On the Problem of Deciding If a Polyomino Tiles the Plane by Translation*

Srečko Brlek and Xavier Provençal

Laboratoire de Combinatoire et d'Informatique Mathématique,
Université du Québec à Montréal,
CP 8888, Succ. Centre-ville, Montréal (QC) Canada H3C3P8
[brlek,provenca]@lacim.uqam.ca

Abstract. The words that tile the plane by translation are characterized by the Beauquier-Nivat condition. By using the constant time algorithms for computing the longest common extensions in two words, we provide a linear time algorithm in the case of pseudo-square polyominoes, improving the previous quadratic algorithm of Gambini and Vuillon. For pseudo-hexagon polyominoes not containing arbitrarily large square factors we also have a linear algorithm.

Keywords: tiling polyominoes, plane tesselation, longest common extensions

1 Introduction

The way of tiling planar surfaces takes its roots in the ancient times for decorative purposes. More recently, connections were established with computational theory, mathematical logic and discrete geometry, where tilings are often regarded as basic objects for proving undecidability results for planar problems. Tilings have been also used in physics, as powerful tools for studying quasi-crystal structures: in particular these structures can be better understood by representing them as rigid tilings decorated by atoms in a uniform fashion. Their long-range order can consequently be investigated in a purely geometrical framework, after assigning to every tiling a structural energy.

A classical result of Berger [2] states that given a set of tiles, it is not decidable whether there exists a tiling of the plane which involves all its elements. This result has been achieved by constructing an aperiodic set of tiles, and it has been strengthened afterwards by Gurevich and Koriakov [12] to the periodic case.

It was therefore natural to seek manageable problems, and polyominoes appeared as good candidates. Invented by Golomb [10] who coined the term *polyomino*, these objects, also called n-ominoes or lattice animals, gained some interest after being popularized by Gardner in mathematical games [9]. In statistical physics they appear as models for percolation theory and their combinatorial properties have been extensively studied. These nowadays well studied combinatorial objects are still related to many challenging problems, such as tiling problems [5, 11], games [9], among many others (see Weisstein [18] for more pointers). Their enumeration is also an open problem despite the fact that restricted classes have been fully described.

There are different types of polyominoes and here we consider a *polyomino* as a finite union of unit lattice closed squares (pixels) in the discrete plane whose boundary consists of a simple closed polygonal path using 4-connectedness (Figure 2(a)). In particular, polyominoes are simply connected (contain no holes), and have no multiple points (Figure 1(a)).

* with the support of NSERC (Canada)

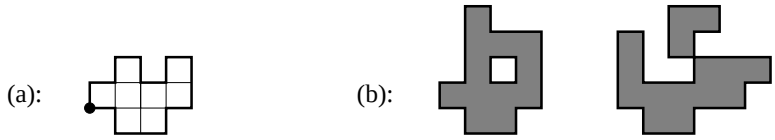


Figure 1. (a) a polyomino; (b) not a polyomino

The problem of deciding if a given polyomino tiles the plane by translation goes back to Wisjhoff and Van Leeuven [19] who coined the term *exact polyomino* for these, and also provided a polynomial $\mathcal{O}(n^4)$ algorithm for solving the problem. Polyominoes may be coded by words on a 4-letter alphabet $\Sigma = \{a, \overline{a}, b, \overline{b}\}$, also known as the Freeman chain codes [6, 7], coding their boundaries (see [3] for further reading). For

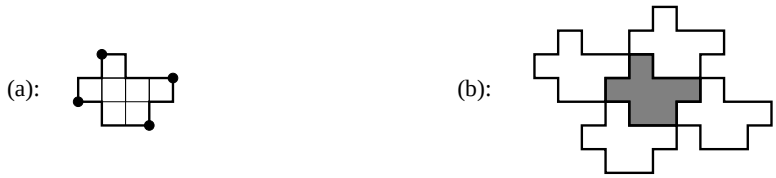


Figure 2. (a) an exact polyomino; (b) the associated tiling

instance, the boundary $\mathbf{b}(P)$ of the polyomino in Figure 2 (a), in a counterclockwise manner, is coded by the word $w = a\,\overline{b}\,a\,a\,b\,a\,b\,b\,\overline{a}\,\overline{b}\,\overline{a}\,b\,\overline{a}\,\overline{b}\,\overline{a}\,\overline{b}$.

Observe that we may consider the words as circular which avoids a fixed origin. The *perimeter* of a polyomino P is the length of its boundary word $\mathbf{b}(P)$ and is of even length $2n$. Beauquier and Nivat [1] gave a characterization stating that the boundary of such a polyomino P may be factorized (not necessarily in a unique way) as

$$\mathbf{b}(P) = A \cdot B \cdot C \cdot \widehat{A} \cdot \widehat{B} \cdot \widehat{C} \tag{1}$$

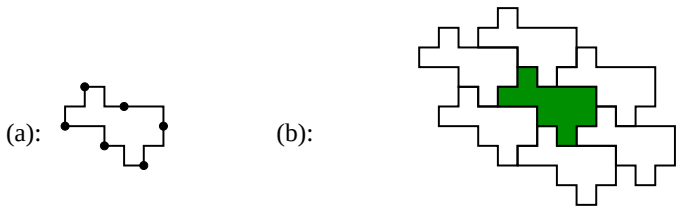
where at most one of the variables is possibly empty. The operation $\widehat{(\ )}$ appearing in (1) is defined by $\widehat{U} = \widetilde{\overline{U}}$, where $\widetilde{(\ )}$ is the usual reversal operation and $\overline{(\ )}$ the complement on $\Sigma = \{a, \overline{a}, b, \overline{b}\}$. For instance, the exact polyomino in Figure 2 (b) is coded by the circular word

$$w = a\,a\,b\,a\,b\,\overline{a}\,\overline{a}\,b\,\overline{a}\,\overline{b}\,\overline{a}\,\overline{b}\,a\,\overline{b},$$

its semi-perimeter is 7, and its boundary may be factorized as

$$\mathbf{b}(P) = A \cdot B \cdot \widehat{A} \cdot \widehat{B} = a\,\overline{b}\,a\,a \cdot b\,a\,b \cdot \overline{a}\,\overline{a}\,b\,\overline{a} \cdot \overline{b}\,\overline{a}\,\overline{b},$$

and for the exact polyomino P' below



the boundary may be factorized as

$$\mathbf{b}(P') \equiv a\,a\,\overline{b}\cdot a\,\overline{b}\,a\cdot b\,a\,b\cdot b\,\overline{a}\,\overline{a}\cdot\overline{a}\,b\,\overline{a}\cdot\overline{b}\,\overline{a}\,\overline{b}.$$

Determining if a given word $w \in \Sigma^n$ is the boundary of a polyomino is computed in $\mathcal{O}(n)$. Therefore the problem reduces to finding a factorization satisfying the Beauquier and Nivat condition. Recently, Gambini and Vuillon [8] improved the Wisjhoof-van Leeuven bound by designing an $\mathcal{O}(n^2)$ algorithm that checks the Beauquier-Nivat condition (1).

The underlying idea of our approach is to search efficiently the pairs of homologue factors $X, \widehat{X}$. Our algorithms borrow from Lothaire [16] (for instance) that the *Longest-Common-Factor*, the *Longest-Common-Prefix* and the *Longest-Common-Suffix* in two words may be computed in linear time. The approach is also inspired by the linear algorithm of Gusfield and Stoye [14] for detecting *tandem repeats* in a word, and by the linear algorithm used to detect repetitions with gaps, as shown in Lothaire [16]. More precisely, the computation of the *Longest-Common-Left-Extension* (LCLE(u, v)) and *Longest-Common-Right-Extension* (LCRE(u, v)) is achieved in constant time, provided a linear pre-processing is performed on u and v, by a clever utilization of suffix-trees (see Gusfield [13]).

Taking advantage of these algorithms we provide a linear algorithm, with respect to the length of words, for pseudo-square polyominoes (Theorem 8). We establish a first step in order to provide a linear algorithm for pseudo-hexagons as well. Indeed, for boundary words not having two large square repetitions there is a linear algorithm to decide whether a polyomino tiles the plane by translation or not (Theorem 11).

2 Preliminaries

Let Σ be a finite alphabet whose elements are called *letters*. Finite words are sequences of letters, that is, functions $w : [0..n-1] \longrightarrow \Sigma$, and the set of words of length n is denoted Σ^n. The free monoid $\Sigma^* = \cup_{n=0}^{\infty}\Sigma^n$ is the set of all finite words and the empty word is denoted ϵ.

A morphism is a function $\sigma : \Sigma^* \longrightarrow \Sigma^*$ such that $\sigma(uv) = \sigma(u)\sigma(v)$. Clearly a morphism is defined by the image of the letters. A *factor* f of w is a word $f \in \Sigma^*$ satisfying

$$\exists x \in \Sigma^*, y \in \Sigma^*, w = xfy.$$

If $x = \epsilon$ (resp. $y = \epsilon$) then f is called *prefix* (resp. *suffix*). The set of all factors of w is denoted by $F(w)$, and those of length n is $F_n(w) = F(w) \cap \Sigma^n$. Finally Pref$(w)$ denotes the set of all prefixes of w. The length of a word w is $|w|$, and the number of occurrences of a factor $f \in \Sigma^*$ is $|w|_f$. A word is said to be *primitive* if it is not a power of another word. If $w = pu$, and $|w| = n, |p| = k$, then $p^{-1}w = w[k+1]..w[n-1] = u$ is the word obtained by erasing p. As a special case when $|p| = 1$ we have the shift operator σ defined by $\sigma(w) = w[1..(n-1)]$. Another useful operator is the circular permutation ρ defined by $\rho(w) = w[1..(n-1)] \cdot w[0]$.

Two words u and v are *conjugate* when there are words x, y such that $u = xy$ and $v = yx$. Equivalently, u and v are conjugate if and only if there exists an index k such that $v = \rho^k(v)$. Conjugation is an equivalence relation written $u \equiv v$. The *reversal* $\widetilde{u}$ of $u = u_1u_2\cdots u_n \in \Sigma^n$ is the word $\widetilde{u} = u_nu_{n-1}\cdots u_1$. A *palindrome* is a word p such that $p = \widetilde{p}$, and for a language $L \subseteq \Sigma^\infty$, we denote by Pal$(L)$ the set of its palindromic factors.

Paths on the square lattice $\mathbb{Z} \times \mathbb{Z}$ are encoded on the alphabet $\Sigma = \{a, \overline{a}, b, \overline{b}\}$ identified with the unit steps $\{\rightarrow, \leftarrow, \uparrow, \downarrow\}$. Parallel paths always define a translation and we say that two words are *homologue* when the corresponding paths define a translation. More precisely, two words u and v are said to be *homologue* when either

(i) $u = v$, or
(ii) $u = \widehat{v}$.

An exact polyomino P whose boundary is $\mathbf{b}(P) = A \cdot B \cdot C \cdot \widehat{A} \cdot \widehat{B} \cdot \widehat{C}$ is called a *pseudo-hexagon* if none of the variables is empty and a *pseudo-square* otherwise. In this factorization A (resp. B, C) and $\widehat{A}$ (resp. $\widehat{B}, \widehat{C}$) are homologue and define the respective translations. For instance, the translations defined by the homologue sides of the pseudo-square polyomino

$$\mathbf{b}(P) = A \cdot B \cdot \widehat{A} \cdot \widehat{B} = a\,\overline{b}\,a\,a \cdot b\,a\,b \cdot \overline{a}\,\overline{a}\,b\,\overline{a} \cdot \overline{b}\,\overline{a}\,\overline{b}$$

are shown in Figure 3 (a). In the case of a pseudo-hexagon, as in Figure 3(b), the

Figure 3. Translations defined by homologue sides of a polyomino tile

translations are related by the relation $t_3 = t_1 + t_2$. Moreover, the relative positions of the starting and ending point of any path is completely determined by the sum of the unit vectors corresponding to each letter. By abuse of notation we write for a path $w : [0..n-1] \rightarrow \Sigma$

$$\overrightarrow{w} = \sum_{k=0}^{n-1} \overrightarrow{w_k}.$$

Note that $\overrightarrow{w} = 0$ if and only if w is a closed path, and that $\overrightarrow{u} = -\overrightarrow{\widehat{u}}$.

3 Searching the homologue factors

Since polyominoes are coded by circular words w, in order to find the homologue factors it is convenient to work with $w \cdot w$ since a pair of homologue factors might be split, depending on the starting point.

Therefore, finding the homologue factors amounts to look for the *longest common factor* of ww and $\widehat{ww}$ denoted $\mathrm{LCF}(ww, \widehat{ww})$.

For instance the longest common factors of the polyomino-tile P in Figure 2 (b) are

$$\mathrm{LCF}(ww, \widehat{ww}) = \{a\,\overline{b}\,a\,a, \overline{a}\,\overline{a}\,b\,\overline{a}\}$$

and they are necessarily homologue sides(!). Indeed, since we know the positions i and j of $a\,b\,a\,a$ and $\overline{a}\,\overline{a}\,b\overline{a}$ in w, this is easy to check in linear time. Clearly the boundary of P may be written as

$$\mathbf{b}(P) = w = a\,\overline{b}\,a\,a \cdot u \cdot \overline{a}\,\overline{a}\,b\,\overline{a} \cdot v$$

and then one easily checks that $v = \widehat{u}$. Unfortunately the situation is not always that good. Indeed, let $w = a\,a\,b\,b\,b\,a\,a\,b\,\overline{a}\,\overline{a}\,b\,\overline{a}\,\overline{b}\,\overline{b}\,\overline{b}\,\overline{a}\,\overline{a}\,\overline{b}\,a\,\overline{b}$. Then the longest homologue factors of w are (see Figure 4)

$$\mathrm{LCF}(ww, \widehat{ww}) = \{a\,a\,b\,b\,b\,a, \overline{a}\,\overline{b}\,\overline{b}\,\overline{b}\,\overline{a}\,\overline{a}\},$$

but $w = a\,a\,b\,b\,b\,a \cdot a\,b\,\overline{a}\,\overline{a}\,b \cdot \overline{a}\,\overline{b}\,\overline{b}\,\overline{b}\,\overline{a}\,\overline{a} \cdot \overline{b}\,a\,\overline{b}$ does not satisfy the Beauquier-Nivat condition. A good factorization is $w \equiv b\,b \cdot b\,a\,a\,b \cdot \overline{a}\,\overline{a}\,b\,\overline{a} \cdot \overline{b}\,\overline{b} \cdot \overline{b}\,\overline{a}\,\overline{a}\,\overline{b} \cdot a\,\overline{b}\,a\,a$. This means

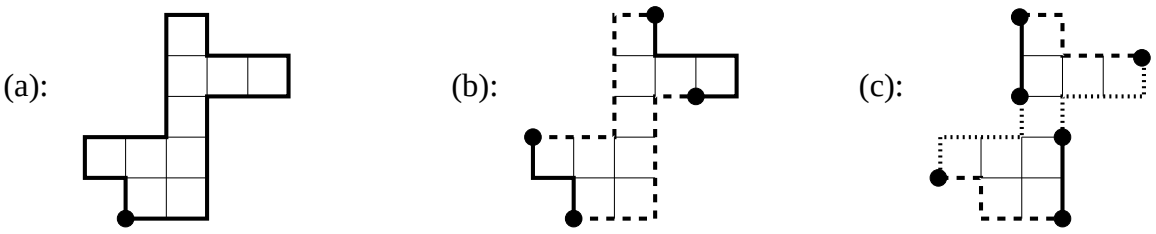


Figure 4. (b) longest homologue factors; (c) a good factorization

that not all the homologue factors provide a factorization, and good candidates are those separated by factors of same length.

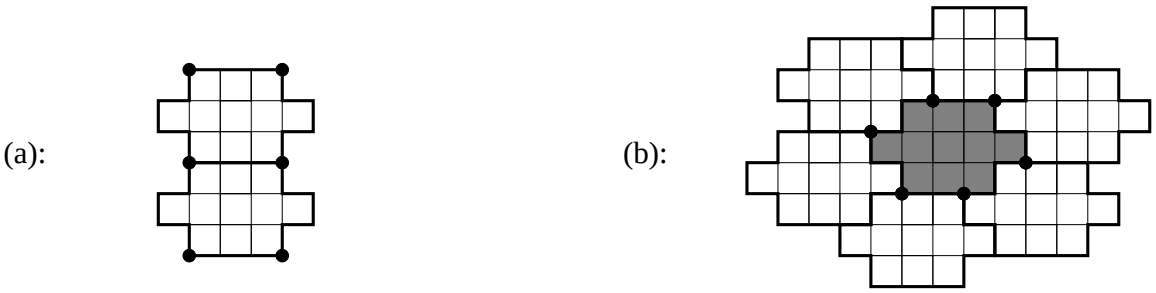
Definition 1. *Let $w \equiv \mathbf{b}(P)$ be the boundary word of a polyomino P. A factor A of w is* admissible *if*

(i) *$w \equiv Ax\widehat{A}y$, for some x, y such that $|x| = |y|$;*
(ii) *A is* saturated*, that is, $x_0 \neq \overline{x_{k-1}}$ and $y_0 \neq \overline{y_{k-1}}$ where $k = |x| = |y|$.*

Nevertheless, admissibility is ensured for words that code the boundary of polyominoes. Indeed, Gambini and Vuillon established the following property ([8], section 3.1) by using a geometric result of Daurat and Nivat [5].

Lemma 2. *Let $w \equiv ABC\widehat{A}\widehat{B}\widehat{C}$ be a Beauquier-Nivat factorization of the boundary $\mathbf{b}(P)$ of an exact polyomino P. Then A, B and C are admissible.*

Conversely, not all admissible factors lead to a Beauquier-Nivat factorization. For instance, in the polyomino $w \equiv a\,a\,a\,b\,a\,b\,\overline{a}\,b\,\overline{a}\,\overline{a}\,\overline{a}\,\overline{b}\,\overline{a}\,\overline{b}\,a\,\overline{b}$ shown below the factor aaa



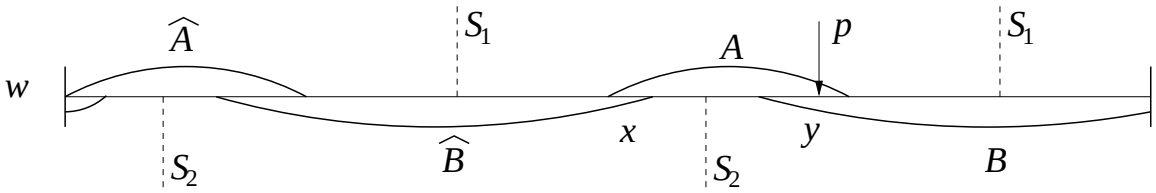
is admissible but does not provide a correct factorization of w. Indeed, $A = aa$ is the admissible factor ($w = Ax\widehat{A}y$ with $x = abab\overline{a}b$, $y = \overline{a}\overline{b}\overline{a}\overline{b}a\overline{b}$) yielding a correct factorization with $B = aba$ and $C = b\overline{a}b$:

$$w \equiv a\,a \cdot a\,b\,a \cdot b\,\overline{a}\,b \cdot \overline{a}\,\overline{a} \cdot \overline{a}\,\overline{b}\,\overline{a} \cdot \overline{b}\,a\,\overline{b}.$$

The following proposition establishes a useful property.

Proposition 3. *Let $w = \mathbf{b}(P) \in \Sigma^{2n}$ be the contour of a polyomino P and let p be any fixed position in w. Let X be the set of all admissible factors overlapping the position p and $\widehat{X}$ be the set of their respective homologue factors. Then, there exist at least one position in w that is not covered by any element of $X \cup \widehat{X}$.*

Proof. By contradiction, assume that there is no such point. Let $A \in X$ be the factor that starts at the leftmost position and $B \in X$ be the one that ends at the righmost position as shown below. The homologue factors $A, \widehat{A}$ and $B, \widehat{B}$ always define two

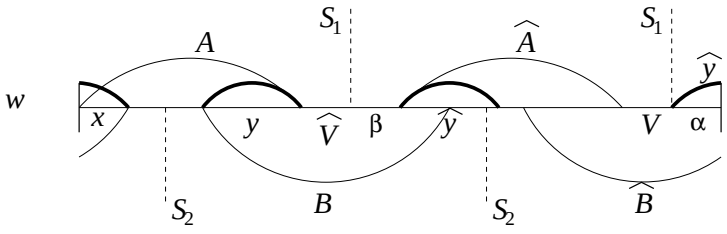


symmetries denoted respectively by S_1 and S_2. Let x be the overlap between A and $\widehat{B}$, and y be the overlap between A and B. Without loss of generality we may consider that $|y| \geq |x|$. If $|x| = |y|$ the symmetry implies that $x = \widehat{y}$ and the factorization is

$$w \equiv \mathbf{x}\, U\, \widehat{x}\, V\, x\, \widehat{U}\, \widehat{x}\, \widehat{V}\,. \tag{2}$$

We use a property proved in Brlek et all. ([4], DLT2005) that simplifies a result of Daurat and Nivat ([5], IWCIA'03) on the number of salient and reentrant points of discrete sets: indeed, the number of right turns minus the number of left turns in a closed and non-intersecting path on a square lattice is 4. In equation 2, notice that all turns in a factor are cancelled by those of its homologue. Therefore we only have to consider the turns between consecutive factors. Reading, the word w from left to right, we see that each pair of consecutive factors is cancelled by its homologue: $\mathbf{x}\,U$ is cancelled by $\widehat{U}\,\widehat{x}$, $U\,\widehat{x}$ by $x\,\widehat{U}$, $\widehat{x}\,V$ by $\widehat{V}\mathbf{x}$ (the word w is circular), and $V\,x$ by $\widehat{x}\,\widehat{V}$. Hence the difference between right and left turns is 0, and w is self intersecting. Contradiction.

If $|x| \neq |y|$ we have the following situation where the factor y (thick line) propa-

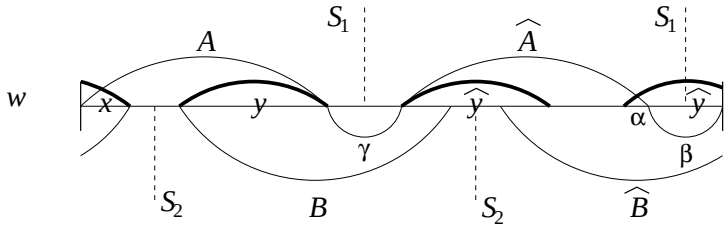


gates as shown by using the symmetries S_1 and S_2. In this case $\widehat{y}$ does not overlap $\widehat{A}$ in $\widehat{B}$, so let V be the factor between $\widehat{A}$ and $\widehat{y}$. We have the following factorization

$$w \equiv A\,\widehat{V}\,\beta\,\widehat{A}\,V\,\alpha\,.$$

Passing to vectors, and using commutativity of addition, we have

$$\overrightarrow{w} = \overrightarrow{A} + \overrightarrow{\widehat{A}} + \overrightarrow{V} + \overrightarrow{\widehat{V}} + \overrightarrow{\beta} + \overrightarrow{\alpha} = \overrightarrow{\beta} + \overrightarrow{\alpha} = \overrightarrow{0}\,.$$



But $\widehat{y} = \alpha x$, so that β is followed by α in w. Therefore $\beta\alpha$ is a nonempty closed path on the boundary of P. Contradiction.

In the case where $\widehat{y}$ does overlap $\widehat{A}$ in $\widehat{B}$ we have the following situation where $\overrightarrow{\gamma} + \overrightarrow{\beta} = 0$ (by closure property $\overrightarrow{w} = 0 = \overrightarrow{A} + \overrightarrow{\gamma} + \overrightarrow{\widehat{A}} + \overrightarrow{\beta}$). Moreover, $\widehat{y} = \alpha\beta x$, so that $y\,\gamma\,\widehat{y}$ contains the nonempty factor $\widehat{\alpha}\,\gamma\,\alpha\,\beta$ corresponding to a closed path. Contradiction. □

Proposition 3 specializes for pseudo-squares as follows. Assume that a pseudo-square P has two factorizations

$$w = \mathbf{b}(P) \equiv AB\widehat{A}\widehat{B} \equiv XY\widehat{X}\widehat{Y}$$

where $A = sXt$. Then, by using the same argument as in the proof above, the boundary of P contains a loop yielding a contradiction.

Corollary 4. *If $w = \mathbf{b}(P) \equiv AB\widehat{A}\widehat{B} \equiv XY\widehat{X}\widehat{Y}$ are two distinct factorizations of the boundary of a pseudo-square P, then there exist α, β, γ such that $A = \alpha\beta$ and $X = \beta\gamma$.*

As an exemple we have the following pseudo-square

$$aba \cdot \overline{b}a\overline{b} \cdot \overline{a}\overline{b}\overline{a} \cdot b\overline{a}b \equiv bab \cdot a\overline{b}a \cdot \overline{b}\overline{a}\overline{b} \cdot \overline{a}b\overline{a},$$

showing two distinct factorizations. The problem of enumerating all the factorizations of a given pseudo-square will be addressed in a forthcoming paper.

3.1 A linear time algorithm for detecting pseudo-squares

The main idea used to achieve linear time factorization, is to choose a position p in w and then list all the *admissible factors* A that overlap this fixed position. The following auxiliary functions are useful. The *Longest-Common-Right-Extension* (LCRE) and *Longest-Common-Left-Extension* (LCLE) of two words u and v at positions respectively m and n are partial functions

$$\mathrm{LCRE}, \mathrm{LCLE} : \Sigma^* \times \Sigma^* \times \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$$

defined as follows. For $u, v \in \Sigma^*$, let m and n be such that $0 \leq m \leq |u|$ and $0 \leq n \leq |v|$, then

$$\begin{aligned} \mathrm{LCRE}(u, v, m, n) &= \mathrm{LCP}(\rho^m(u), \rho^n(v)) \\ \mathrm{LCLE}(u, v, m, n) &= \mathrm{LCS}(\rho^{|u|-m}(u), \rho^{|v|-n}(v)) \end{aligned}$$

Remark 5. It is clear from the definition above that LCRE and LCLE may be computed in linear time. Their computation may also be performed directly by the following formulas. Since we use circular words w, denote $\underline{m} = m \mod |w|$. If $u[\underline{m}] = v[\underline{n}]$ then

(i) $\mathrm{LCRE}(u, v, \underline{m}, \underline{n}) = \mathbf{max}\{k \in \mathbb{N} \mid u[\underline{m}..(\underline{m} + \underline{k})] = v[\underline{n}..(\underline{n} + \underline{k})]\} + 1,$
(ii) $\mathrm{LCLE}(u, v, \underline{m}, \underline{n}) = \mathbf{max}\{k \in \mathbb{N} \mid u[(\underline{m} - \underline{k})..\underline{m}] = v[(\underline{n} - \underline{k})..\underline{n}]\} + 1,$

and, otherwise, $\mathrm{LCRE}(u, v, \underline{m}, \underline{n}) = \mathrm{LCLE}(u, v, \underline{m}, \underline{n}) = 0$.

For example, if $u = aabbbaabaababababa$, $v = babaabbbaabbababababb$, $i = 4$ and $j = 7$ then (note that the words all starts at position 0) we have

$$u = \underline{a\,a\,b\,b}\,\mathbf{b}\,\underline{a\,a\,b}\,a\,a\,b\,a\,b\,a\,b\,a\,b\,a,$$
$$v = b\,a\,b\,\underline{a\,a\,b\,b}\,\mathbf{b}\,\underline{a\,a\,b}\,b\,a\,b\,a\,b\,a\,b\,b,$$

and $\mathrm{LCRE}(u, v, 4, 7) = 4, \mathrm{LCLE}(u, v, 4, 7) = 5$. On the other hand LCRE(u,v,4,1) = LCLE(u,v,4,1)=0.

Later we will need to perform these computations $\mathcal{O}(n)$ times. Fortunately, the computation of LCLE and LCRE is achieved in constant time, provided a linear pre-processing is performed on u and v, by a clever utilization of suffix-trees (see Gusfield [13], section 9.1, or Gusfield and Stoye [14], page 531).

Lemma 6. *Let $w = \mathbf{b}(P)$ be the boundary of P. For each occurrence of A in w and each occurrence of A in $\widehat{w}$, whether A is admissible or not is decidable in constant time.*

Proof. Given an occurrence of A in $\widehat{w}$, one computes in constant time the corresponding position of $\widehat{A}$ in w. If $\widehat{A}$ overlaps A in w is decidable in constant time. If $\widehat{A}$ and A do not overlap then, $u \equiv Ax\widehat{A}y$ and A is an admissible factor, by definition, if and only if the three following conditions are verified : $|x| = |y|$, $x_0 \neq \overline{x_{k-1}}$ and $y_0 \neq \overline{y_{k-1}}$ where $k = |x| = |y|$. □

Lemma 7. *Let $w = \mathbf{b}(P) \in \Sigma^{2n}$ be the boundary of P. For any position p in w, listing all the admissible factors overlapping p is computed in linear time.*

Proof. The following algorithm lists all admissible factors containing the p-th letter w. Since the longest common right and left extension problem can be solved in constant time after linear time pre-processing.

Algorithm 1
Input: $w = \mathbf{b}(P) \in \Sigma^{2n}$
0 : *Pre-process w and $\widehat{w}$ so that* LCLE *and* LCRE *take constant time*
1 : **For** $i := 0$ *to* $2n - 1$ **do**
2 : $l := \mathrm{LCLE}(w, \widehat{w}, p, i) - 1$
3 : $r := \mathrm{LCRE}(w, \widehat{w}, p, i) - 1$
4 : $A := w[p - l, \ldots, p + r]$
5 : **If** $w \equiv Ax\widehat{A}y$ *with* $|x| = |y|$ **then**
6 : *Add A to the list of admissible factors.*
7 : **end if**
8 : **end for**

Using the modulo in managing the positions is superfluous because we may assume, without loss of generality (since w is a circular word) that $p = n$. Note that, by definition of LCRE and LCLE, the factor A in this algorithm is necessarily saturated. As shown in Lemma 6, the condition can be tested in constant time by direct computation of positions in w. Finally, the loop is performed exactly $2n$ times. □

This lemma implies that the number of admissible factors in a word is linear. To determine a precise upper bound remains an open problem which is similar to the problem of determining a tight upper bound for the number of distinct squares in a word (see for instance Lothaire [16] or Ilie [15]).

Theorem 8. *Let $w = \mathbf{b}(P) \in \Sigma^{2n}$ be the boundary of P. Determining if w codes a* pseudo-square *is decidable in linear time.*

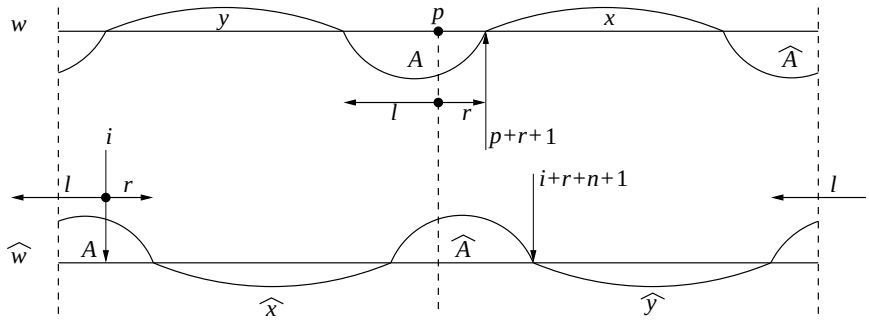


Figure 5. An admissible factor A in w and $\widehat{w}$

Proof. If w encodes an exact polyomino, any position belongs to some admissible factor of the Beauquier-Nivat factorization. Therefore, it suffices to apply Lemma 7 to an arbitrary position p. Then, Algorithm 1 provides the list of all admissible factors overlapping the position p, and it only remains to check, for each admissible factor, if $x = \widehat{y}$. Lemma 2 ensures that if $w \equiv AB\widehat{A}\widehat{B}$ then B is saturated. As shown in Figure 5, it suffices now to replace step 6 in Algorithm 1 by:

6a : **If** LCRE$(w, \widehat{w}, p + r + 1, \underline{i + r + n + 1}) = |x|$ **then**
6b : P is a *pseudo-square.*
6c : **End if**

Since LCRE is computed in constant time, the overall algorithm is linear. □

3.2 A linear algorithm for *k*-square-free pseudo-hexagons

Let $w \equiv \mathbf{b}(P)$ be the boundary word of an exact polyomino P. A factor f of a word w is called a *square* if $f = xx$ for some $x \in \Sigma^+$. The set of squares of a word w is Squares(w).

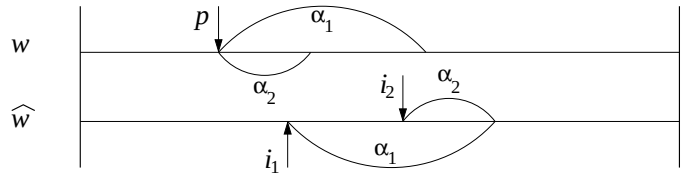
Definition 9. *A word w is* k-square-free $\iff$ $\boldsymbol{max}\{|f| : f \in \text{Squares}(w)\} < k$.

The following technical lemma is useful.

Lemma 10. *Let $w = \mathbf{b}(P) \in \Sigma^{2n}$ a k-square-free word, and let p be any position in w. Then, the number of admissible factors that overlap the position p in w is bounded by $4k + 2\log(n)$.*

Proof. Let $A_1, A_2, \ldots, A_k$ be all the admissible factors that overlaps p in w. Since $w \equiv A_i x \widehat{A_i} y$ with $|x| = |y|$ for all $1 \leq i \leq k$, all their homologue occurrences $\widehat{A_i}$ overlaps the position $p' = p + n$. Thus, there is a position q such that all A_i overlap q in $\widehat{w}$. In Algorithm 1, all the admissible factors overlapping p in w are listed through a loop such that each iteration can detect at most one of them. Let i_1, i_2 be such that $0 \leq i_1 < i_2 < q$ and assume that admissible factors are detected when $i = i_1$ and $i = i_2$.

Let $\alpha_1 = \widehat{w}[i_1, \ldots, q]$ and $\alpha_2 = \widehat{w}[i_2, \ldots, q]$. By definition of *common extension* we also have that $\alpha_1 = w[p, \ldots, p + |\alpha_1| - 1]$ and $\alpha_2 = w[p, \ldots, p + |\alpha_2| - 1]$, as shown below.



This implies that α_2 is prefix and suffix of α_1, so that Lothaire's Proposition 1.3.4 [17] applies. It follows that there are two words $u, v \in \Sigma^*$ such that $\alpha_1 = (uv)^m u$, for some integer m with

$$m(|\alpha_1| - |\alpha_2|) \geq |\alpha_2| \geq (m-1)(|\alpha_1| - |\alpha_2|). \tag{3}$$

If $|\alpha_1| < 2|\alpha_2|$, equation (3) requires m to be greater than 1 and thus α_1 contains a square of length at least $\frac{1}{2}|\alpha_1|$. So, in Algorithm 1, as i goes from 0 to $2n-1$, the number of admissible factors detected is bounded by :

- $\log n$ for i from 0 to $q - 2k$.
- $4k$ for i from $q - 2k + 1$ to $q + 2k - 1$.
- $\log n$ for i from $q + 2k$ to $2n - 1$.

Summing up all these provides the bound. □

Theorem 11. *Let $w = \mathbf{b}(P) \in \Sigma^{2n}$ be a k-square-free word, with $k \in \mathcal{O}(\sqrt{n})$. Determining if w codes a* pseudo-hexagon *is decidable in linear time.*

Proof. The idea is to construct convenient, and not too long, lists of admissible factors and then to use the constant time LCRE function.

Algorithm 2
Input : $w = \mathbf{b}(P) \in \Sigma^{2n}$
Build L_1 : *list of all admissible factors that overlap position p in w;*
$m :=$ *position of the rightmost letter of w included in a factor of L_1;*
Build L_2 : *list of all admissible factors that overlap position $(m+1)$ in w.*
For *all* $A \in L_1$ **do**
  For *all* $B \in L_2$ **do**
    If $w \equiv ABx\widehat{A}\widehat{B}y$ **then**
      Compute i : *position of x in w;*
      Compute j : *position of $\widehat{y}$ in $\widehat{w}$;*
      If $\mathrm{LCRE}(w, \widehat{w}, i, j) = |x|$ **then**
        P is a pseudo-hexagon*;*
      End if

End if
End for
End for

Since w is k-square-free, Lemma 10 ensures that L_1 and L_2 each contain less than $4k+2\log n$ elements. The nested loops perform at most $(4k+2\log n)^2$ iterations, and thus, the overall complexity is $\mathcal{O}(n+(4k+2\log n)^2)=\mathcal{O}(n)$. □

4 Concluding remarks

The results above generalize to more general tilings. Indeed, since the Beauquier-Nivat factorization involves path properties, there is no need for a tile to be a polyomino. For instance, the tile T in Figure 6

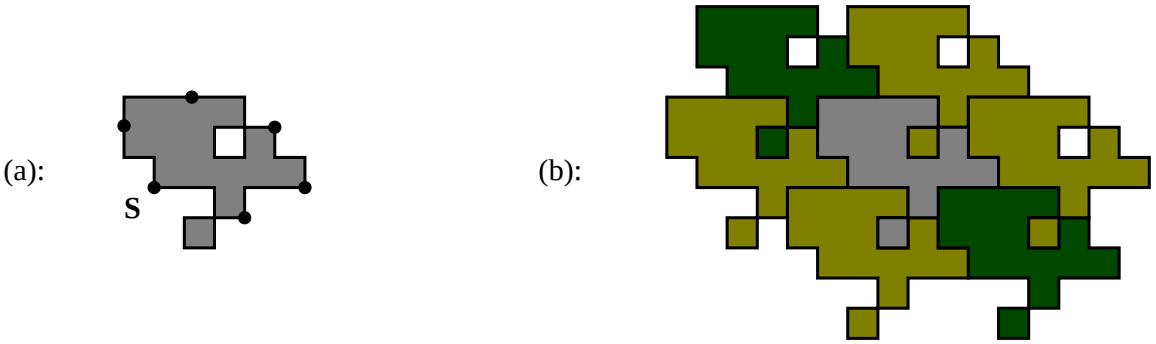


Figure 6. (a) An hexagonal tile with (b) the associated tiling

is hexagonal and its Beauquier-Nivat factorization is (starting from **S**)

$$\begin{aligned}\mathbf{b}(T) &= X\cdot Y\cdot Z\cdot \widehat{X}\cdot \widehat{Y}\cdot \widehat{Z}\\ &= a\,a\,\overline{b}\,\overline{a}\,\overline{b}\,a\,b\,a\cdot b\,a\,a\cdot b\,\overline{a}\,b\cdot \overline{a}\,\overline{b}\,\overline{a}\,b\,a\,b\,\overline{a}\,\overline{a}\cdot \overline{a}\,\overline{a}\,\overline{b}\cdot \overline{b}\,a\,\overline{b}\,.\end{aligned}$$

The contour path is non-crossing, instead of self-avoiding as in the case of polyominoes, and provides an instance of an 8-connected set of cells that tiles the plane by translation. This leads naturally to the problem of characterizing the 8-connected sets of cells that tile the plane by translation. On the other hand there is still a gap to fill. A deeper analysis is needed to lift the condition on the number of square factors in the contour word, in order to provide an optimal algorithm for deciding if a polyomino tiles the plane by translation.

Acknowledgements The authors are grateful to the anonymous referees for their careful reading and valuable comments.

References

[1] D. Beauquier and M. Nivat: *On translating one polyomino to tile the plane.* Discrete Comput. Geom., 6 1991, pp. 575–592.

[2] R. Berger: *The undecidability of the domino problem.* Mem. Amer. Math. Soc., 66 1966.

[3] J.-P. Braquelaire and A. Vialard: *Euclidean paths: A new representation of boundary of discrete regions.* Graphical Models and Image Processing, 61 1999, pp. 16–43.

[4] S. Brlek, G. Labelle, and A. Lacasse: *A note on a result of Daurat and Nivat*, in Proc. DLT 2005, 9-th International Conference on Developments in Language Theory, C. de Felice and A. Restivo, eds., no. 3572 in LNCS, Palermo, Italia, 4–8 July 2005, Springer-Verlag, pp. 189–198.

[5] A. Daurat and M. Nivat: *Salient and reentrant points of discrete sets*, in Proc. IWCIA'03, International Workshop on Combinatorial Image Analysis, A. del Lungo, V. di Gesu, and A. Kuba, eds., Electronic Notes in Discrete Mathematics, Palermo, Italia, 14–16 May 2003, Elsevier Science.

[6] H. Freeman: *On the encoding of arbitrary geometric configurations.* IRE Trans. Electronic Computer, 10 1961, pp. 260–268.

[7] H. Freeman: *Boundary encoding and processing*, in Picture Processing and Psychopictorics, B. Lipkin and A. Rosenfeld, eds., Academic Press, New York, 1970, pp. 241–266.

[8] I. Gambini and L. Vuillon: *An algorithm for deciding if a polyomino tiles the plane by translations*, tech. rep., LAMA, 2003.

[9] M. Gardner: *Mathematical games.* Scientific American, 1958, Sept. pp. 182–192, Nov. pp. 136–142.

[10] S. W. Golomb: *Checker boards and polyominoes.* Amer. Math. Monthly, 61 1954, pp. 675–682.

[11] S. W. Golomb: *Polyominoes: Puzzles, Patterns, Problems, and Packings*, Princeton Academic Press, 1996.

[12] Y. Gurevich and I. Koriakov: *A remark on Berger's paper on the domino problem.* Siberian Journal of Mathematics, 13 1972, pp. 459–463, (in Russian).

[13] D. Gusfield: *Algorithms on Strings, Trees and Sequences*, Cambridge University Press, Cambridge (UK), 1997.

[14] D. Gusfield and J. Stoye: *Linear time algorithms for finding and representing all the tandem repeats in a string.* Journal of Computer and System Sciences, 69 2004, pp. 525–546.

[15] L. Ilie: *A note on the number of distinct squares in a word*, in Proc. Words2005, 5-th International Conference on Words, S. Brlek and C. Reutenauer, eds., vol. 36, Montreal, Canada, 13–17 Sept. 2005, Publications du LaCIM, pp. 289–294.

[16] M. Lothaire: *Applied Combinatorics on Words*, Cambridge University Press, Cambridge (UK), 2005.

[17] M. Lothaire: *Combinatorics on Words*, Cambridge University Press, Cambridge (UK), 2005.

[18] E. Weisstein: *Polyomino, from Wolfram Mathworld.* Available electronically at http://mathworld.wolfram.com/Polyomino.html, 2006.

[19] H. A. G. Wijshoff and J. van Leeuven: *Arbitrary versus periodic storage schemes and tesselations of the plane using one type of polyomino.* Inform. Control, 62 1984, pp. 1–25.