On the Arithmetics of Discrete Figures

Alexandre Blondin Massé[⋆,1,2], Amadou Makhtar Tall[1], and Hugo Tremblay[1,2]

[1] Laboratoire d'informatique formelle, Université du Québec à Chicoutimi, 555, boul. de l'Université, Chicoutimi, G7H 2B1, Canada
[2] Laboratoire de combinatoire et d'informatique mathématique, Université du Québec à Montréal, CP 8888, Succ. Centre-ville, Montral (Qubec) H3C 3P8
ablondin@uqac.ca,amadou-makhtar.tall1@uqac.ca
hugo.tremblay@lacim.ca
http://lif.uqac.ca

Abstract. Discrete figures (or polyominoes) are fundamental objects in combinatorics and discrete geometry, having been studied in many contexts, ranging from game theory to tiling problems. In 2008, Provençal introduced the concept of prime and composed polyominoes, which arises naturally from a *composition* operator acting on these discrete figures. Our goal is to study further polyomino composition and, in particular, factorization of polyominoes as a product of prime ones. We provide a polynomial time (with respect to the perimeter of the polyomino) algorithm that allows one to compute such a factorization. As a consequence, primality of polyominoes can be decided in polynomial time.

Keywords: Discrete figures, polyominoes, boundary words, primality, tiling, morphism.

1 Introduction

Although polyominoes are known since antiquity, it is only in 1953 that the word was coined by S.W. Golomb and was later popularized by M. Gardner, who was very active in recreational mathematics for a large part of the 20th century [7]. In fact, polyominoes are well-known from the general public: One only needs to think about the very popular Tetris video game, which consists in filling lines with tetrominoes (i.e. polyominoes composed of four unit cells). Polyominoes are also well-known and useful in combinatorics and theoretical computer science. For instance, one of the applications of the famous dancing links algorithm proposed by D. Knuth in 2000 consists in solving polyomino tiling puzzles efficiently [8] (an implementation of such a solver is found in [9]).

Polyominoes have also been fundamental objects in the study of tilings. It is known since 1970 that the problem of tiling the plane with free polyominoes (polyominoes that can be rotated and reflected) picked from a finite set is undecidable [6]. When restricted to only one tile, it is not known if it is decidable or

⋆ Corresponding author. This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

(a)

(b)

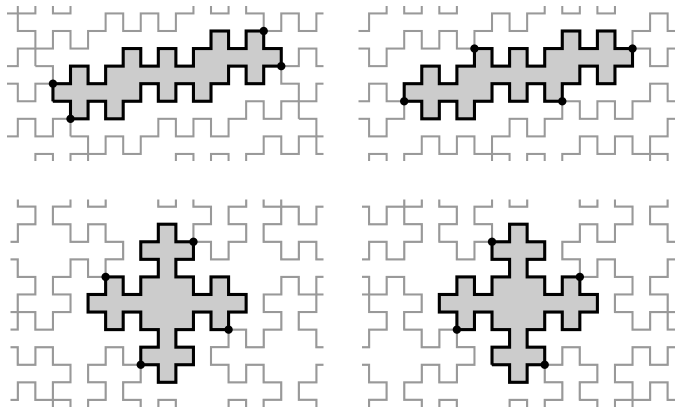

Fig. 1. Two double parallelogram tiles and the tilings they induce. The black dots indicate the points shared by four copies of the tile. (a) A Christoffel tile. (b) A Fibonacci tile (see [2] for more details).

not. Also, the problem of deciding if a given polyomino tiles another polyomino is known to be NP-complete [11]. In the case where no rotation and no reflection is allowed, the problem becomes much simpler. Indeed, not only is it decidable, but it can also be determined in polynomial time (a $\mathcal{O}(n^2)$ bound is proved in [5], which was reduced to $\mathcal{O}(n)$ for tilings whose boundary has bounded local periodicity [4]) by using results from [1,14]. The basic idea comes from D. Beauquier and M. Nivat, who characterized such objects by the shape of their boundary: Indeed, they are polyominoes whose boundary can be divided into four or six pieces that are pairwise parallel [1]. Pseudo-square tiles have been extensively studied (see for instance [**?**] and [**?**]).

In [3], Blondin Massé et al. considered the problem of enumerating polyominoes called *double parallelogram tiles*, i.e. polyominoes yielding two distinct parallelogram tilings (see Figure 1). In order to prove one of their main results, they had to rely on the concept of *prime* and *composed* polyominoes, introduced by Provençal in his Ph.D thesis [13]. However, very little is known about that classification.

This article is divided as follows. In Section 2, we introduce the usual definitions and notation. Section 3 is devoted to the link between discrete paths and words. Composition, prime and composed polyominoes are defined in Section 4. We provide the main algorithms of this article in Section 5 and we briefly conclude in Section 6.

2 Definitions and Notation

The *square grid* is the set $\mathbb{Z}^2$. A *cell* is a unit square in $\mathbb{R}^2$ whose corners are points of $\mathbb{Z}^2$. We shall denote by $c(i, j)$ the cell

$$c(i, j) = \{(x, y) \in \mathbb{R}^2 \mid i \leq x \leq i + 1, j \leq y \leq j + 1\},$$

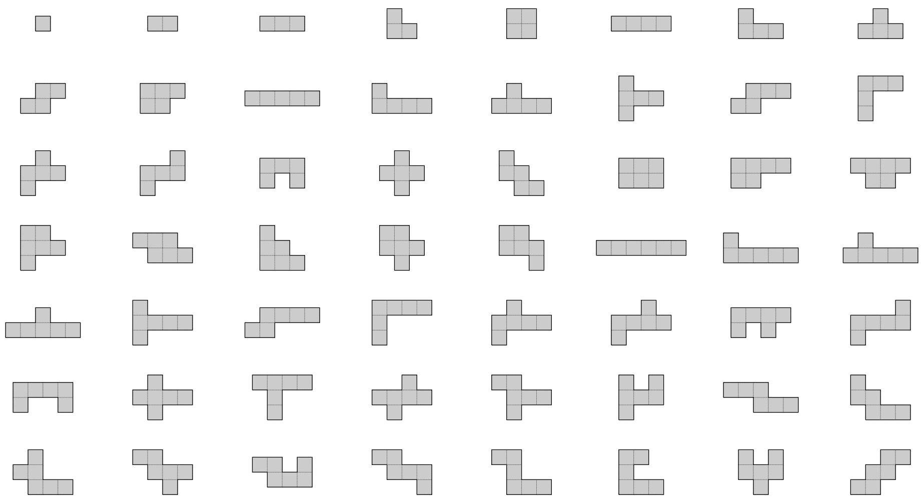
Fig. 2. Free polyominoes without hole having area at most 6.

i.e. (i, j) is the bottom left corner of the cell. The set of all cells is denoted by $\mathcal{C}$. Two cells $c, d \in \mathcal{C}$ are called 4-*neighbors* if they have exactly one side in common. Clearly, every cell $c(i, j)$ has exactly four 4-neighbors: $c(i+1, j)$, $c(i, j+1)$, $c(i-1, j)$ and $c(i, j-1)$. A 4-*connected region of* $\mathbb{Z}^2$ is any subset R of $\mathbb{R}^2$ such that for every $c, d \in R$, there exist cells $c_1, c_2, \ldots, c_k$ such that $c = c_1$, $d = c_k$, and c_i, c_{i+1} are 4-neighbors for $i = 1, 2, \ldots, k-1$.

The usual isometries (translation, rotation and reflections) are naturally defined on 4-connected regions. In some cases, it is convenient to consider two regions equivalent up to some isometries. For instance, the relation $R \sim_\theta S$ defined by "R is a translated copy of S" is an equivalence relation: A *fixed polyomino* (or simply *polyomino*) is any equivalence class of $\sim_\theta$. In the same spirit, if the relation $R \sim_\rho S$ is defined by "R is a translated or rotated copy of S", then a *one-sided polyomino* is any equivalence class of $\sim_\rho$. Finally, a *free polyomino* is any equivalence class of the relation $R \sim_\sigma S$ defined by "R is a translated, rotated, reflected or glided reflected copy of S".

Another notion of interest is that of *holes*. Given a region R, let $\overline{R}$ denote its *complement*, i.e. $\overline{R} = \mathcal{C} - R$. We say that R is *without hole* if $\overline{R}$ is a 4-connected region. All free polyominoes of c cells, $c = 1, 2, 3, 4, 5, 6$, are illustrated in Figure 2.

In many situations, it is convenient to represent a polyomino by its boundary, which in turn is easily represented by a word on a 4-letter alphabet encoding the elementary steps $\rightarrow$ (east), $\uparrow$ (north), $\leftarrow$ (west) and $\downarrow$ (south). In the following, we recall basic definitions from combinatorics on words [10].

An *alphabet* $\mathcal{A}$ is a finite set whose elements are *letters*. A finite word w is a function $w : \{1, 2, \ldots, n\} \rightarrow \mathcal{A}$, where w_i (also denoted by $w[i]$) is the i-th letter, $1 \leq i \leq n$. The *length* of w, denoted by $|w|$, is the integer n. The *empty word* ε is the unique word having length 0.

The *free monoid* $\mathcal{A}^*$ is the set of all finite words over $\mathcal{A}$. The *reversal* of $w = w_1w_2\cdots w_n$ is the word $\widetilde{w} = w_nw_{n-1}\cdots w_1$. Given a nonempty word w, let $\textsc{Fst}(w) = w_1$ and $\textsc{Lst}(w) = w_n$ denote respectively the first and last letter of the word w. A word u is a *factor* of another word w if there exist $x, y \in \mathcal{A}^*$ such that $w = xuy$. A *proper factor* of w is any factor u such that $u \neq \varepsilon$ and $u \neq w$. The factor u of w starting at position i and ending with position j is denoted by $w[i:j]$ and the integer i is called an *occurrence of* u *in* w. Moreover, if $x = \varepsilon$, then u is called *prefix* and if $y = \varepsilon$, it is called a *suffix* of w. We denote by $|w|_u$ the number of occurrences of u in w. Two words u and v are *conjugate*, written $u \equiv v$ or sometimes $u \equiv_{|x|} v$, when x, y are such that $u = xy$ and $v = yx$. Conjugacy is an equivalence relation and the class of a word w is denoted by $[w]$. A *power* of a word u is a word of the form u^k for some integer $k \in \mathbb{N}$. It is convenient to set $u^0 = \varepsilon$ for each word u.

Given two alphabets $\mathcal{A}$ and $\mathcal{B}$, a *morphism* is a function $\varphi : \mathcal{A}^* \to \mathcal{B}^*$ compatible with concatenation, that is, $\varphi(uv) = \varphi(u)\varphi(v)$ for any $u, v \in \mathcal{A}^*$. It is clear that a morphism is completely defined by its action on the letters of A. In the same spirit, an *antimorphism* is a function $\varphi : \mathcal{A}^* \to \mathcal{B}^*$ such that $\varphi(uv) = \varphi(v)\varphi(u)$ whenever $u, v \in \mathcal{A}^*$. The reversal $\widetilde{\cdot}$ is an antimorphism and it is not difficult to show that for any antimorphism φ, we have $\varphi = \widetilde{\cdot} \circ \varphi'$, where φ' is a morphism, i.e. φ can be expressed as the composition of a morphism and the reversal operator.

3 Paths as Words

The Freeman chain code $\mathcal{F} = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$ is considered as the additive group of integers modulo 4. To distinguish the number (for instance 0) from the letter (for instance $\mathbf{0}$), we shall denote the latter with bold font. Basic transformations on $\mathcal{F}$ are rotations $\rho^i : x \mapsto x + i$ and reflections $\sigma_i : x \mapsto i - x$, which extend uniquely to morphisms on $\mathcal{F}^*$. Two other useful functions for our purpose are the morphism $\bar{\cdot}$ defined by $\mathbf{0} \leftrightarrow \mathbf{2}$ and $\mathbf{1} \leftrightarrow \mathbf{3}$ and the antimorphism $\widehat{\cdot} = \bar{\cdot} \circ \widetilde{\cdot}$.

From now on, words are considered over $\mathcal{F}$ and are called *paths* to emphasize their geometrical nature. A path w is *closed* if it satisfies $|w|_{\mathbf{0}} = |w|_{\mathbf{2}}$ and $|w|_{\mathbf{1}} = |w|_{\mathbf{3}}$, and it is *simple* if no proper factor of w is closed. A *boundary word* is a simple and closed path. It is convenient to represent each closed path w by its conjugacy class $[w]$, also called *circular word*. It follows from this definition that counter-clockwise circular boundary words and the polyominoes without hole they describe are in bijection.

In [1], D. Beauquier and M. Nivat proved that a polyomino P tiles the plane by translation if and only if it admits a boundary word

$$w = xyz\widehat{x}\widehat{y}\widehat{z},$$

with at most one empty word among x, y and z. If x, y or z is empty, then P is called *parallelogram polyomino* (*pseudo-square* in [4]). Otherwise, it is called *hexagon polyomino* (*pseudo-hexagon* in [4]). It should be noted that parallelogram polyominoes were introduced more than 40 years ago with a completely different meaning [12], but we still prefer to use the word “parallelogram” in this case as it seems the most appropriate word for the concept (see Figure 3).

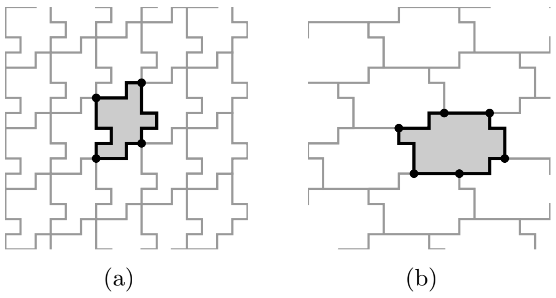

Fig. 3. (a) A parallelogram polyomino and the tiling it induces (one of its boundary words admits a BN-factorization $\mathbf{0010}\cdot\mathbf{101211}\cdot\mathbf{2322}\cdot\mathbf{330323}$). The black dots indicate the factorization points of the boundary word and they clearly form a parallelogram. (b) An hexagon polyomino having boundary word $\mathbf{000}\cdot\mathbf{0010}\cdot\mathbf{1121}\cdot\mathbf{222}\cdot\mathbf{2322}\cdot\mathbf{3033}$ and the corresponding tiling.

The following simple fact, proven in [3], is one of the key idea for designing Algorithm 1. Roughly speaking, it states that the factors of any parallelogram polyomino always start and end with the same elementary step, and that all four letters are exactly covered once:

Proposition 1 ([3]). *Let* $w \equiv xy\widehat{x}\widehat{y}$ *be a boundary word of a parallelogram polyomino. Then* $\textsc{Fst}(x) = \textsc{Lst}(x)$*,* $\textsc{Fst}(y) = \textsc{Lst}(y)$ *and the first letter of* x*,* $\widehat{x}$*,* y*,* $\widehat{y}$ *are mutually distinct, that is,*

$$\{\textsc{Fst}(x), \textsc{Fst}(\widehat{x}), \textsc{Fst}(y), \textsc{Fst}(\widehat{y})\} = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}.$$

4 Prime and Composed Polyominoes

On the Freeman alphabet, a class of morphisms is of particular interest for our purpose:

Definition 1 ([3,13]). *A morphism* $\varphi : \mathcal{F}^* \to \mathcal{F}^*$ *is called* homologous *if* $\varphi(a) = \widehat{\varphi(\overline{a})}$ *for every* $a \in \mathcal{F}$*.*

Roughly speaking, homologous morphisms replace the two horizontal elementary steps by an arbitrary path ($\varphi(\mathbf{0})$ is traveled in the opposite direction with respect to $\varphi(\mathbf{2})$) and the same idea applies to vertical steps. One proves easily that homologous morphisms satisfy $\widehat{\varphi(w)} = \varphi(\widehat{w})$ for any $w \in \mathcal{F}^*$.

As polyominoes without hole have simple boundary word, an additional condition on homologous morphism is necessary in order to define prime and composed polyominoes without hole.

Definition 2. *Let φ be an homologous morphism. We say that φ is a* parallelogram morphism *if*

(i) $\varphi(\mathbf{0123})$ *is the boundary word of a parallelogram polyomino;*
(ii) $\textsc{Fst}(\varphi(a)) = a$ *for every* $a \in \mathcal{F}$.

The purpose of condition (ii) of Definition 2 is justified by the following extension of Proposition 1:

Proposition 2. *Let φ be a parallelogram morphism. Then for every $a \in \mathcal{F}$, $\textsc{Fst}(\varphi(a)) = a = \textsc{Lst}(\varphi(a))$.*

Let $\mathcal{M}$ be the set of morphisms on $\mathcal{F}$, $\mathcal{H}$ the set of homologous morphisms and $\mathcal{P}$ the set of parallelogram morphisms.

Proposition 3. *$\mathcal{H}$ is a submonoid of $\mathcal{M}$ and $\mathcal{P}$ is a submonoid of $\mathcal{H}$ with respect to the concatenation.*

Proof. The identity morphism Id is both an homologous and parallelogram morphism. Now, let $\varphi, \varphi' \in \mathcal{H}$. Then for every letter $a \in \mathcal{F}$,

$$(\widehat{\varphi \circ \varphi'})(\overline{a}) = \varphi(\widehat{\varphi'(\overline{a})}) = \varphi(\widehat{\varphi'(\overline{a})}) = \varphi(\varphi'(a)) = (\varphi \circ \varphi')(a),$$

so that $\varphi \circ \varphi' \in \mathcal{H}$ as well. Similarly, on one hand, if $\varphi, \varphi' \in \mathcal{P}$, then

$$\textsc{Fst}((\varphi \circ \varphi')(a)) = \textsc{Fst}((\varphi(\varphi'(a))) = \textsc{Fst}(\varphi'(a)) = a.$$

On the other hand $(\varphi \circ \varphi')(\mathbf{0123})$ is closed. The proof that $(\varphi \circ \varphi')(\mathbf{0123})$ is simple is more technical and is ommitted due to lack of space. □

Every parallelogram morphism φ is associated with a unique polyomino, denoted by $\textsc{Poly}(\varphi)$. Conversely, one might be tempted to state that each parallelogram polyomino is uniquely represented by a parallelogram morphism. Indeed, condition (i) in Definition 2 ensures that the boundary word is traveled counterclockwise and that only one of its conjugate is chosen. However, even when taking those restrictions into account, there exist parallelogram polyominoes with two distinct parallelogram factorizations:

Example 1. The parallelogram morphisms φ defined by $\varphi(0) = 010, \varphi(1) = 121$ and φ' defined by $\varphi'(0) = 030, \varphi'(1) = 101$ both yield the X pentamino (see the polyomino in the third row, fourth column of Fig. 2).

In fact, every double parallelogram tile admits two nontrivial distinct factorizations [3].

We are now ready to define prime and composed polyominoes. It shall be noted that prime polyominoes were defined in [13], but the (equivalent) definition below relies on parallelogram morphisms:

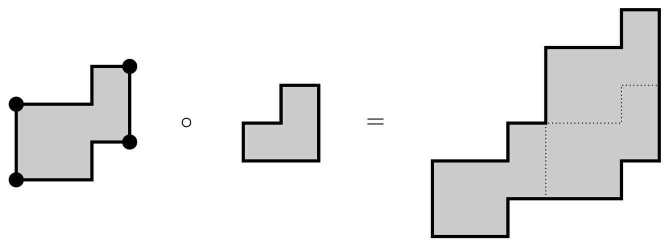

Fig. 4. Composition of a polyomino with a parallelogram polyomino. The parallelogram factorization is illustrated with black dots. The resulting composed polyomino can be tiled by copies of the parallelogram polyomino.

Definition 3 ([13]). *A polyomino P distinct from the one cell polyomino is called* composed *if there exists some boundary word u and some parallelogram morphism φ such that*

(i) $\textsc{Poly}(\varphi(u)) = P$*;*
(ii) $\textsc{Poly}(u)$ *is not the unit square;*
(iii) $\varphi \neq \mathrm{Id}$.

Otherwise, P is called prime*.*

In other words, a polyomino is prime if and only if it cannot be decomposed as a product of some parallelogram polyomino and some other polyomino. It is worth mentioning that the problem of deciding if a polyomino is prime is as least as hard as the problem of deciding if a number is prime. Indeed, one notices that the rectangle $1 \times n$ having boundary word $\mathbf{0}^n\mathbf{1}\mathbf{2}^n\mathbf{3}$ is prime if and only if n is prime. In the same spirit, a boundary word u is called *prime* if $\textsc{Poly}(u)$ is a prime polyomino and a parallelogram morphism φ is called *prime* if $\textsc{Poly}(\varphi(\mathbf{0123}))$ is a prime polyomino.

Following Definition 3, one is naturally led to ask whether the fundamental theorem of arithmetic can be extended to polyominoes. There are two conditions to verify: the existence of a prime factorization and the unicity of this factorization. The former is easy to prove:

Theorem 1. *Let P be any non-unit polyomino without hole. Then either P is a prime polyomino or there exist prime parallelogram morphisms $\varphi_1, \varphi_2, \ldots, \varphi_n$ and a prime boundary word u such that $P = \textsc{Poly}((\varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n)(u))$.*

Proof. By induction on the perimeter of P. If P is prime, then there is nothing to prove. Otherwise, there exist a parallelogram morphism $\varphi \neq \mathrm{Id}$ and a boundary word u, with $\textsc{Poly}(u)$ different for the unit square, such that $P = \textsc{Poly}(\varphi(u))$. By induction applied to $\textsc{Poly}(u)$, there exist parallelogram morphisms φ_1, φ_2, $\ldots$, φ_n and a prime boundary word v such that

$$u = (\varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n)(v),$$

so that $P = \textsc{Poly}((\varphi \circ \varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n)(v))$. If φ is prime, then the claim is proved. Otherwise, using induction, one shows that $\varphi = \varphi'_1 \circ \varphi'_2 \circ \ldots \circ \varphi'_k$ for some prime parallelogram morphisms φ'_1, φ'_1, …, φ'_k, which concludes the proof. □

The proof of unicity seems rather involved and is discussed in the last section (see Conjecture 1).

5 Algorithms

In this section, we shift our attention to the algorithmic aspects surrounding composed and prime polyominoes. Basically, to determine if a polyomino P is composed, we need to find a parallelogram morphism $\varphi \neq \mathrm{Id}$ and a boundary word $u \notin [\mathbf{0123}]$ such that $\varphi(u)$ is a boundary word of P. If no such morphism exists, then we may conclude that the polyomino is prime.

5.1 Naive version

The simplest straightforward approach consists of trying every possible factorization until either one is found or all have been exhausted. To reduce the number of cases to be considered, by Proposition 2, we may restrict ourselves to factors starting and ending with the same letter. More precisely, let P be any polyomino and w one of its boundary word. For every $a \in \mathcal{F}$, let F_a be the set of factors of w^2 starting and ending with a. The following steps can be used to factorize P:

1. Compute $F_\mathbf{0}$ and $F_\mathbf{1}$;
2. Let $u \in F_\mathbf{0}$, $v \in F_\mathbf{1}$;
3. Let φ be the parallelogram morphism induced by u and v;
4. If there is some conjugate w' of w such that $w = \varphi(w')$, then return (φ, u).
5. Otherwise, repeat steps 2–4 until all possible u and v have been exhausted.

The complexity of the previous algorithm is clearly polynomial.

Theorem 2. *Any polyomino P may be factorized as a product of prime polyominoes in $\mathcal{O}(n^6)$, where n is the perimeter of P.*

Proof. Let w be any boundary word of P. Step 1 is done in $\mathcal{O}(n^2)$, as there are $\mathcal{O}(n^2)$ factors starting and ending with some letter in any word on $\mathcal{F}$. Steps 2–4 are then repeated $\mathcal{O}(n^4)$. The construction of φ in Step 3 is done in constant time, while Step 4 takes $\mathcal{O}(n)$ since it must be performed for every conjugate of w. Therefore, decomposing P as $\varphi(u)$, for some parallelogram morphism φ and some boundary word u is done in $\mathcal{O}(n^5)$. Since it must be repeated as long as either φ or u is not prime, the overall complexity is $\mathcal{O}(n^6)$. □

As a consequence:

Corollary 1. *Given a polyomino P having perimeter n, it can be decided in polynomial time with respect to n whether P is prime or composed.* □

The algorithm was implemented in Python and tested for polyominoes having number of cells between 1 and 10. Figure 5 contains the result. As there are many more prime than composed polyominoes, only the composed ones are illustrated.

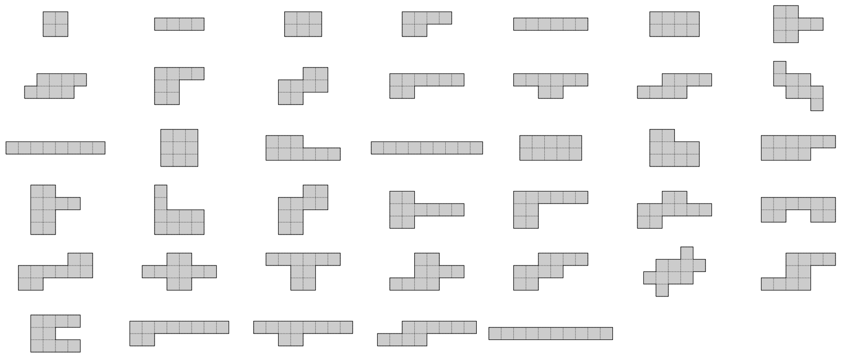
Fig. 5. Free polyominoes having area at most 10 that are composed.

5.2 Improving the naive algorithm

An upper bound of $\mathcal{O}(n^6)$ is rather crude and one should expect to reduce it. More precisely, instead of enumerating all possible factors u in $F_{\mathbf{0}}$ and v in $F_{\mathbf{1}}$, the choice should be sensible about whether u and v occur contiguously or with overlap. This yields an algorithm such as the following one. As a first step, choose any boundary word w starting with $\mathbf{0}$. The first idea consists in trying to divide w into blocks starting and ending with the same letter, in virtue of Proposition 2. It suffices to look at every occurrence of $\mathbf{0}$ for the ending of $\varphi(\mathbf{0})$. When such a block is chosen, then $\varphi(\mathbf{0})$ and $\varphi(\mathbf{2})$ are completely determined.

The next step consists in checking the letter following the first block. If it is $\mathbf{0}$ or $\mathbf{2}$, then we check if the following letters match $\varphi(\mathbf{0})$. If it is not the case, then we have to try with another choice for $\varphi(\mathbf{0})$. On the other hand, if there is a match, then we go on to the next block. We repeat the previous steps until we reach either the letter $\mathbf{1}$ or $\mathbf{3}$. In the same manner as for the letter $\mathbf{0}$, we then try every possible block for either $\mathbf{1}$ or $\mathbf{3}$ (and then the image of the last letter under φ is uniquely determined). When the four images of φ over single letters are chosen, it only remains to verify if the boundary word may be factorized as a product of the four blocks (this step is called the *decoding step*).

Based on the previous paragraphs, one might design a branch-and-bound algorithm for factorizing any polyomino. The pseudocode is found in Algorithm 1 and it was also implemented in Python.

Theorem 3. *Any polyomino P may be factorized as a product of prime polyominoes in $\mathcal{O}(n^5)$, where n is the perimeter of P.*

Proof. Let w be any boundary word of P starting with $\mathbf{0}$. Choosing each occurrence of $\mathbf{0}$ in w to construct $\varphi(\mathbf{0})$, there are at most n possible values (and then $\varphi(\mathbf{2})$ is determined). Once $\varphi(\mathbf{0})$ is chosen, there are at most n possible values for $\varphi(\mathbf{1})$ (and then $\varphi(\mathbf{3})$ is determined). Finally, when $\varphi(a)$ is known for each $a \in \mathcal{F}$, it remains to verify if w might be decoded from φ, which is done in linear time at most. Therefore, it can be decided in $\mathcal{O}(n^3)$ whether w is decodable according

Algorithm 1: Improved factorization of polyomino

```
Require: $w$ is simple;
Function Factorize($w$ : boundary word)
begin
    for $i \in$ Occurrences($\mathbf{0}, w$) do
        /* We try to factorize every conjugate starting with $\mathbf{0}$ */
        $u \leftarrow$ Conjugate($w, i$)
        $\varphi \leftarrow$ FactorizeRec($w, \emptyset, 0$)
        if $\varphi \neq \emptyset$ then
            return $\varphi$
    return $\emptyset$

Function FactorizeRec($w$ : boundary word, $\varphi$ : morphism, $i$ : integer)
begin
    if $i \geq |w|$ then
        /* We first check if the decoding is complete */
        if $\varphi$ is completely defined and non-trivial then
            return $\varphi$
        else
            return $\emptyset$
    else
        $\ell \leftarrow w[i]$
        if $\varphi(\ell)$ is defined then
            /* The next block should match $\varphi(\ell)$ */
            $k \leftarrow |\varphi(\ell)|$
            if $k > |w| - i$ or $w[i : i + k]$ then
                return $\emptyset$
            else
                return FactorizeRec($w[i + k : |w| - 1], \varphi, i + k$)
        else
            /* We consider constructions of the next block */
            for $j \in$ Occurrences($\ell, w[i : |w| - 1]$) do
                $\varphi(\ell) \leftarrow w[i : j]$
                $\varphi(\overline{\ell}) \leftarrow \widehat{w[i : j]}$
                $\varphi \leftarrow$ FactorizeRec($w[j + 1 : |w| - 1], \varphi, j + 1$)
                if $\varphi$ is not trivial then
                    return $\varphi$
                else
                    return $\emptyset$
        return $\emptyset$
```

to some parallelogram morphism φ. Since the test must be performed for every conjugate of w starting with $\mathbf{0}$, we obtain a bound of $\mathcal{O}(n^4)$. Finally, repeating this reduction until a prime decomposition is obtained yields the claimed $\mathcal{O}(n^5)$ complexity. $\square$

5.3 Improving the upper bound

The $\mathcal{O}(n^5)$ bound seems rather high and it is not clear whether it can be realized. To support this impression, let us study Algorithm 1 on the polyomino P having boundary word

$$w = \mathbf{0}^k\mathbf{1}^k\mathbf{2}^{k-1}\mathbf{323}^{k-1}.$$

Let $n = |w| = 4k$. Since P is a square $k \times k$ minus one corner cell, one might prove that it is prime. For the algorithm to take as much time as possible, assume that k has d divisors. Then the algorithm will try the k conjugate of w starting with 0 and will construct d images for $\varphi(\mathbf{0})$. Similarly, we will have to consider d possible images for $\varphi(\mathbf{1})$. The decoding being performed in linear time, we get an overall bound of $\mathcal{O}(kd^2n) = \mathcal{O}(n^2d^2)$. But d is in general much smaller than k: It is easy to see for instance that $d \leq \sqrt{k}$ (tighter bounds from number theory can be derived). Therefore, we obtain in that case a bound of $\mathcal{O}(n^3)$.

We believe that a significant improvement could reduce the theoretical bound to $\mathcal{O}(n^4)$ or even $\mathcal{O}(n^3)$ by taking into account the *repetitions* of some factors. For instance, when we try to factorize the polyomino P of the previous paragraph with $\varphi(\mathbf{0}) = \mathbf{0}$ and we read the factor $\mathbf{0}^k$, it would be more efficient to keep in memory the fact that $\varphi(\mathbf{0})$ can be set only to powers of $\mathbf{0}$ that divide k.

6 Concluding Remarks

In this paper, we have provided an algorithm to express any polyomino as a product of prime polyominoes in polynomial time. As a consequence, it follows that we can decide if a polyomino is prime or composed in polynomial time as well. Another result worth mentioning is Theorem 1, which guarantees the existence of a prime factorization. However, it seems more difficult to verify if such a factorization is unique. Hence, we are led to propose the following conjecture:

Conjecture 1. Let P be some composed polyomino. Then there exist unique prime parallelogram morphisms φ_1, φ_2, ..., φ_n and a unique prime boundary word u such that $P = \textsc{Poly}((\varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n)(u))$.

Indeed, actual computational explorations have not succeeded in providing a counter-example to this conjecture for polyominoes having area at most 10.

As mentioned above, the $\mathcal{O}(n^5)$ bound is rather crude and it would not be surprising to design more efficient algorithms for solving the factorization problem.

The reader interested in toying with an implementation of Algorithm 1 is invited to look at the publicly available source code hosted on Github[3], which only depends on a basic Python installation to be run.

References

1. D. Beauquier and M. Nivat. On translating one polyomino to tile the plane. *Discrete Comput. Geom.*, 6:575–592, 1991.
2. A. Blondin Massé, S. Brlek, A. Garon, and S. Labbé. Christoffel and Fibonacci tiles. In S. Brlek, X. Provençal, and C. Reutenauer, editors, *DGCI 2009, 15th IAPR Int. Conf. on Discrete Geometry for Computer Imagery*, Springer-Verlag LNCS 5810, 67–78, 2009.
3. A. Blondin Massé, A. Garon, and S. Labbé. Combinatorial properties of double square tiles. *Theoretical Computer Science*, 2012. Available at `http://www.sciencedirect.com/science/article/pii/S0304397512009723`.
4. S. Brlek and X. Provençal. On the problem of deciding if a polyomino tiles the plane by translation. In Jan Holub and Jan Žďárek, editors, *Proceedings of the Prague Stringology Conference '06*, ISBN80-01-03533-6, pages 65–76, Prague, Czech Republic, 28–30 August 2006. Czech Technical University in Prague.
5. I. Gambini and L. Vuillon. An algorithm for deciding if a polyomino tiles the plane by translations. *Theoret. Informatics Appl.*, 41:147–155, 2007.
6. S. W. Golomb. Tiling with sets of polyominoes. *Journal of Combinatorial Theory*, 9(1):60 – 71, 1970.
7. S. W. Golomb. *Polyominoes: Puzzles, Patterns, Problems, and Packings.* Princeton Academic Press, Princeton, 1996.
8. D. E. Knuth. Dancing links, 2000. `http://arxiv.org/abs/cs/0011047`.
9. S. Labbé. Tiling solver in Sage, 2011. `http://www.sagemath.org/doc/reference/combinat/sage/combinat/tiling.html`.
10. M. Lothaire. *Combinatorics on Words.* Cambridge University Press, Cambridge, 1997.
11. C. Moore and J. Michael. Hard tiling problems with simple tiles, 2000. `http://arxiv.org/abs/math/0003039`.
12. G. Polyá. On the number of certain lattice polygons. *Journal of Combinatorial Theory*, 6(1):102 – 105, 1969.
13. X. Provençal. *Combinatoire des mots, géométrie discrète et pavages.* PhD thesis, D1715, Université du Québec à Montréal, 2008.
14. 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*, 62:1–25, 1984.

[3] `http://github.com/ablondin/prime-polyominoes`