n$ shuffled-square-free words of length $n$ over a 6-letter alphabet. In particular, there exist infinite shuffled-square-free words over an alphabet of size 6.*

Our proof is not constructive, but relies on counting arguments. Below, we set $\sigma = 6$, and we denote by $\Sigma$ an alphabet of size $\sigma$. Then, let $A$ be the set of minimal (for the subfactor relation) shuffled-squares in $\Sigma^*$, and let $L$ be the set of words without factors in $A$.

Below, for every set $X$ of finite words, we denote by $X_n$ the set $X \cap \Sigma^n$ and by $x_n$ the cardinality of $X_n$. For every word $\lambda$, we also denote by $X^{\circ\lambda}$ and by $X^{\lambda\circ}$ the sets $X \cap (\Sigma^* \cdot \lambda)$ and $X \cap (\lambda \cdot \Sigma^*)$, respectively. We will focus on proving that $\ell_n \geqslant \rho^n$ for all $n \geqslant 0$, where $\rho \approx 4.56594$ is the largest real root of the polynomial

$$P(X) = X^5 - 5X^4 + 18X^3 - 210X^2 + 626X - 5.$$

It will follow that the prefix-closed set $L$ is infinite, and König's lemma [14] will then show that there exists an infinite word whose finite prefixes all belong to $L$, thereby being shuffled-square-free.

Consider a word $w \cdot \theta$ in the set $L_n \cdot \Sigma$, i.e., a $(n+1)$-letter-long word whose $n$-letter-long prefix is shuffled-square-free. If $w \cdot \theta$ has a factor $f$ that is a shuffled-square, such a factor must be a suffix of $w \cdot \theta$. Then, let $f$ be the smallest such factor:

| | | | | | |
|---|---|---|---|---|---|
| $a$ | | $b$ | | $c$ | |
| $ab$ | $ac$ | $ba$ | $bc$ | $ca$ | $cb$ |
| $aba$ | $aca$ | $bab$ | $bca$ | $cab$ | $cba$ |
| $abac$ | $acab$ | $babc$ | $bcab$ | $caba$ | $cbab$ |
| $abaca$ | $acaba$ | $babca$ | $bcaba$ | $cabac$ | $cbabc$ |
| $abacab$ | $acabac$ | $babcab$ | $bcabac$ | $cabaca$ | $cbabca$ |
| $abacaba$ | $acabaca$ | $babcaba$ | $bcabaca$ | $cabacab$ | $cbabcab$ |
| $abacabc$ | $acabacb$ | $babcabac$ | $bcabacab$ | $cabacabc$ | $cbabcaba$ |
| $abacb$ | $acabc$ | $babcabacb$ | $bcabacabc$ | $cabacb$ | $cbabcabac$ |
| $abacba$ | $acabca$ | $babcb$ | $bcabacb$ | $cabacba$ | $cbabcabacb$ |
| $abacbab$ | $acabcac$ | $babcba$ | $bcabacba$ | $cabacbab$ | $cbabcb$ |
| $abacbabc$ | $acabcacb$ | $babcbab$ | $bcabacbab$ | $cabacbabc$ | $cbabcba$ |
| $abacbabca$ | $acabcacba$ | $babcbac$ | $bcabacbabc$ | $cabacbabca$ | $cbabcbac$ |
| $abc$ | $acb$ | $bac$ | $bcabacbabca$ | $cabc$ | $cbac$ |
| $abca$ | $acba$ | $baca$ | $bcabc$ | $cabca$ | $cbaca$ |
| $abcab$ | $acbab$ | $bacab$ | $bcabcb$ | $cabcac$ | $cbacab$ |
| $abcaba$ | $acbabc$ | $bacaba$ | $bcabcba$ | $cabcacb$ | $cbacaba$ |
| $abcabac$ | $acbabca$ | $bacabac$ | $bcabcbac$ | $cabcacba$ | $cbacabac$ |
| $abcabacb$ | $acbabcab$ | $bacabacb$ | $bcabcbacb$ | $cabcacbac$ | $cbacabacb$ |
| $abcabacba$ | $acbabcaba$ | $bacabc$ | $bcac$ | $cabcb$ | $cbacabc$ |
| $abcac$ | $acbabcabac$ | $bacabca$ | $bcacb$ | $cabcba$ | $cbacabca$ |
| $abcacb$ | $acbabcabacb$ | $bacabcac$ | $bcacba$ | $cabcbab$ | $cbacabcac$ |
| $abcacba$ | $acbabcb$ | $bacabcacb$ | $bcacbac$ | $cabcbabc$ | $cbacabcacb$ |
| $abcacbac$ | $acbabcba$ | $bacabcacba$ | $bcacbaca$ | $cabcbabca$ | $cbacabcacba$ |
| $abcacbaca$ | $acbabcbac$ | $bacb$ | $bcacbacab$ | $cabcbac$ | $cbacb$ |
| $abcacbacab$ | $acbac$ | $bacba$ | $bcacbacabc$ | $cabcbacb$ | $cbacbc$ |
| $abcacbacabc$ | $acbaca$ | $bacbab$ | $bcacbc$ | $cabcbacbc$ | $cbacbca$ |
| $abcacbc$ | $acbacab$ | $bacbabc$ | $bcacbca$ | $cabcbacbca$ | $cbacbcab$ |
| $abcacbca$ | $acbacabc$ | $bacbabca$ | $bcacbcab$ | $cabcbacbcab$ | $cbacbcabc$ |
| $abcacbcab$ | $acbacabca$ | $bacbabcab$ | $bcb$ | $cac$ | $cbc$ |
| $abcb$ | $acbc$ | $bacbc$ | $bcba$ | $cacb$ | $cbca$ |
| $abcba$ | $acbca$ | $bacbca$ | $bcbab$ | $cacba$ | $cbcab$ |
| $abcbab$ | $acbcab$ | $bacbcab$ | $bcbabc$ | $cacbac$ | $cbcabc$ |
| $abcbabc$ | $acbcabc$ | $bacbcabc$ | $bcbabca$ | $cacbaca$ | $cbcabcb$ |
| $abcbabca$ | $acbcabcb$ | $bacbcabcb$ | $bcbabcb$ | $cacbacab$ | $cbcabcba$ |
| $abcbac$ | $acbcabcba$ | $bacbcabcba$ | $bcbac$ | $cacbacabc$ | $cbcabcbac$ |
| $abcbacb$ | $acbcabcbac$ | $bacbcabcbac$ | $bcbacb$ | $cacbc$ | $cbcac$ |
| $abcbacbc$ | $acbcac$ | $bacbcac$ | $bcbacbc$ | $cacbca$ | $cbcacb$ |
| $abcbacbca$ | $acbcacb$ | $bacbcacb$ | $bcbacbca$ | $cacbcab$ | $cbcacba$ |
| $abcbacbcab$ | $acbcacba$ | $bacbcacba$ | $bcbacbcab$ | $cacbcac$ | $cbcacbc$ |

**Figure 1.** The 237 shuffled-square-free words over the ternary alphabet $\{a, b, c\}$.

– if the word $w \cdot \theta$ is shuffled-square-free, it belongs to $L_{n+1}$;
– if not, and if $f$ has length 2, then $f = \theta^2$, and $w$ belongs to $L_n^{\circ\theta}$;
– if not, and if $f$ has length $2k$ for some integer $k \geqslant 2$, let $\lambda$ the 2-letter-long prefix of $f$; the words $w \cdot \sigma \cdot f^{-1} \cdot \lambda$ and $f$ belong to $L_{n+3-2k}^{\circ\lambda}$ and to $A_{2k}^{\lambda\circ}$, respectively.

From this disjunction of cases, we deduce the following inclusion relation over sets of words of length $n+1$:

$$L_n \cdot \Sigma \subseteq L_{n+1} \cup \bigcup_{\theta\in\Sigma} (L_n^{\circ\theta} \cdot \theta) \cup \bigcup_{\lambda\in\Sigma^2} \bigcup_{k=2}^{(n+1)/2} L_{n+3-2k}^{\circ\lambda} \cdot \lambda^{-1} \cdot A_{2k}^{\lambda\circ}.$$

This inclusion relation translates immediately into the inequality

$$\ell_n\sigma \leqslant \ell_{n+1} + \sum_{\theta\in\Sigma} \ell_n^{\circ\theta} + \sum_{\lambda\in\Sigma^2} \sum_{k=2}^{(n+1)/2} \ell_{n+3-2k}^{\circ\lambda} a_{2k}^{\lambda\circ}.$$

The quantity $\ell_n^{\circ\theta}$ does not depend on $\theta$, and the sets $L_n^{\circ\theta}$ form a partition of $L_n$. It follows that $\sigma\ell_n^{\circ\theta} = \ell_n$ for all $\theta \in \Sigma$. Then, when $k \geqslant 2$ and $\lambda$ is a square word of length 2, the sets $L_n^{\circ\lambda}$ and $A_{2k}^{\lambda\circ}$ are empty, *i.e.*, $\ell_{n+3-2k}^{\circ\lambda} = a_{2k}^{\lambda\circ} = 0$. Finally, when $k \geqslant 2$ and $\lambda$ is a non-square word of length 2, the quantities $\ell_n^{\circ\lambda}$ and $a_{2k}^{\lambda\circ}$ do not depend on $\lambda$, and the sets $L_n^{\circ\lambda}$ and $A_{2k}^{\lambda\circ}$ form a partition of $L_n$ and of $A_{2k}$, respectively. Hence, $\sigma(\sigma-1)\ell_n^{\circ\lambda} = \ell_n$ and $\sigma(\sigma-1)a_n^{\lambda\circ} = a_n$ for all non-square words $\lambda \in \Sigma^2$. It follows that

$$\ell_n\sigma \leqslant \ell_{n+1} + \ell_n + \frac{1}{\sigma(\sigma-1)} \sum_{k=2}^{(n+1)/2} \ell_{n+3-2k} a_{2k}.$$

Unfortunately, finding a tractable expression of $a_{2k}$ for all $k \geqslant 2$ seems out of reach. Therefore, we just focus on computing simple upper bounds $b_{2k}$ on the size of the set $A_{2k}$. It will then follow that

$$\ell_n\sigma \leqslant \ell_{n+1} + \ell_n + \frac{1}{\sigma(\sigma-1)} \sum_{k=2}^{(n+1)/2} \ell_{n+3-2k} b_{2k}. \tag{1}$$

Consider now some word $w \in A_{2k}$, with $k \geqslant 2$. The set $\{1, 2, \ldots, 2k\}$ can be partitioned into two subsets $U = \{u_1, u_2, \ldots, u_k\}$ and $V = \{v_1, v_2, \ldots, v_k\}$, where $u_1 < u_2 < \ldots < u_k$ and $v_1 < v_2 < \ldots < v_k$, such that $w_{u_1} w_{u_2} \cdots w_{u_k} = w_{v_1} w_{v_2} \cdots w_{v_k}$. Without loss of generality, we assume that $u_1 = 1$.

We associate $w$ with a path $\mathbf{D}$ that goes from the point of coordinates $(0,0)$ to the point of coordinates $(2k, 0)$. This path uses $2k$ unit steps that go one unit to the east and one unit either to the north or to the south, as follows: the $\ell$-th step of that path is a north-east step when $\ell \in U$, and a south-east step when $\ell \in V$. One checks easily that the $\ell$-th step arrives at the point of coordinates $(\ell, m)$, where $m \stackrel{\text{def}}{=\!=} \#\{i\colon u_i \leqslant \ell\} - \#\{i\colon v_i \leqslant \ell\}$.

The first step goes from the point $(0,0)$ to the point $(1,1)$. Furthermore, if the path reaches the line $y = 0$ at some point $(\ell, 0)$, the word $w_1 w_2 \cdots w_\ell$ is a shuffled-square, and the minimality of $w$ proves that $\ell = 2k$. Consequently, our path keeps staying strictly above the line $y = 0$ between its first and last steps: this is a Dyck path, and even after deleting these first and last steps, we still get a Dyck path (of length $2k-2$), which we denote by $\mathbf{D}'$.

*Example 1.* If $\Sigma$ contains the letters 1, 2 and 3, and if´ $k = 4$, the word $w = 12131231$ is the shuffled-square of the word $w_1w_2w_4w_5 = w_3w_6w_7w_8 = 1231$. Thus, we associate $w$ with the Dyck path shown in Figure 2, which never goes below the line $y = 1$ except at its endpoints.

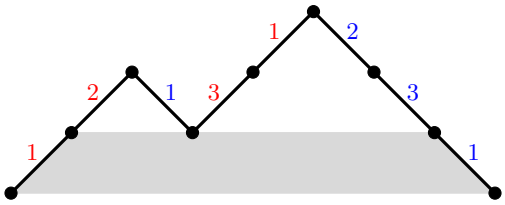



**Figure 2.** Dyck path associated with a factorization of a minimal shuffled-square.

Moreover, once the path $\mathbf{D}$ is chosen, recovering the word $w$ itself amounts to choosing the letters $w_{u_1}, w_{u_2}, \dots, w_{u_k}$, in that order. There are $\sigma$ ways to choose $w_{u_1}$. Then, once the letters $w_{u_1}, \dots, w_{u_{\ell-1}}$ have been chosen, the letter $w_{u_\ell - 1}$ is fixed, and therefore there are at most $\sigma - 1$ ways to choose $w_{u_\ell}$.

In case $u_\ell \geqslant 3$ and $u_\ell + 1 \in V$, we can even go further. Indeed, when choosing the value of $w_{u_\ell}$, the values of $w_{u_\ell - 2}$, $w_{u_\ell - 1}$ and $w_{u_\ell + 1}$ have already been chosen. Then,

- if $w_{u_\ell - 1} \neq w_{u_\ell + 1}$, the letter $w_{u_\ell}$ must be chosen among the $\sigma - 2$ elements of $\Sigma \setminus \{w_{u_\ell - 1}, w_{u_\ell + 1}\}$;
- if $w_{u_\ell - 1} = w_{u_\ell + 1}$, and since the factor $w_{u_\ell - 2} w_{u_\ell - 1} w_{u_\ell} w_{u_\ell + 1}$ is not allowed to be a shuffled-square, the letter $w_{u_\ell}$ must be chosen among the $\sigma - 2$ elements of $\Sigma \setminus \{w_{u_\ell - 1}, w_{u_\ell - 2}\}$.

Hence, in each of these two cases, there are at most $\sigma - 2$ ways to choose $w_{u_\ell}$.

*Example 2.* We consider the Dyck path obtained in Example 1, and look for letters $a$, $b$, $c$, $d$ that would result in an element of $A_8$, as illustrated in Figure 3.

There are $\sigma$ choices for the letter $a$. Then, the constraints $b \neq a$ and $c \neq a$ leave us with $\sigma - 1$ choices for each of the letters $b$ and $c$. Finally, if $c = b$, we know that $d \notin \{a, c\}$, because the word $acdb$ is shuffled-square-free; if $c \neq b$, we also know that $d \notin \{c, b\}$. This leaves us with $\sigma - 2$ choices for the letter $d$.

Note that, in this case, we omitted some constraints, e.g., we also need to have $b \neq c$.

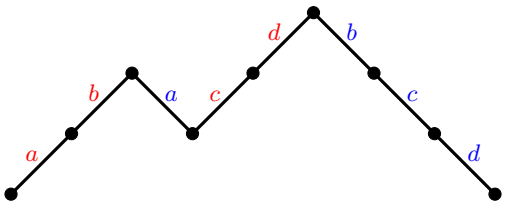
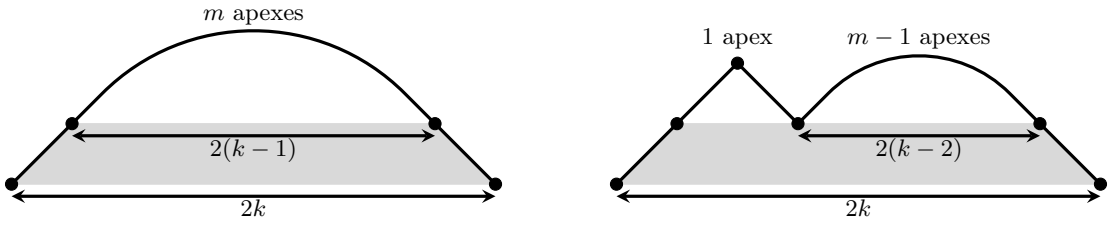



**Figure 3.** Choosing labels on increasing steps of a Dyck path.

This provides us with the following estimation. Let us denote by $D_{k,m}$ the number of Dyck paths of length $2k$ such that there are exactly $m$ north-east steps that are immediately followed by a south-east step: we say that these steps are the *apexes* of the path. The paths $\mathbf{D}$ and $\mathbf{D}'$ have as many apexes as each other, and choosing $\mathbf{D}'$ amounts to choosing $\mathbf{D}$.

Consequently, there are $D_{k-1,m}$ ways to choose $\mathbf{D}$ so that it has $m$ apexes. As illustrated in Figure 4 (right), in the first $D_{k-2,m-1}$ cases, the first apex of $\mathbf{D}$ is its second step. In the remaining $D_{k-1,m} - D_{k-2,m-1}$ cases, the first apex is its $\ell$-th step for some integer $\ell \geqslant 3$.

- In the first $D_{k-2,m-1}$ cases, we have $\sigma$ ways to choose the letter $w_1$, (at most) $\sigma - 2$ ways to choose each of the $m - 1$ letters $w_\ell$ such that $\ell \geqslant 3$ and the $\ell$-th step of $\mathbf{D}$ is an apex, and (at most) $\sigma - 1$ ways to choose each of the remaining $k - m$ letters $w_\ell$ such that $\ell = 2$




$D_{k-1,m}$ ways to choose $\mathbf{D}$ with $m$ apexes

$D_{k-2,m-1}$ ways to choose $\mathbf{D}$ with $m$ apexes, one of which is the second step

**Figure 4.** Counting ways of choosing Dyck paths with $2k$ steps and $m$ apexes.

or the $\ell$-th and $(\ell+1)$-th steps of $\mathbf{D}$ are north-east steps. Consequently, there are at most $\sigma(\sigma-2)^{m-1}(\sigma-1)^{k-m}$ ways to choose the letters $w_{u_1}, w_{u_2}, \ldots, w_{u_k}$ once the path $\mathbf{D}$ is given.

– In the remaining $D_{k-1,m} - D_{k-2,m-1}$ cases, the first apex of $\mathbf{D}$ is not its second step. In each of these cases, we have $\sigma$ ways to choose the letter $w_1$, $\sigma-2$ ways to choose each of the $m$ letters $w_\ell$ such that the $\ell$-th step is an apex, and $\sigma-1$ ways to choose each of the remaining $k-m-1$ letters such that the $\ell$-th and $(\ell+1)$-th steps are north-east steps. Consequently, there are at most $\sigma(\sigma-2)^{m}(\sigma-1)^{k-m-1}$ ways to choose the letters $w_{u_1}, w_{u_2}, \ldots, w_{u_k}$ once the path $\mathbf{D}$ is given.

Since $D_{k,0} = 0$ for all $k \geqslant 1$, it follows that

$$\begin{aligned} b_{2k} &= \sum_{m\geqslant 1} D_{k-2,m-1}\sigma(\sigma-2)^{m-1}(\sigma-1)^{k-m} + \\ &\quad \sum_{m\geqslant 1} \left(D_{k-1,m} - D_{k-2,m-1}\right)\sigma(\sigma-2)^{m}(\sigma-1)^{k-m-1} \\ &= \sum_{m\geqslant 1} D_{k-2,m-1}\sigma(\sigma-2)^{m-1}(\sigma-1)^{k-m-1} + \\ &\quad \sum_{m\geqslant 1} D_{k-1,m}\sigma(\sigma-2)^{m}(\sigma-1)^{k-m-1} \\ &= \sum_{m\geqslant 0} D_{k-2,m}\sigma(\sigma-2)^{m}(\sigma-1)^{k-m-2} + \\ &\quad \sum_{m\geqslant 0} D_{k-1,m}\sigma(\sigma-2)^{m}(\sigma-1)^{k-m-1} \end{aligned} \tag{2}$$

when $k \geqslant 2$.

Finally, consider the power series

$$L(x) = \sum_{n\geqslant 0} \ell_n x^n,\ B(x) = \sum_{k\geqslant 2} b_{2k} x^{2k} \text{ and } D(x,y) = \sum_{k,m\geqslant 0} D_{k,m} x^k y^m,$$

whose coefficients are non-negative integers. The series $D(x,1)$ counts Dyck paths without any concern for the number of their apexes, and thus it coincides with the *Catalan series*

$$D(x,1) = \sum_{n\geqslant 0} \frac{x^n}{n+1}\binom{2n}{n} = \frac{1-\sqrt{1-4x}}{2x},$$

whose radius of convergence is $1/4$. Thus, for every value $\hat{y}$ such that $|\hat{y}| \leqslant 1$, we already know that the series $D(x,\hat{y})$ in the variable $x$ has radius of convergence at least $1/4$.

Dyck paths are counted by the bivariate series $D(x,y)$ and can be decomposed recursively as follows. A Dyck path of length $2k$ with $m$ apexes is either:

– empty, in which case $k = m = 0$;
– made of a north-east step, then a south-east step, and finally a Dyck path of length $2(k-1)$ with $m-1$ apexes;
– made of a north-east step, then a Dyck path of length $2a$ with $b$ apexes, for some integers $a \geqslant 1$ and $b \geqslant 1$, then a south-east step, and finally another Dyck path of length $2(k-1-a)$ with $m-b$ apexes.

It follows that $D_{k,m} = \mathbf{1}_{k=m=0}$ when $k \leqslant 0$ or $m \leqslant 0$, and that

$$D_{k,m} = D_{k-1,m-1} + \sum_{a \geqslant 1} \sum_{b \geqslant 1} D_{a,b} D_{k-1-a,m-b} \tag{3}$$

when $k \geqslant 1$ and $m \geqslant 1$.

Alternatively, and denoting by $\mathbb{D}$ the class of Dyck paths and by $\mathbf{1}$ the (empty) Dyck path of length 0, we can write the above recursive decomposition as:

$$\mathbb{D} = \mathbf{1} \cup \nearrow \cdot \searrow \cdot \mathbb{D} \cup \nearrow \cdot (\mathbb{D} \setminus \{\mathbf{1}\}) \cdot \searrow \cdot \mathbb{D}.$$

This relation immediately translates into the equality

$$D(x,y) = 1 + xyD(x,y) + xD(x,y)(D(x,y)-1) \tag{4}$$

over generating functions. This equality can also be viewed as a consequence of (3).

The two solutions of (4) are

$$D(x,y) = \frac{1+x-xy \pm \sqrt{(1+x-xy)^2 - 4x}}{2x}.$$

The solution

$$\frac{1+x-xy+\sqrt{(1+x-xy)^2-4x}}{2x}$$

is not continuous around 0, and since we know that $D$ is continuous, it follows that

$$D(x,y) = \frac{1+x-xy-\sqrt{(1+x-xy)^2-4x}}{2x}, \tag{5}$$

this equality being valid on the domain $\mathrm{Dom} \stackrel{\text{def}}{=\!=} \{(x,y)\colon 0 \leqslant y \text{ and } 0 \leqslant x \leqslant 1/(1+\sqrt{y})^2\}$, on which $D$ is absolutely convergent.

We then prove that the series $B(x)$ coincides with the power series

$$E(x) \stackrel{\text{def}}{=\!=} \sigma x^2 (1+x^2) D\left((\sigma-1)x^2, \frac{\sigma-2}{\sigma-1}\right) - \sigma x^2,$$

which has radius of convergence

$$r = \frac{1}{\sqrt{\sigma-2}+\sqrt{\sigma-1}}.$$

Indeed, $E(x)$ is clearly even, and since $D_{0,m} = \mathbf{1}_{m=0}$ for all $m \geqslant 0$, its coefficients $e_{2k}$ satisfy the relation

$$\begin{aligned}\sum_{k\geqslant 0} e_{2k}x^{2k} = E(x) &= \sigma x^2\left(-1 + D\left((\sigma-1)x^2, \frac{\sigma-2}{\sigma-1}\right)\right) + \\ &\quad \sigma x^4 D\left((\sigma-1)x^2, \frac{\sigma-2}{\sigma-1}\right) \\ &= \sum_{k\geqslant 1}\sum_{m\geqslant 0} \sigma(\sigma-1)^{k-m}(\sigma-2)^m D_{k,m}x^{2k+2} + \\ &\quad \sum_{k\geqslant 0}\sum_{m\geqslant 0} \sigma(\sigma-1)^{k-m}(\sigma-2)^m D_{k,m}x^{2k+4} \\ &= \sum_{k\geqslant 2}\sum_{m\geqslant 0} \sigma(\sigma-1)^{k-m-1}(\sigma-2)^m D_{k-1,m}x^{2k} + \\ &\quad \sum_{k\geqslant 2}\sum_{m\geqslant 0} \sigma(\sigma-1)^{k-m-2}(\sigma-2)^m D_{k-2,m}x^{2k} \\ &= \sum_{k\geqslant 2} b_{2k}x^{2k} = B(x). \end{aligned} \tag{6}$$

In particular, both series $E(x)$ and $B(x)$ are absolutely convergent on the open interval $(0, r)$.

Finally, let $s$ be the minimum of the radii of convergence of the series $B(x)$ and $L(x)$. For each integer $n \geqslant 0$ and for each real number $x \in (0, s)$, we rewrite the inequality (1) as:

$$\sigma(\sigma-1)\left((\sigma-1)\ell_n - \ell_{n+1}\right)x^{n+3} \leqslant \sum_{k=2}^{(n+1)/2} \ell_{n+3-2k}b_{2k}x^{n+3}. \tag{7}$$

Summing these inequalities provides us with an absolutely converging sum on the left side, and a sum with non-negative terms on the right side. Thus, the left sum is still bounded from above by the right sum. Up to using the change of variable $m \stackrel{\text{def}}{=\!=} n + 3 - 2k$, this means that

$$\begin{aligned} B(x)\left(L(x) - \ell_0 - \ell_1 x\right) &= \sum_{k\geqslant 2}\sum_{m\geqslant 2} \ell_m x^m b_{2k}x^{2k} \\ &= \sum_{k\geqslant 2}\sum_{n\geqslant 2k-1} \ell_{n+3-2k}b_{2k}x^{n+3} \\ &= \sum_{n\geqslant 0}\sum_{k=2}^{(n+1)/2} \ell_{n+3-2k}b_{2k}x^{n+3} \\ &\geqslant \sum_{n\geqslant 0} \sigma(\sigma-1)\left((\sigma-1)\ell_n - \ell_{n+1}\right)x^{n+3} \\ &\geqslant \sigma(\sigma-1)^2x^3\sum_{n\geqslant 0}\ell_n x^n - \sigma(\sigma-1)x^2\sum_{n\geqslant 1}\ell_n x^n \\ &\geqslant \sigma(\sigma-1)x^2\left((\sigma-1)xL(x) - L(x) + \ell_0\right). \end{aligned} \tag{8}$$

Since $\ell_0 = 1$ and $\ell_1 = \sigma$, the latter inequality can be rewritten as

$$\left(B(x) + \sigma(\sigma-1)x^2(1+x-\sigma x)\right)L(x) \geqslant (\sigma x+1)B(x) + \sigma(\sigma-1)x^2. \tag{9}$$

Then, one checks (e.g., by using a formal calculus software) that $(\sigma x+1)B(x) + \sigma(\sigma-1)x^2 > 0$ whenever $x \in (0, r)$, and that $B(x)+\sigma(\sigma-1)x^2(1+x-\sigma x) < 0$ whenever $x \in (\rho^{-1}, \rho_2^{-1})$, where $\rho_2$ is the second largest real root of the polynomial $P(X)$. Meanwhile, $L(x)$ has non-negative coefficients, thereby being positive on the interval $(0, s)$. Consequently, no real number belongs simultaneously to the three intervals $(0, r)$, $(0, s)$ and $(\rho^{-1}, \rho_2^{-1})$, which means that $s \leqslant \rho^{-1}$.

Furthermore, since the sequence $(\ell_n)_{n\geqslant 0}$ is sub-multiplicative, we know that $\ell_n \geqslant s^{-n}$ for all $n \geqslant 0$. It follows that $\ell_n \geqslant \rho^n$ for all $n \geqslant 0$.

## 4 Recognizing shuffled-square-free words

Deciding whether a word $u$ is a shuffled-square (*i.e.*, $u \in v \sqcup\!\sqcup v$ for some $v \in \Sigma^+$) is NP-complete [5,19], even for binary alphabets [4]. These results leave open the problem of recognizing words that do not contain any shuffled-square as a factor. This section is devoted to proving hardness of recognizing shuffled-square-free words on arbitrary alphabets.

**Theorem 2.** *Deciding shuffled-square-free words is* NP*-complete.*

We proceed by using a reduction from 3SAT. Our reduction can be decomposed into three steps. We successively translate each 3SAT formula $\varphi$ into a circuit $C^\varphi$, then into a graph $G^\varphi$, and finally into a word $w^\varphi$ whose strict factors are never shuffled-squares. Each positive instance of $\varphi$ is translated into a successful computation of $C^\varphi$, into a perfect matching of $G^\varphi$ that satisfies some *non-crossing* property, and into a decomposition of $w^\varphi$ into a shuffled-square. By contrast, if $\varphi$ is not satisfiable, $C^\varphi$ has no successful computation, no perfect matching of $G^\varphi$ has the non-crossing property, and $w^\varphi$ is not a shuffled-square.

*Step 1: From 3SAT formulæ to circuits.* Let $\varphi$ be a 3SAT formula with $m$ clauses, each formed of three literals, and $n$ variables. The problem 3SAT consists in deciding whether there exists a way to choose one literal $\ell$ in each clause (we say that this is a *clause-literal*) and one literal $x$ or $\overline{x}$ per variable (we say that this is a *variable-literal*) so that each chosen clause-literal coincides with a chosen variable-literal. Such choices, if they exist, form a *positive instance* of $\varphi$.

Given the formula $\varphi$, we build a rectangular circuit through which will run a certain number of wires. Each wire is associated with a clause- or variable-literal, and thus there are $3m + 2n$ wires. Wires interact as follows.

On each row of the circuit, the wires are ordered from left to right, each wire occupying one cell of the circuit grid, which is thus $3m + 2n$-column-wide. On the two extremities of each wire lie a *source gadget* and a *sink gadget*, which correspond with a given clause or variable. In each source gadget, we give a special status to one of the (two or three) wires that exit the gadget. This corresponds to choosing one literal per clause, and one of the two literals $x$ and $\overline{x}$ per variable $x$.

On the top row of the circuit, clause source gadgets occupy the leftmost cells, and variable source gadgets occupy the rightmost cells. On the bottom row of the circuit, clause source gadgets occupy the rightmost cells, and variable source gadgets occupy the leftmost cells. Thus, the challenge is to make each variable-literal wire cross each clause-literal wire.

The way wires cross each other is controlled as follows. In general, wires cannot cross each other, *i.e.*, swap the columns they occupy, unless they go through *swap gates*. Each swap gate occupies three adjacent cells on one row. Its input consists of one clause-literal wire $\ell$ and two variable-literal wires $x$ and $\overline{x}$, from left to right. Its output, one row below, consists in the variable-literal wires $x$ and $\overline{x}$ and then the clause-literal wire $\ell$.

There are three variants of swap gates. First, *cross* gates (or $\otimes$ gates) are used to make two variable-literal wires $x$ and $\overline{x}$ cross a clause-literal wire $\ell$ that concerns a variable distinct from $x$. Second, *positive* gates (or $\oplus$ gates) are used to make two variable-literal wires $x$ and $\overline{x}$ cross a clause-literal wire $x$. Such a gate triggers an error if both the clause-literal wire $x$ and the variable-literal wire $\overline{x}$ were chosen by their respective source gadgets. Similarly, *negative* gates (or $\ominus$ gates) are used to make two variable-literal wires $x$ and $\overline{x}$ cross a clause-literal wire $\overline{x}$. Such a gate triggers an error if both the clause-literal wire $\overline{x}$ and the variable-literal wire $x$ were chosen by their respective source gadgets.

In conclusion, when source gates select one of their outgoing wires, they conjointly choose a family of literals, which may or may not form a positive instance of the formula $\varphi$. This family is a positive instance of $\varphi$ if and only if no swap gate triggers any error.

In Figure 5, we present the circuit associated with a 3SAT formula $\varphi$, and a choice of literals that is *not* a positive instance of $\varphi$.

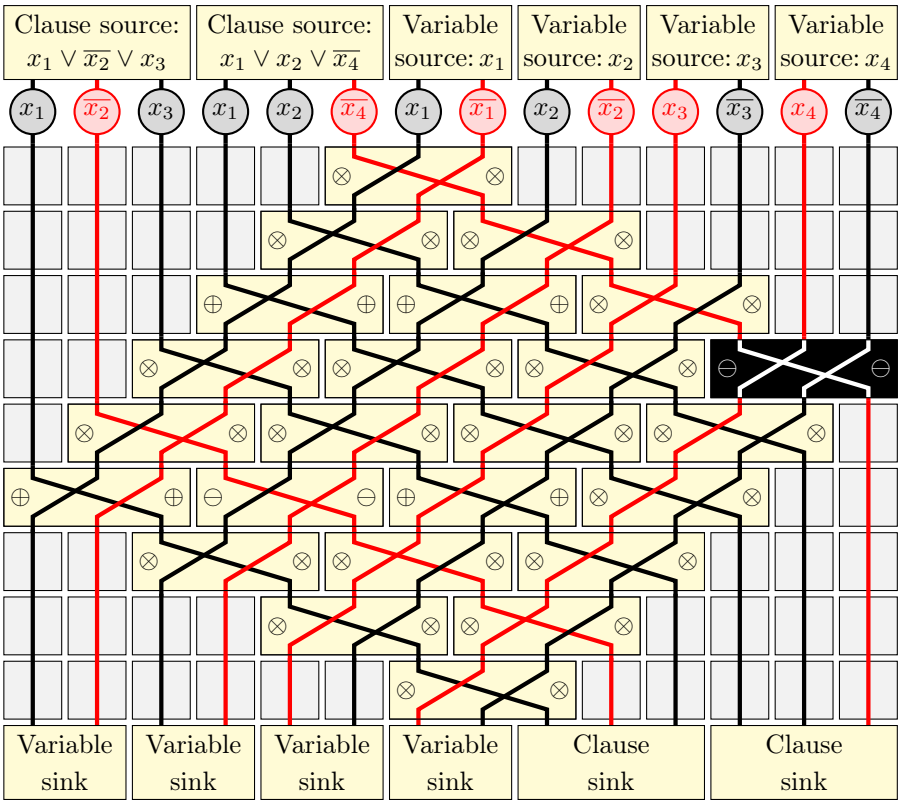



**Figure 5.** Circuit associated with the formula $\varphi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x_4})$. Wires associated with chosen literals are painted red. Other wires are painted black. Choosing the clause literals $\overline{x_2}, \overline{x_4}$ and the variable literals $\overline{x_1}, \overline{x_2}, x_3, x_4$ does *not* provide us with a positive instance of $\varphi$, and the $\ominus$ gate painted in black triggers an error. Had we chosen the variable literal $\overline{x_4}$ instead of $x_4$, we would have obtained a positive instance of $\varphi$, and no error would have been triggered.

*Step 2: From circuits to labeled graphs.* Our second step consists in transforming $C^\varphi$ into a labeled graph $G^\varphi$ constructed as follows. The vertices of $G^\varphi$ are placed in the cells of a rectangular grid, which may be seen as a subdivision of the rectangular grid containing the circuit $C^\varphi$. Then, two vertices of $G^\varphi$ are connected with each other if they lie in adjacent rows and if they have the same label. Identifying each edge with a segment between the two vertices it connects, this graph comes with a notion of *crossing* between edges.

Our transformation is designed to ensure that the 3SAT formula $\varphi$ from which we built the circuit $C^\varphi$ is satisfiable if and only if $G^\varphi$ has a *non-crossing* perfect matching, *i.e.*, if there is a collection of pairwise non-crossing edges of $G^\varphi$ such that each vertex of $G^\varphi$ belongs to exactly one edge of the collection.

In order to do so, we build the graph $G^\varphi$ with two kinds of connected components. Some connected components consist of two vertices and one edge. Such a component is obtained by giving its vertices a label that no other vertex uses. We say that such vertices and the edge that connects them are *blocking*. Indeed, when an edge $e$ is blocking, every perfect matching of $G^\varphi$ contains $e$, and thus non-crossing perfect matchings of $G^\varphi$ cannot contain any edge that crosses $e$. Thus, in what follows, we will represent edges only if they do not cross any blocking edge.

In addition to blocking components, $G^\varphi$ also contains connected components with arbitrarily many vertices and edges. Such components are meant to represent wires of the circuit $C^\varphi$ with a common clause or variable source, and we say that the edges involved in such large-size components are *main* edges of the graph.

The circuit $C^\varphi$ is composed of several circuit units: variable and clause sources, which have fan-out two or three; unit cell grids, with fan-in and fan-out one; swap gates, with fan-in and fan-out three; variable and clause sinks, which have fan-in two or three. Each circuit unit $\mathbf{U}$ covers one to three grid cells, and is translated into a graph unit $\overline{\mathbf{U}}$ that covers a tiny rectangle within the rectangular grid in which $G^\varphi$ is embedded.

A circuit unit $\mathbf{U}$ with fan-in $x$ and fan-out $y$ is translated into a graph unit $\overline{\mathbf{U}}$ with $3x$ half-edges on its top side and $3y$ half-edges on its bottom side. More precisely, the top side of $\overline{\mathbf{U}}$ contains $x$ main half-edges, each of which is surrounded by two blocking half-edges. Similarly, the bottom side of $\overline{\mathbf{U}}$ contains $y$ main half-edges, each of which is surrounded by two blocking half-edges. In addition, both the left and right sides of each graph unit are surrounded by blocking edges, and we make sure that any two main vertices are separated by at least one blocking vertex.

Below, let us order wires and edges from left to right. The $k^{\text{th}}$ wire that leaves a circuit unit $\mathbf{U}$ is translated into the $k^{\text{th}}$ main half-edge, say $e_k$, on the bottom side of $\overline{\mathbf{U}}$. Similarly, the $\ell^{\text{th}}$ wire that enters a circuit unit $\mathbf{U}'$ is translated into the $\ell^{\text{th}}$ main half-edge, say $e'_\ell$, on the top side of $\overline{\mathbf{U}'}$. Then, if the two wires coincide with each other, the half-edges $e_k$ and $e'_\ell$ belong to the same edge $e$, and the graph units $\overline{\mathbf{U}}$ and $\overline{\mathbf{U}'}$ share the edge $e$ as well as the two surrounding blocking edges.

Finally, we translate each choice of clause or variable literals of $C^\varphi$ into a non-crossing perfect matching of $G^\varphi$. This translation will fail if some gate of $C^\varphi$ triggers an error, and succeed otherwise. The translation is as follows: if the literal associated with a wire entering (and then leaving) a circuit unit $\mathbf{U}$ has been chosen, the two half-edges corresponding to that wire, in the top and bottom borders of the graph unit $\overline{\mathbf{U}}$, are *excluded* from the non-crossing perfect matching. If that literal is not chosen, these half-edges are *included* into the matching.

Thus, we keep following the convention from Figure 5 about coloring wires black or red depending on whether they were associated with chosen literals or not: in Figures 6 and 7 below, the main half-edges included in (portions of) non-crossing perfect matching are painted black, and the other half-edges are painted red. In both Figures 6 and 7, we represent the labels of main vertices, and omit the labels of blocking vertices.

Now, we present the different graph units that correspond to a given circuit unit. Each graph unit is represented as a small graph with half-edges, and we present the possible portions of non-crossing perfect graph matchings that concern the edges and half-edges of the graph unit.

Sources, sinks and unit cells are translated into the graph units represented in Figure 6:

1. each clause source is translated into ClauseSource;
2. each clause sink is translated into ClauseSink;
3. each variable source is translated into VariableSource;
4. each variable sink is translated into VariableSource;
5. each unit cell is translated into a version of Cell whose main vertices are labeled either by $c$ or by $v$, depending on if the unit cell contains a wire associated with a clause literal or with a variable literal.

The first four graph units are two-row-high, whereas Cell is eight-row-high.

Then, swap gates are translated into the eight-row-high graph units represented in Figure 7:

1. each cross gate is translated into $\mathsf{Gate}^{\otimes}$;
2. each positive gate is translated into $\mathsf{Gate}^{\oplus}$;
3. each negative gate is translated into $\mathsf{Gate}^{\ominus}$.

The graph unit $\mathsf{Gate}^{\otimes}$ has four (portions of) non-crossing perfect matchings. Each matching coincides with choosing one of the variable literals $x$ and $\overline{x}$, and either choosing (or not) the clause literal $\ell$. Thus, using $\mathsf{Gate}^{\otimes}$ allows these four literal choices.

Then, the graph unit $\mathsf{Gate}^{\oplus}$ is obtained by adding one blocking edge, which results in deleting one main edge of $\mathsf{Gate}^{\otimes}$ from ever belonging to a non-crossing perfect matching. As a result, the portion of non-crossing perfect matching that coincides with choosing both literals $\ell$ and $\overline{x}$ is no

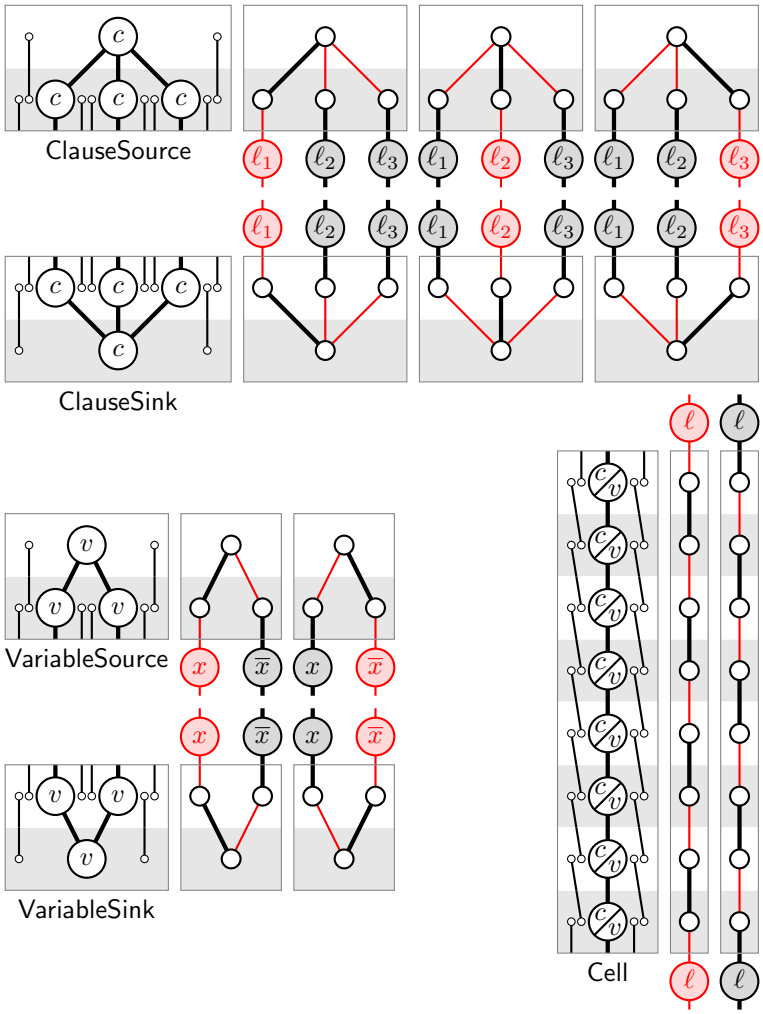



**Figure 6.** The source and sink associated with a clause $\ell_1 \vee \ell_2 \vee \ell_3$ are translated into the graph units ClauseSource and ClauseSink. The source and sink associated with a variable $x$ are translated into the graph units VariableSource and VariableSink. Each unit cell associated with a clause (resp., a variable) is translated into a graph unit Cell whose main vertices are labelled $c$ (resp., $v$). Thick edges represent the main edges of the graph units. The rows of the grid into which the graph $G^\varphi$ is scattered are alternatively painted white and gray.
Next to each graph unit are represented the portions of perfect non-crossing matchings that concern the unit's edges and half-edges, where blocking edges and half-edges have been omitted. Main edges and half-edges are painted black if they belong to the matching, and red otherwise. Each portion coincides with choosing zero or one wire, and we represent in red literals that were chosen: these are the clause literals $\ell_1$, $\ell_2$, $\ell_3$, the variable literals $x$, $\overline{x}$, and, possibly, the literal $\ell$ of the unit cell.

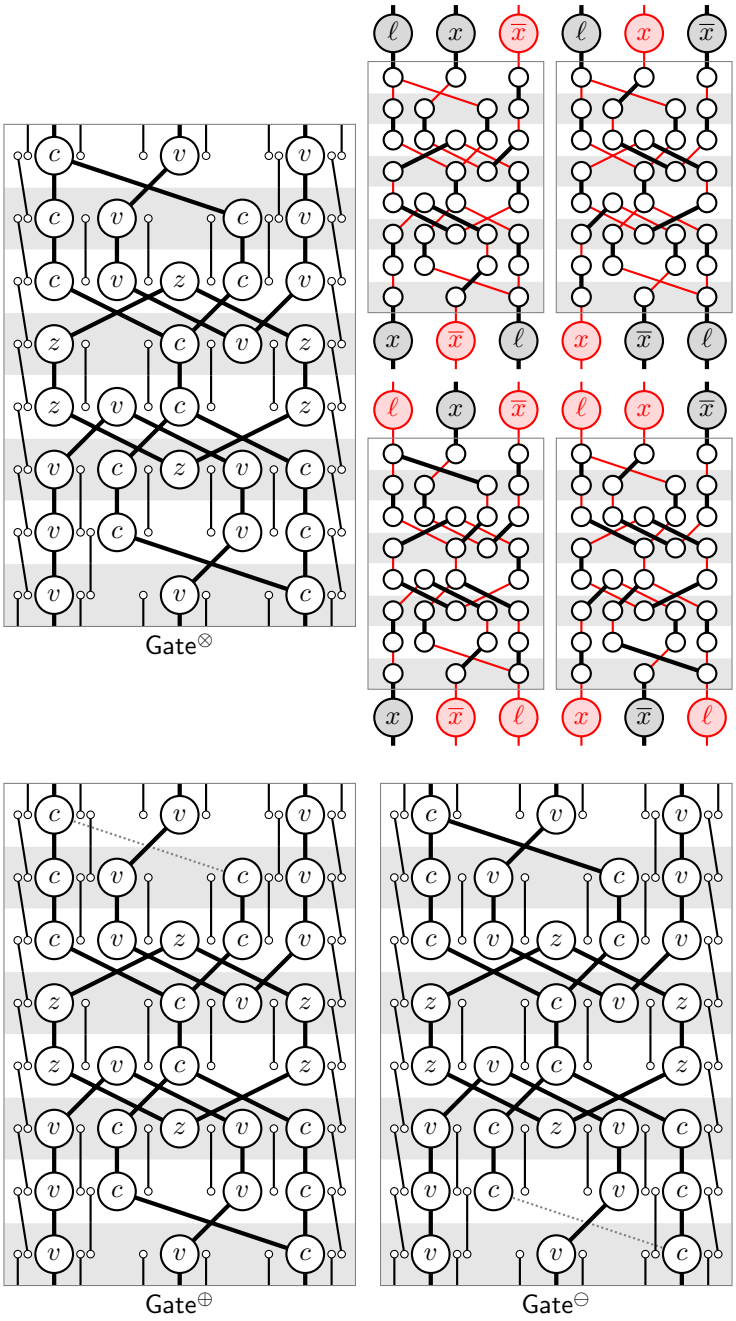



**Figure 7.** Swap gates involving one clause-literal $\ell$ and two variable-literals $x$ and $\overline{x}$ are translated into the graph units $\mathsf{Gate}^{\otimes}$, $\mathsf{Gate}^{\oplus}$ and $\mathsf{Gate}^{\ominus}$.
Next to the graph unit $\mathsf{Gate}^{\otimes}$ are represented the portions of perfect non-crossing matchings that concern the unit's edges and half-edges, where isolated edges have been omitted. Main edges and half-edges are painted black if they belong to the matching, and red otherwise. Above and below each portion are indicated those literals that were chosen.
The graph units $\mathsf{Gate}^{\oplus}$ and $\mathsf{Gate}^{\ominus}$ are obtained from $\mathsf{Gate}^{\otimes}$ by adding a blocking edge, which results in deleting one main edge (drawn as a dotted line) and preventing one portion of non-crossing perfect matching from being realized.

longer possible if we use $\mathsf{Gate}^{\oplus}$ instead of $\mathsf{Gate}^{\otimes}$. By using the same mechanisms, choosing both literals $\ell$ and $x$ is no longer possible if we use $\mathsf{Gate}^{\ominus}$ instead of $\mathsf{Gate}^{\otimes}$.

*Step 3: From labeled graphs to words.* Our third and last step consists in transforming $G^{\varphi}$ into a word $w^{\varphi}$. The letters of $w^{\varphi}$ are obtained by listing, from left to right, the letters on the top row of $G^{\varphi}$, then on the second top row of $G^{\varphi}$, ..., and finally on the bottom row of $G^{\varphi}$. If $G^{\varphi}$ has $n$ vertices, we thereby identify these vertices with the integers $1, 2, \ldots, n$, each vertex being identified with its position in the top-down, left-to-right ordering. If $G^{\varphi}$ consists of $\ell$ rows, we also order the rows of $G^{\varphi}$ from 1 to $\ell$, and we denote by $a_r$ and $b_r$ the smallest and largest vertices included in the row $r$, respectively.

Given a non-empty subset $X$ of the interval $\{1, 2, \ldots, n\}$ with $|X|$ elements, we denote by $X_k$ the $k^{\text{th}}$ element of $X$, for all $k \leqslant |X|$. Then, we denote by $w^{\varphi}_X$ the sub-word $w^{\varphi}_{X_1} w^{\varphi}_{X_2} \cdots w^{\varphi}_{X_{|X|}}$ of $w^{\varphi}$, and by $G^{\varphi}_X$ the sub-graph of $G^{\varphi}$ whose vertices belong to $X$. Finally, we say that the set $X$ is *matchable* if each vertex $x \in X$ belongs to some edge $(x, y)$ of $G^{\varphi}$ such that $y \in X$.

We prove now that, if $K$ is a matchable sub-interval of $\{1, 2, \ldots, n-1\}$, then $K = \{1, 2, \ldots, n\}$. Indeed, since no two consecutive vertices are main vertices, $K$ must contain a blocking vertex $x$, in a row $r$. Then, $K$ also contains the unique vertex $y$ to which is $x$ connected, and up to assuming that $x < y$, the vertex $y$ lies in the row $r + 1$. Consequently, $K$ contains the vertices $b_r$ and $a_{r+1}$.

Then, the graph units presented in Figures 6 and 7 were designed so that, for each row $r < \ell$, the vertex $a_r$ is a blocking vertex connected to a vertex in row $r + 1$, whereas the vertex $b_{r+1}$ is a blocking vertex connected to a vertex in row $r$. Hence, if $K$ contains a vertex $a_\rho$ (with $\rho < \ell$), it also contains $a_{\rho+1}$, and if $K$ contains the vertex $b_\rho$ (with $\rho > 1$), it also contains $b_{\rho-1}$.

Therefore, $K$ contains the vertices $b_1$ and $a_{r+1}$, so it also contains the vertex $a_2 + 1$, which is a blocking vertex connected to the vertex $a_1 = 1$. Similarly, $K$ contains the vertices $b_r$ and $a_\ell$, so it also contains the vertex $a_{\ell-1} - 1$, which is a blocking vertex connected to the vertex $b_\ell = n$. In particular, $K$ contains both vertices 1 and $n$, and $K = \{1, 2, \ldots, n\}$.

**Lemma 1.** *Let $(I, J)$ be a partition of a non-empty interval $K \subseteq \{1, 2, \ldots, n\}$ into sets of same cardinality such that $I_k < J_k$ for all $k < |I|$. The words $w^{\varphi}_I$ and $w^{\varphi}_J$ coincide with each other if and only if the graph $G^{\varphi}_K$ admits a non-crossing perfect matching $\mathbf{M}$ such that each edge in $\mathbf{M}$ has one endpoint in $I$ and one endpoint in $J$. Moreover, in that case, $K = \{1, 2, \ldots, n\}$.*

*Proof.* First, if such a matching $\mathbf{M}$ exists, let $(x, y)$ and $(x', y')$ be two edges of $\mathbf{M}$, with $x < y$, $x' < y'$ and $x < x'$. If $x$ and $x'$ are on the same row $r$, the vertices $y$ and $y'$ are on the row $r + 1$, and since $(x, y)$ and $(x', y')$ do not cross each other, it must be the case that $y < y'$. Otherwise, if $x$ is on some row $r$, the vertex $y$ is on the row $r + 1$, and $x'$ is on some row $r' \geqslant r + 1$, so that $y'$ is on the row $r' + 1 > r + 1$ and $y < y'$. Consequently, sorting the edges in $\mathbf{M}$ by the smallest endpoint or by their largest endpoint does not change anything. This proves that $\mathbf{M}$ connects each vertex $I_k$ to the vertex $J_k$, and since these vertices must have the same labels, the words $w^{\varphi}_I$ and $w^{\varphi}_J$ indeed coincide with each other.

Conversely, if a partition $(I, J)$ satisfies the conditions mentioned above, let us prove that the pairs $(I_k, J_k)$ form a non-crossing perfect matching of $G^{\varphi}_K$. The set $K$ is matchable, which already proves that $K = \{1, 2, \ldots, n\}$. Then, it suffices to prove that the vertices $I_k$ and $J_k$ always belong to adjacent rows: the pairs $(I_k, J_k)$ will then form a perfect matching $\mathbf{M}$ of $G^{\varphi}_K$, and the reasoning already used above will prove that this matching is non-crossing.

Let $r$ be the row to which belongs the vertex $I_k$. If $r$ is the top row, we are already done. Then, if $I_k$ is blocking, we also know that $J_k$ belongs to the row $r + 1$. Otherwise, note that $a_r \leqslant I_k \leqslant b_r - 1$ and that both blocking vertices $a_r$ and $b_r - 1$ are connected to vertices in row $r + 1$. It follows that $J_k$ is in row $r + 1$, which completes the proof. □

Consequently, if the graph $G^{\varphi}$ has a non-crossing perfect matching $\mathbf{M}$, we may just define $I$ as the set of vertices $x$ that are smaller than the vertex they are matched to, and $J$ as the set of vertices outside $I$. Lemma 1 then states precisely that $w^{\varphi}_I$ and $w^{\varphi}_J$ form a factorization

of $w^\varphi$ into a shuffled-square. Conversely, if some factor $w_K^\varphi$ is a shuffled-square, Lemma 1 proves that $w_K^\varphi = w^\varphi$ and that $G^\varphi$ has a non-crossing perfect matching.

In conclusion, the 3SAT formula $\varphi$ is satisfiable if and only if the circuit $C^\varphi$ has an execution that triggers no error, which happens if and only if the graph $G^\varphi$ has a non-crossing perfect matching, which happens if and only if the word $w^\varphi$ is a shuffled-square (and no strict factor of $w^\varphi$ is a shuffled-square). Moreover, constructing the word $w^\varphi$ from the formula $\varphi$ can be done in polynomial time. This completes the reduction and the proof of Theorem 2.

## 5 Concluding remarks

We have shown that there exist infinite shuffled-square-free words over an alphabet of size 6. This raises many questions and challenges, and we mention some of them. The most natural question is to ask about the minimum $k$ such that there exist infinite shuffled-square-free words over an alphabet of size $k$. We conjecture that there exist infinite shuffled-square-free words over an alphabet of size 4 (recall that there exist arbitrarily long square-free words in any alphabet with three letters, as proved by Thue).

As for generating words, Shur [20] has proposed an algorithm called R2F that can generate a square-free word of length $n$ over any alphabet with three or more letters. However, similarly to Guégan and Ochem, our approach relies on counting arguments and is inherently non-constructive. Therefore, two challenging problems are: (i) to decide for which alphabet sizes $|\Sigma|$, if any, there exists a polynomial-time algorithm that produces, when given an integer $n \geqslant 0$ as input, a shuffled-square-free-words of length $n$ over $\Sigma$; (ii) to decide the existence of an algorithm that checks, when given a finite alphabet $\Sigma$ and a word $w \in \Sigma^*$ as inputs, whether $w$ is a factor of an infinite shuffle-square-free word over the alphabet $\Sigma$.

## References


1. A. Apostolico and F.P. Preparata. Optimal off-line detection of repetitions in a string. *Theor. Comput. Sci.*, 22:297–315, 1983.
2. J. Berstel. Axel Thue's papers on repetitions in words: a translation. In *Monographies du LaCIM*, volume 11, pages 65–80. LaCIM, 1992. incollection.
3. F.-J. Brandenburg. Uniformly growing k-th power-free homomorphisms. *Theoretical Computer Science*, 23(1):69–82, 1988.
4. L. Bulteau and S. Vialette. Recognizing binary shuffle squares is NP-hard. *Theoretical Computer Science*, 806:116–132, 2020.
5. S. Buss and M. Soltys. Unshuffling a square is NP-hard. *J. Comput. Syst. Sci.*, 80(4):766–776, 2014.
6. C. Choffrut and J. Karhumäki. Combinatorics of words. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Volume 1: Word, Language, Grammar*, pages 329–438. Springer, 1997.
7. M. Crochemore. An optimal algorithm for computing the repetitions in a word. *Inf. Process. Lett.*, 12(5):244–250, 1981.
8. M. Crochemore, J.D. Currie, G. Kucherov, and D. Nowotka. Combinatorics and algorithmics of strings (Dagstuhl seminar 14111). *Dagstuhl Reports*, 4(3):28–46, 2014.
9. J.D. Currie. Shuffle squares are avoidable. unpublished, 2014.
10. A.S. Fraenkel and J. Simpson. How many squares must a binary sequence contain? *Electron. J. Comb.*, 2, 1995.
11. J. Grytczuk, H. Kordulewski, and A. Niewiadomski. Extremal square-free words. *Electron. J. Comb.*, 27(1):P1.48, 2020.
12. G. Guégan and P. Ochem. A short proof that shuffle squares are 7-avoidable. *RAIRO Theor. Informatics Appl.*, 50(1):101–103, 2016.
13. L. Ilie. A note on the number of squares in a word. *Theoretical Computer Science*, 380(3):373–376, 2007.
14. D. Kőnig. Über eine Schlussweise aus dem Endlichen ins Unendliche. *Acta litt. sci. Reg. Univ. Hung. Francisco-Josephinae, Sect. sci. math.*, 3(2-3):121–130, June 1927.
15. M. Lothaire. *Applied Combinatorics on Words*. Cambridge University Press, 2005.
16. M.G. Main and R.J. Lorentz. An O(n log n) algorithm for finding all repetitions in a string. *J. Algorithms*, 5(3):422–432, 1984.
17. L. Mol and N. Rampersad. Lengths of extremal square-free ternary words. *Contributions Discret. Math.*, 16(1):8–19, 2021.

18. M. Müller. *Avoiding and Enforcing Repetitive Structures in Words*. PhD thesis, University of Kiel, 2015.
19. R. Rizzi and S. Vialette. On recognising words that are squares for the shuffle product. *Theoretical Computer Science*, 2017.
20. A.M. Shur. Generating square-free words efficiently. *Theoretical Computer Science*, 601:67–72, 2015.
21. A. Thue. Über die gegenseitige Lage gleicher Teile gewisser Zeichenreihen.
22. A. Thue. Über unendliche Zeichenreihen. Christiania Vidensk.-Selsk. Skr. 1906, Nr 7, 22 S., 1906.