# From Linear Partitions to Parallelogram Polyominoes

Roberto Mantaci[1] and Paolo Massazza[2,⋆]

[1] LIAFA, CNRS UMR 7089, Université Paris Diderot - Paris 7,
Case 7014, 75205 Paris Cedex 13, France
mantaci@liafa.jussieu.fr
[2] Università degli Studi dell'Insubria, Dipartimento di Informatica e Comunicazione,
Via Mazzini 5, 21100 Varese, Italy
paolo.massazza@uninsubria.it

**Abstract.** We provide a bijection between parallelogram polyominoes and suitable pairs of linear partitions. This lets us design a CAT (Constant Amortized Time) algorithm for generating all parallelogram polyominoes of size $n$ using $O(\sqrt{n})$ space.

**Keywords:** Polyominoes, Exhaustive generation, CAT algorithms.

## 1 Introduction

As combinatorial objects, polyominoes are simply defined as finite connected sets of edge-to-edge adjacent square unit cells in the cartesian two-dimensional plane. Polyominoes and their applications appear more and more often in several contexts, with interesting overlaps and interactions betweens them.

In enumerative combinatorics, where several families of polyominoes have been characterized and enumerated according to several characteristics (area, perimeter, ...), providing in some cases their generating functions [4,8];

In bijective combinatorics; a polyomino $P$ can be described by a pair of appropriate paths in the $\mathbb{N} \times \mathbb{N}$ lattice. This consideration relates polyominoes to lattice paths theory [1,10,17];

In two-dimensional language theory and image treatment, where a polyomino is considered as a two-dimensional word and appropriate families of polyominoes turn out to be two-dimensional languages with specific properties [5,6], or where the problem of its reconstruction from partial informations is considered [2,12];

In tiling theory, where tile shapes can be described by polyominoes [11,16].

Here we are interested in particular polyominoes called *Parallelogram Polyominoes*. This class consists of all polyominoes bounded by two paths in the $\mathbb{N}\times\mathbb{N}$ lattice composed of north and east steps and intersecting each other only in the initial and the final points. This class has first been studied by Polya [18], who counted the number of parallelogram polyominoes with respect to the perimeter,

⋆ Partially supported by Project M.I.U.R. PRIN 2007–2009: Mathematical aspects and forthcoming applications of automata and formal languages.

G. Mauri and A. Leporati (Eds.): DLT 2011, LNCS 6795, pp. 350–361, 2011.

and provided a (non closed) formula for the generating function by perimeter and area. A nice bijection between parallelogram polyominoes of perimeter $2n+2$ and Dick words of length $2n$ was given in [8] and used in [7] to produce an equation for the generating function according to the area, the width and the number of left path corners. The relation between parallelogram polyominoes and two-dimensional languages has also been studied. In particular, in [6] it is shown that parallelogram polyominoes can be represented by two-dimensional words of a tiling recognizable language. This class of polyominoes has also been investigated from the exhaustive generation point of view. More precisely, in [3] the ECO method has been applied to the recursive generation of parallelogram polyominoes with semi-perimeter $n$. This method has also been used later to generate and enumerate some classes of convex polyominoes, see [1,9].

In this paper, we highlight a natural bijection between parallelogram polyominoes having size (or area) equal to $n$ and pairs $(s,t)$ of integer partitions, where $t$ is a partition of an integer $m$ and $s$ a partition of the integer $m+n$.

Using this bijection, we provide an algorithm for the exhaustive generation of all parallelogram polyominoes of a given area. This algorithm runs in constant amortized time (CAT), that is, if $N$ is the number of all parallelogram polyominoes of area $n$, our algorithm generates them all in time $kN$ where $k$ is a constant. Its space complexity is $O(\sqrt{n})$.

The algorithm provided is based on an idea of the second author and R. Radicioni, who in their article [14] describe a CAT algorithm to generate particular linear partitions of an integer $n$ called ice piles. The algorithm is based on the preorder traversal of a spanning tree for the Hasse diagram of the partial order defined by the dominating relation between two partitions.

The article is divided into two main sections and ends with a short conclusion presenting some possible developments of this work.

In section 2 we provide the definitions, the order structure of the set of partitions and the combinatorial results allowing to define the bijection between parallelogram polyominoes and pairs of partitions. In section 3 we describe the algorithm by providing its pseudo-code and analyze its complexity.

## 2 Preliminaries

A linear partition of $n$ is a non-increasing sequence of nonnegative integers with sum $n$. We indicate by $\mathrm{LP}(n)$ the set of the linear partitions of $n$.
For any two integers $x, p$ with $p > 0$, we denote by $x^{[p]}$ the sequence $(\underbrace{x,\ldots,x}_{p})$ and by $\cdot$ the *catenation product* of sequences. The *length* of $s = (s_1,\ldots,s_l) \in \mathrm{LP}(n)$ is $l(s) = l$, the *height* is $h(s) = s_1$ and the *size* (or *weight*) is $w(s) = \sum s_i = n$. Moreover, we define $\Delta(s) = \sum_{i<l}(s_i - s_{i+1}) = h(s) - s_l$. Note that $s$ can be univocally written as $s = s_1^{[m_1]} \cdot s_{j_2}^{[m_2]} \cdot \ldots \cdot s_{j_k}^{[m_k]}$ with $s_1 > s_{j_2} > \cdots > s_{j_k}$ and $m_i > 0$. The following notations are useful when dealing with sequences, $s_{<i} = (s_1,\ldots,s_{i-1})$ $(s_{\leq i} = (s_1,\ldots,s_i))$ and $s_{>i} = (s_{i+1},\ldots,s_l)$ $(s_{\geq i} = (s_i,\ldots,s_l))$.

The set $\mathrm{LP}(n)$ can be ordered with respect to the *negative lexicographic* order:

**Definition 1.** *Let $s = (s_1, \ldots, s_l)$ and $t = (t_1, \ldots, t_m)$ belong to $LP(n)$. Then, $s <_{nlex} t$ if and only if there is $i$ such that $s_{<i} = t_{<i}$ and $s_i > t_i$.*

**Covering Partitions** On the set of integer partitions we define a binary relation $\frown$ that is of particular interest for generating parallelogram polyominoes.

**Definition 2.** *Let $s \in LP(n_1)$ and $t \in LP(n_2)$. Then, $s$ strongly covers $t$, denoted as $s \frown t$, if and only if $l = l(s) = l(t)$, $s_1 > t_1$ and $s_j > t_{j-1}$ for all $j$ with $1 < j \leq l$.*

From the previous definition we immediately get:

**Lemma 1.** *Let $s \in LP(n_1)$, $t \in LP(n_2)$ and $s \frown t$. Then $n_1 \geq n_2 + l(t) + \Delta(t)$.*

*Proof.* By Definition 2 one has

$$n_1 = s_1 + \sum_{i=2}^{l(s)} s_i \geq t_1 + 1 + \sum_{i=1}^{l(t)-1} (t_i + 1) = t_{l(t)} + \Delta(t) + l(t) + \sum_{i=1}^{l(t)-1} t_i = n_2 + l(t) + \Delta(t).$$

□

Figure 1 shows the linear partition $s$ of smallest weight that strongly covers a linear partition $t$. It clearly follows that $w(s) = w(t) + l(t) + \Delta(t)$.

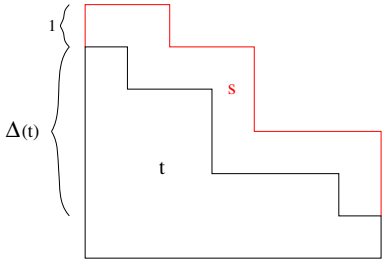



**Fig. 1.** The smallest covering partition

Given $t \in \mathrm{LP}(n_2)$, the set of linear partitions of $n_1$ that strongly cover $t$ is denoted by $\mathrm{LP}(n_1|t) = \{s \in \mathrm{LP}(n_1) | s \frown t\}$. Observe that for all $v \in \mathrm{LP}(n_1|t)$ one has $l(v) = l(t)$ and that $\mathrm{LP}(n_1) = \bigcup_{r=1\ldots n_1} \mathrm{LP}(n_1|0^{[r]})$. Definitions 1 and 2 let us easily characterize $\hat{s} = \min_{<_{\mathrm{nlex}}}(\mathrm{LP}(n_1|t))$.

**Lemma 2.** *Let $t = (t_1, \ldots, t_l) \in LP(n_2)$. Then, for $n_1 \geq n_2 + l + \Delta(t)$ one has*

$$\hat{s} = (t_1 + 1 + n_1 - n_2 - \Delta(t) - l, t_1 + 1, t_2 + 1, \ldots, t_{l-1} + 1).$$

We can interpret the elements of $\mathrm{LP}(n_1|t)$ as states of a simple discrete dynamical system which simulates the movements occurring into a heap of $n_1 - n_2$ grains stacked into a (two-dimensional) silo of radius $l(t)$ and whose bottom is shaped like the profile of $t$. This system has an evolution rule called *Move* which corresponds to moving one grain from column $i$ of $s$ to the first column $k$, with $k > i$, such that $s_i > s_{i+1}$ and $s_i - s_k \geq 2$. Figuratively, the rightmost grain of a *plateau* can slide on the plateau on its right and find place at the bottom of the next *cliff*. More formally, one has:

**Definition 3.** *Given $t \in LP(n_2)$, $s \in LP(n_1|t)$ and an integer $i$, let*

$$k = \begin{cases} i+1 & \text{if } s_i - s_{i+1} \geq 2 \\ j & \text{if } s_i - s_j \geq 2 \wedge s_{i+1} = s_{j-1} = s_i - 1 \\ \bot & \text{otherwise} \end{cases}$$

*Then*

$$\mathit{Move}(s,t,i) = \begin{cases} s_{<i} \cdot (s_i - 1, s_{i+1}, \ldots, s_{k-1}, s_k + 1) \cdot s_{>k} & \text{if } k \neq \bot, s_i - 1 > t_{i-1} \\ \bot & \text{otherwise} \end{cases}$$

The initial state of the dynamical system is $\hat{s}$, corresponding to the configuration where all the grains are in column 1, with the exception of those that create a covering layer of height 1 at the bottom of the silo. Figure 2 shows $s$ and $\mathrm{Move}(s,t,2)$ for $t = (3,3,3,2,1,1,1)$. Note that $\mathrm{Move}(s,t,i) = \bot$ for $i = 3,4,6,7$. We write $s \overset{i,t}{\Rightarrow} s'$ if $s' = \mathrm{Move}(s,t,i)$ and, more generally, $s \overset{\star}{\Rightarrow} v$ if there is a

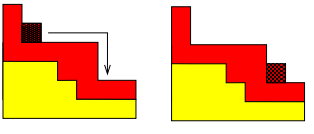


**Fig. 2.** $s = (6,5,4,4,4,2,2)$ and $\mathrm{Move}(s,t,2) = (6,4,4,4,4,3,2)$

sequence of moves leading from $s$ to $v$. The dynamical system has a set of states denoted by $G(\hat{s}|t) = \{v \in \mathrm{LP}(n_1|t) | \hat{s} \overset{\star}{\Rightarrow} v\}$, which turns out to be equal to $\mathrm{LP}(n_1|t)$, that is, all partitions of $n_1$ covering $t$ can be generated starting from the minimum in the neglex order.

**Lemma 3.** *Let $t = (t_1, \ldots, t_l) \in LP(n_2)$. Then, for any $n_1 \geq n_2 + l + \Delta(t)$ one has $LP(n_1|t) = G(\hat{s}|t)$.*

*Proof.* Suppose $G(\hat{s}|t) \subsetneq \mathrm{LP}(n_1|t)$ and let $s' = \min_{<_{\mathrm{nlex}}}(\mathrm{LP}(n_1|t) \setminus G(\hat{s}|t))$. Let $i$ be the largest integer such that $s'_i > t_{i-1} + 1$. If $i = 1$ then $s'_{>1} = \hat{s}_{>1}$ and so $s' = \hat{s}$. Thus, let $j_1 = \min\{k \leq i | s'_k = s'_i\}$, $j_2 = \min\{k < j_1 | s'_k = s'_{j_1 - 1}\}$ and set either $j = j_1$ (if $j_1 < i$) or $j = j_2$ (if $j_1 = i$). Now, consider $s'' = s'_{<j} \cdot (s'_j + 1, s'_{j+1}, \ldots, s'_{i-1}, s'_i - 1) \cdot s'_{>i}$ and note that $s'' \in \mathrm{LP}(n_1|t)$ and $s'' <_{\mathrm{nlex}} s'$. This implies $s'' \in G(\hat{s}|t)$. Lastly, from $s'' \overset{j,t}{\Rightarrow} s'$ we get $s' \in G(\hat{s}|t)$. □

The *set of moves* of $s \in \mathrm{LP}(n_1|t)$ is defined as the set of integers $\mathrm{M}(s|t) = \{i|\mathrm{Move}(s,t,i) \neq \perp\}$. $\mathrm{LP}(n_1|t)$ admits exactly one element $v$ such that $\mathrm{M}(v|t) = \emptyset$, called *fixed point* and denoted as $\mathrm{FP}(n_1|t)$. In particular, $\mathrm{LP}(n_1|t)$ turns out to be a lattice with respect to the relation induced by $\Rightarrow$, with the top and the bottom given by $\hat{s}$ and $\mathrm{FP}(n_1|t)$, respectively.

We denote by $s^{\downarrow}$ the linear partition in $\mathrm{LP}(n_1+1|t)$ given by $s^{\downarrow} = s_{<i} \cdot (s_i + 1) \cdot s_{>i}$, where $i$ is the largest integer such that $i \leq l(s)$ and $s_{i-1} > s_i$ ($i = 1$ if $l(s) = 1$). The following lemma provides a characterization of $\mathrm{FP}(n_1|t)$.

**Lemma 4.** *Let $t = (t_1, \ldots, t_l) \in LP(n_2)$ and $n_1 \geq n_2 + l + \Delta(t)$. Then*

$$v = FP(n_1|t) = \begin{cases} (t_1+1, t_1+1, t_2+1, \ldots, t_{l-1}+1) & \text{if } n_1 = n_2 + l + \Delta(t) \\ FP(n_1-1|t)^{\downarrow} & \text{otherwise} \end{cases}$$

*Proof.* The case $n_1 = n_2 + l + \Delta(t)$ follows by noting that $v \in \mathrm{LP}(n_1|t)$ and $\mathrm{Move}(v,t,i) = \perp$ for all $i$. Otherwise, let $u = \mathrm{FP}(n_1 - 1|t)$ and let $i$ be the only integer such that $u_i \neq u_i^{\downarrow}$. Since $u$ is a fixed point, it is sufficient to prove that $\mathrm{Move}(u^{\downarrow}, t, i) = \perp$. This follows from $u_{>i} = u^{\downarrow}_{>i}$ and $u_i = u_{i+1} = \ldots = u_l$. □

Figure 3 shows the linear partition $t = (5,4,4,2,2,2,1)$ together with the two fixed points $s = \mathrm{FP}(31|t)$ and $r = \mathrm{FP}(36|t)$ (note that $r \not\prec s$).

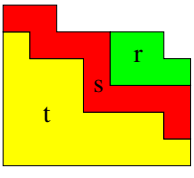



**Fig. 3.** $s = \mathrm{FP}(31|t)$ and $r = \mathrm{FP}(36|t)$

We associate with each linear partition in $\mathrm{LP}(n_1|t)$ its *grand ancestor*, that is, a particular element of $\mathrm{LP}(n_1|t)$ playing an important role in the process of generating $\mathrm{LP}(n_1|t)$. To enumerate the elements in neglex order, we should apply to the current partition $s$ the rightmost move, in order to preserve the longest possible prefix. However, whenever such move is also yet to be done in another partition $s'$ previously generated (and hence $s' <_{neglex} s$), the move needs to be applied not to $s$, but to $s'$ that is still waiting to make that move. $s'$ is the grand ancestor of $s$, it is the smallest partition for which $\max(\mathrm{M}(s|t))$ is still a valid move. Proper grand ancestors correspond to branching nodes in the implicit spanning tree that is being traversed to generate the partitions.

**Definition 4.** *Let $s \in LP(n_1|t)$. The* grand ancestor *of $s$ is the linear partition $A(s|t) = \min_{<_{nlex}}\{v \in LP(n_1|t) | s_{\leq i} = v_{\leq i}, i = \max(M(s|t)) \in M(v|t)\}$.*

The grand ancestor of a linear partition is characterized in

**Lemma 5.** *Let $s \in LP(n_1|t)$, $i = \max(M(s|t))$ and $q = \sum_{j>i}(s_j - t_{j-1} - 1)$. If $p$ is the largest integer such that $d = \sum_{j=i+1}^{i+p}(s_i - 1 - t_{j-1} - 1) \le q$, then $A(s|t)$ is the linear partition*

$$s' = s_{\le i} \cdot (s_i - 1)^{[p]} \cdot (t_{i+p} + 1 + q - d, t_{i+p+1} + 1, \ldots, t_{l-1} + 1).$$

*Proof.* First, note that $s' \in \mathrm{LP}(n_1|t)$, $i \in \mathrm{M}(s'|t)$ and $s'_{\le i} = s_{\le i}$. Thus we have only to prove that $s'$ is the smallest linear partition with such properties. In fact, suppose that $z \in \mathrm{LP}(n_1|t)$ satisfies $z <_{\mathrm{nlex}} s'$, with $z_{\le i} = s_{\le i}$ and $i \in \mathrm{M}(z|t)$. Let $j$ be the smallest integer such that $z_j > s'_j$ (obviously $j > i + p$). Then, one has $\sum_{e>j} z_e < \sum_{e>j} s'_e = \sum_{e>j}(t_{e-1} + 1)$. This implies the existence of an index $\hat{\jmath}$ such that $z_{\hat{\jmath}} < t_{\hat{\jmath}-1} + 1$ and so $z \not\prec t$. □

*Example 1.* Let $t = (5, 4, 4, 3, 2, 2, 2, 1, 1, 1, 1)$ and $s = (8, 8, 8, 8, 6, 5, 5, 4, 4, 4, 4)$. Then, $A(s|t) = (8, 8, 8, 8, 6, 5, 5, 5, 5, 4, 2)$ (see Figure 4). Note how $A(s|t)$ has the same prefix as $s$ up to position $\max(\mathrm{M}(s|t))$, whereas the remaining suffix has the leftmost entries as large as possible, while ensuring the covering condition.

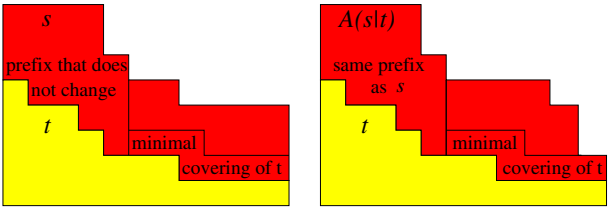



**Fig. 4.** A linear partition and its grand ancestor (right)

When $n_1$ and $t$ are clear from the context, we denote by $s^{(e)}$ the $e$th linear partition in $\mathrm{LP}(n_1|t)$. We can obtain a spanning tree associated with the lattice $\mathrm{LP}(n_1|t)$ by defining, for $e > 1$, $\mathrm{A}(s^{(e-1)}|t)$ as the father of $s^{(e)}$ ($\hat{s} = s^{(1)}$ is the root). The following lemma shows how we can implicitly traverse such tree in order to generate the sequence of elements in $\mathrm{LP}(n_1|t)$.

**Lemma 6.** *Let $t \in LP(n_2)$ and $n_1 \ge n_2 + l(t) + \Delta(t)$. Then, for any $e > 0$, we have*

$$A(s^{(e)}|t) \overset{i_e}{\Rightarrow} s^{(e+1)}, \qquad \text{with } i_e = \max(M(s^{(e)}|t)).$$

*Proof.* It is sufficient to follows the proof of [14, Lemma 2.16], where the same result is stated with respect to the states of a dynamical system associated with the Ice Pile Model. We have only to take into account the (slightly) different definition of the evolution rule (the function *Move*). □

We define a function Next which, having as inputs $s^{(e)}, t$, determines first $i = \max(\mathrm{M}(s^{(e)}|t))$ and $s' = \mathrm{A}(s^{(e)}|t)$, and then outputs $s^{(e+1)} = \mathrm{Move}(s', t, i)$ ($\mathrm{Next}(s^{(e),t}) = \perp$ if $s^{(e)} = \mathrm{FP}(n_1|t)$). Hence, Algorithm 1 generates $\mathrm{LP}(n_1|t)$.

**Theorem 1.** LinPartGen$(n_1, t)$ *generates* $LP(n_1|t)$ *in time* $O(\sharp LP(n_1|t))$ *using* $O(\sqrt{n_1})$ *space.*

*Proof.* (Outline) The space requirement follows from section 3.1 and by implementing $\mathrm{M}(s|t)$ as a stack of links to nodes (in the list of $s$) where moves occur. With respect to the time, the result follows by reasoning as in [14, Lemma 4.2] and showing that, by Lemma 5, $s' = \mathrm{A}(s^{(e)}|t)$ can be computed in time $O(d_e)$, where $d_e$ is the distance between $s'$ and $s^{(e)}$ in the spanning tree of $\mathrm{LP}(n_1|t)$. □

**Algorithm 1.** Exhaustive Generation of $\mathrm{LP}(n_1|t)$.

```
1: PROCEDURE LINPARTGEN(n_1, t)
2: s:=MINLINPART(n_1, t); {s = min_{<nlex}(LP(n_1|t))}
3: while M(s|t) ≠ ∅ do
4:    s:=NEXT(s, t);
5: end while
```

Similarly, we can easily generate the set $\{v \in \mathrm{LP}(n_1|t) | s \stackrel{\star}{\Rightarrow} v\}$ for any $s \in \mathrm{LP}(n_1|t)$. Then, one has:

**Corollary 1.** *For any* $s \in LP(n_1|t)$ *the set* $V = \{v \in LP(n_1|t) | s \stackrel{\star}{\Rightarrow} v\}$ *can be generated in time* $O(\sharp V)$ *using* $O(\sqrt{n_1})$ *space.*

**Parallelogram Polyominoes** A parallelogram polyomino is a polyomino whose boundary consists of two north-east lattice paths which are non-crossing (apart the starting and ending points). Without loss of generality, we suppose that the starting point of such paths is $(1, 0)$. Thus, a polyomino $P$ can be specified by a pair of paths $(p, q)$ where $p = (1, 0), (1, 1), \ldots, (x - 1, y), (x, y)$ and $q = (1, 0), (2, 0), \ldots, (x, y - 1), (x, y)$. The *height* of $P$ is $y$, the *width* is $x - 1$, while its *size* (or *area*) is the number of unit squares within the boundary identified by $p$ and $q$. The following theorem provides a useful bijection between polyominoes of size $n$ and suitable pairs of linear partitions.

**Theorem 2.** *The set of parallelogram polyominoes of size* $n$ *is in bijective correspondence with the set of pairs of linear partitions* $(s, t)$ *such that*

- $1 \leq l = l(t) \leq n$;
- $t_l = 1$;
- $l + \Delta(t) \leq n$;
- $t \in LP(m)$ *with* $l \leq m \leq (l - 1)(\Delta(t) + 1) + 1$;
- $s \in LP(m + n|t)$.

*Proof.* First, let us see how a polyomino $P$ with area $n$ univocally identifies a pair $(s, t)$ of linear partitions. Let $p = (1, 0), (1, 1), (x_1, y_1), \ldots, (x_k, y_k), (x - 1, y), (x, y)$ and $q = (1, 0), (2, 0), (x'_1, y'_1), \ldots, (x'_k, y'_k), (x, y - 1), (x, y)$ be the two

paths defining $P$. Obviously one has $y + x - 1 \leq n$. Note that there is $h$ such that the first $h + 2$ steps of $p$

$$(1,0),(1,1),(x_1,y_1),\ldots,,(x_h,y-1),(x_{h+1},y),$$

with $x_h = x_{h+1} \leq x-1$, identify a linear partition $t = (t_1,\ldots,t_l)$ with $l = y \leq n$, $t_1 = x_h$, $t_l = 1$ and $\Delta(t) = x_h - 1$. Thus, one has $l + \Delta(t) = y + x_h - 1 \leq n$ and $t \in \mathrm{LP}(m)$ for a suitable integer $m$ with $l \leq m \leq (l-1)t_1 + 1 = (l-1)(\Delta(t) + 1) + 1$. Similarly, by considering $q$ we can find an index $r$ such that the sequence $(x'_r, 0),(x'_{r+1},1),\ldots,(x'_k,y'_k),(x,y-1),(x,y)$ identifies a partition $s$ of size $m+n$ with $s \frown t$ (recall that the two paths are non-crossing), that is, $s \in \mathrm{LP}(m+n|t)$. Lastly, observe that if two different polyominoes of size $n$ are identified by two pairs of paths, say $(p,q)$ and $(p',q')$, then one necessarily has $p \neq p'$ or $q \neq q'$, and the associated pairs of linear partitions are different.

Now, we show how a pair $(s,t)$ of linear partitions satisfying the given conditions uniquely defines two non-crossing paths $p$, $q$ which start at $(1,0)$ and end at $(s_1, l(t))$. Indeed, $p$ is the only north-east path passing through the points

- for all $j$, with $1 \leq j \leq l$, $(t_j, l-j)$;
- if $t_{j-1} > t_j$ $(t_j, l-j+1),(t_j+1,l-j+1),\ldots,(t_{j-1}-1,l-j+1)$;
- $(t_1+1,l),(t_1+2,l),\ldots,(s_1-1,l)$.

Analogously, $q$ is identified by the points

- $(1,0),(2,0),\ldots,(s_l-1,0)$;
- for all $j$, with $1 \leq j \leq l$, $(s_j, l-j)$;
- if $s_{j-1} > s_j$ $(s_j,l-j+1),(s_j+1,l-j+1),\ldots,(s_{j-1}-1,l-j+1)$.

Note that $p$ and $q$ are non-crossing since $s \frown t$. Lastly, the size of the polyomino is just $w(s) - w(t) = n$. Moreover, if $(s,t) \neq (s',t')$ then the pair $(p,q)$ identified by $(s,t)$ is different from the pair $(p',q')$ associated with $(s',t')$ and the two polyominoes are different. □

*Example 2.* Figure 5 illustrates the parallelogram polyomino of size 27 identified by $(s,t)$ with $t = (8,8,6,5,5,5,2,2,1,1)$ and $s = (9,9,9,8,8,6,6,6,6,3)$. The two north-east paths $p,q$ associated with $(s,t)$ are drawn as dashed and dotted lines, respectively. You can see the two partitions $s$ and $t$ represented in the usual way by turning the picture 90 degrees counterclockwise.

We denote by $\mathrm{PPol}(n)$ the set of parallelogram polyominoes of size $n$. In the sequel, the generation of $\mathrm{PPol}(n)$ respects the following ordering which can be obtained by considering the bijection in Theorem 2.

**Definition 5.** *Let $(s,t)$ and $(s',t')$ be two pairs of linear partitions identifying $P, P' \in PPol(n)$, respectively. Then $P < P'$ if and only if*

$$l(s) > l(s')$$

*or*

$$l(s) = l(s'), t \in LP(m), t' \in LP(m'), m > m'$$

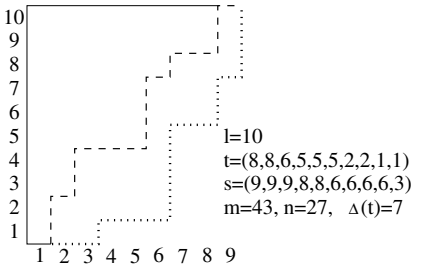



**Fig. 5.** A parallelogram polyomino identified by two linear partitions

*or*

$$l(s) = l(s'), t, t' \in LP(m), t <_{nlex} t'$$

*or*

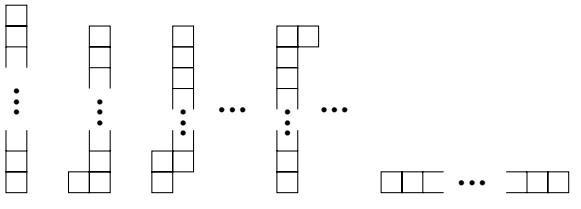
$$l(s) = l(s'), t = t', s <_{nlex} s'.$$

## 3 The Algorithm

Here we present an algorithm which solves the following problem:

**Problem** Parallelogram Polyominoes Generation (PPG)
**Input** an integer $n$
**Output** the ordered sequence of parallelogram polyominoes in PPol($n$).

Our approach is that of producing the sequence of parallelogram polyominoes (ordered with respect to Definition 5)



by generating the associated sequence of pairs $(s, t)$ of linear partitions characterized in Theorem 2. Thus, Definition 5 directly leads to Algorithm 2 shown below.

Procedure GenParPol generates the sequence of pairs $(s, t)$ of linear partitions associated with PPol($n$). The two loops at lines 2, 3 are used to set the height $l$ of the polyomino, that is, the length of $t$ (from $n$ down to 1) and its size $m$, respectively. Because of Lemma 1 and Theorem 2, one has $n + m \geq m + l + \Delta(t)$: this means that the largest value for $\Delta(t)$ is $n - l$ and thus $m$ has to satisfy the relation $l \leq m \leq (l-1)(n-l) + l$. The code

**Algorithm 2.** Generation of parallelogram polyominoes of size $n$.

```
 1: Procedure GenParPol(n)
 2: for l:=n downto 1 do
 3:    for m:=(l − 1)(n − l) + l downto l do
 4:       t:= InitLinPart(m, l, n − l + 1);
 5:       while M(t, 0) ≠ ∅ do
 6:          s:= MinLinPart(m + n, t);
 7:          while M(s, t) ≠ ∅ do
 8:             s:=Next(s, t);
 9:          end while
10:          t:=Next(t, 0^[l]);
11:       end while
12:    end for
13: end for
```

at lines 4–11 generates all the pairs $(s,t)$ such that $l(t) = l$, $t \in \mathrm{LP}(m)$ and $s \in \mathrm{LP}(m+n|t)$. InitLinPart$(m, l, n-l+1)$ returns the representation of the smallest linear partition of weight $m$ and length $l$ having height $n-l+1$, while MinLinPart$(m+n, t)$ returns $\min_{<_{\mathrm{nlex}}}(\mathrm{LP}(m+n|t))$. Next$(s,t)$ returns the linear partition in $\mathrm{LP}(m+n|t)$ which follows $s$ (w.r.t $<_{\mathrm{nlex}}$), while Next$(t, 0^{[l]})$ returns the linear partition of length $l$ following $t$.

### 3.1 The Data Structure

In order to efficiently deal with a pair $(s,t)$ of linear partitions with $s \frown t$, we define a data structure consisting of two double-linked lists. The list representing $t$ has as many elements as different values in $t$, that is, it has $d = \sharp\{i | t_i > t_{i+1}\}$ nodes which contain two integers (a value $t_i$ and the largest $k$ such that $t_k = t_i$),

$$(t_1, k_1), (t_{j_2}, k_2), \ldots, (t_{j_d}, k_d) \quad \approx \quad t_1^{[k_1]} \cdot t_{j_2}^{[k_2]} \cdot \ldots \cdot t_{j_d}^{[k_d]} = t.$$

The list representing $s$ has a similar structure, with the only difference that each node $(s_{i_p}, h_p)$ has a link to the node $(t_{j_q}, k_q)$ in the list of $t$ such that $k_{q-1} < h_p \leq k_q$. Moreover, in order to consider $t$ itself as a covering partition, we add a zero node representing $\mathrm{LP}(0)$. Figure 6 illustrates the representation of the linear partitions $s$ and $t$ drawn in Figure 5. Obviously, the sum of the lengths of the lists representing $t$ and $s \in \mathrm{LP}(n|t)$ is $O(\sqrt{n})$. This data structure is used in Algorithm 1 and its properties are useful to prove Theorem 1. In particular, it lets develop an implementation of function Move running in time $O(1)$, while permitting to update $\mathrm{M}(s|t)$ in time $O(1)$ too. Lastly, one has:

*Property 1.* Let $t \in \mathrm{LP}(m)$ and $n > l(t) + \Delta(t)$. If $k$ is the length of the list representing $s = \min_{<_{\mathrm{nlex}}}(\mathrm{LP}(m+n|t))$ then:

- $k = 2$ if and only if $s$ is a fixed point;
- $\sharp\{v \in \mathrm{LP}(m+n|t) | s \stackrel{\star}{\Rightarrow} v\} = \Omega(k)$.

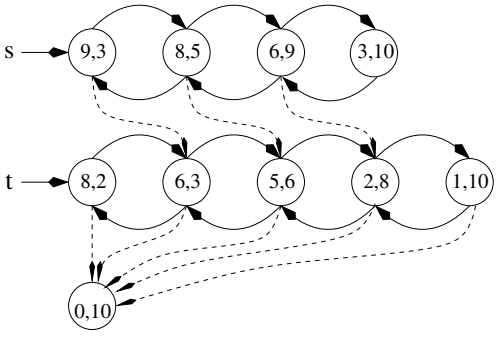



**Fig. 6.** Representing $(s, t)$

### 3.2 Complexity

By considering the data structure in Section 3.1, it is immediate to see that procedure InitLinPart runs in time $O(1)$ since it creates a list of length at most 3. Similarly, MinLinPart$(m+n, t)$ returns a list of length $l(t)$ and so it runs in time $O(l(t))$. Then, we can state:

**Theorem 3.** GenParPol*$(n)$ runs in time $O(\sharp PPol(n))$ and uses $O(\sqrt{n})$ space.*

*Proof.* First, note that the two for-loops at lines 2,3 set $O(n^2)$ different integer pairs $(l, m)$. Moreover, each pair $(l, m)$ is used (lines 4-11) to generate all polyominoes identified by a pair $(s, t)$ with $t \in \mathrm{LP}(m|0^{[l]})$ and $s \in \mathrm{LP}(m+n|t)$. Therefore, it is sufficient to prove that the code consisting of lines 4-11 is CAT. To this aim, by Theorem 1 and Property 1, for each fixed $t$, the cost of instructions at lines 6-9 is $O(\sharp V)$, where $V = \{v | s \stackrel{*}{\Rightarrow} v\}$ and $s = \min_{<_{\mathrm{nlex}}}(\mathrm{LP}(m+n|t))$. Similarly, by Corollary 1 the generation of all $t \in \mathrm{LP}(m|0^{[l]})$ with $h(t) \leq n-l+1$ (the outer while -loop) is CAT. Hence, the set $F(l,m) = \{(s,t) | t \in \mathrm{LP}(m|0^{[l]}), s \in \mathrm{LP}(m+n|t), h(t) \leq n-l+1\}$ is CAT generated and, since $\sharp\mathrm{PPol}(n) = \sum_{l=1}^{n} \sum_{m=l}^{(l-1)(n-l)+l} \sharp F(l,m)$, the result follows. With respect to the space complexity, we recall that the length of the lists representing $s$ and $t$ is $O(\sqrt{n})$. □

## 4 Conclusions

It is noteworthy that the idea behind the initial algorithm intended to generate ice piles has proved to be extremely adaptable, allowing to provide CAT algorithms for the exhausting generation of a still growing set of classes of combinatorial objects : (linear) integer partitions, symmetric ice and sand piles [13,15], unimodal sequences, plain partitions, standard Young tableaux and now parallelogram polyominoes. It seems to be possible to modify the algorithms to generates other families of combinatorial objects, such as semi-standard Young tableaux. This raises a more general question in order theory, namely, finding a

characterization of posets whose Hasse diagram admits a spanning tree having a natural way to be traversed, defining this way a total order on the elements of the poset. Our approach to generate all the elements of such posets could likely be applied to all of them.

## References


1. Barcucci, E., Frosini, A., Rinaldi, S.: Direct-convex polyominoes: ECO method and bijective results. In: Brak, R., Foda, O., Greenhill, C., Guttman, T., Owczarek, A. (eds.) Proceedings of Formal Power Series and Algebraic Combinatorics 2002, Melbourne (2002)
2. Barcucci, E., Del Lungo, A., Nivat, M., Pinzani, R.: Reconstructing convex polyominoes from horizontal and vertical projections. Theoret. Comp. Sci. 155(2), 321–347 (1996)
3. Barcucci, E., Del Lungo, A., Pergola, E., Pinzani, R.: ECO:a methodology for the enumeration of combinatorial objects. J. of Diff. Eq. and App. 5, 435–490 (1999)
4. Bousquet-Mélou, M.: A method for the enumeration of various classes of column-convex polygons. Discrete Math. 154, 1–25 (1996)
5. Castiglione, G., Vaglica, R.: Recognizable Picture Languages and Polyominoes. In: Bozapalidis, S., Rahonis, G. (eds.) CAI 2007. LNCS, vol. 4728, pp. 160–171. Springer, Heidelberg (2007)
6. De Carli, F., Frosini, A., Rinaldi, S., Vuillon, L.: On the Tiling System Recognizability of Various Classes of Convex Polyominoes. Ann. Comb. 13, 169–191 (2009)
7. Delest, M., Dubernard, J.P., Dutour, I.: Parallelogram Polyominoes and Corners. J. Symb. Comp. 20, 503–515 (1995)
8. Delest, M., Viennot, X.G.: Algebraic languages and polyominoes enumeration. Theoret. Comp. Sci. 34, 169–206 (1984)
9. Del Lungo, A., Duchi, E., Frosini, A., Rinaldi, S.: On the generation and enumeration of various classes of convex polyominoes. Electronic Journal of Combinatorics 11, 60 (2004)
10. Del Lungo, A., Mirolli, M., Pinzani, R., Rinaldi, S.: A Bijection for Directed-Convex Polyominoes. In: Proc. of DM-CCG 2001, Discrete Mathematics and Theoretical Computer Science AA, pp. 133–144 (2001)
11. Golomb, S.W.: Checker Boards and Polyominoes. The American Mathematical Monthly 61, 675–682 (1954)
12. Kuba, A., Balogh, E.: Reconstruction of convex 2D discrete sets in polynomial time. Theoret. Comp. Sci. 283(1), 223–242 (2002)
13. Massazza, P.: A CAT algorithm for sand piles. PU.M.A. 19(2-3), 147–158 (2008)
14. Massazza, P., Radicioni, R.: A CAT algorithm for the exhaustive generation of ice piles. RAIRO Theoretical Informatics and Applications 44, 525–543 (2010)
15. Massazza, P., Radicioni, R.: On the Exhaustive Generation of Symmetric Sand Piles. In Proc. of GASCom 2010, Montréal, September 2-4 (2010)
16. Ollinger, N.: Tiling the Plane with a Fixed Number of Polyominoes. In: Dediu, A.H., Ionescu, A.M., Martín-Vide, C. (eds.) LATA 2009. LNCS, vol. 5457, pp. 638–647. Springer, Heidelberg (2009)
17. Pergola, E., Sulanke, R.A.: Schröder Triangles, Paths, and Parallelogram Polyominoes. Journal of Integer Sequences, 1 Article 98.1.7 (1998)
18. Polya, G.: On the number of certain lattice polygons. J. Comb. Theory 6, 102–105 (1969)