On the generation of 2-polyominoes

Enrico Formenti[1] and Paolo Massazza[2]

[1] Université Côte d'Azur (UCA), CNRS, I3S, France
enrico.formenti@unice.fr
[2] Università degli Studi dell'Insubria, Department of Theoretical and Applied Sciences - Computer Science Section, Via Mazzini 5, 21100 Varese, Italy
paolo.massazza@uninsubria.it

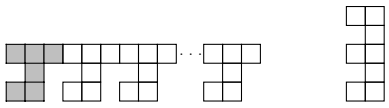
Abstract. The class of 2-polyominoes contains all polyominoes P such that for any integer i, the first i columns of P consist of at most 2 polyominoes. We provide a decomposition that allows us to exploit suitable discrete dynamical systems to define an algorithm for generating all 2-polyominoes of area n in constant amortized time and space $O(n)$.

1 Introduction

A *polyomino* is a finite and connected union of unitary squares (*cells*) in the plane $\mathbb{Z} \times \mathbb{Z}$, considered up to translations [8]. The number of cells of a polyomino is its *area*. The problem of counting the number of polyominoes of area n (*i.e.* determining a closed formula for $|\mathsf{Pol}(n)|$) is still open and the most used algorithm to generate $\mathsf{Pol}(n)$ runs in exponential time and uses exponential space [9]. Due to the space requirement, such algorithm has been used to compute $|\mathsf{Pol}(n)|$ only for $n \leq 56$ [10]. So, it is interesting to study suitable representations of polyominoes that allow to design efficient generating algorithms with strict constraints on the space. To classify the polyominoes and to tackle some difficult questions about them, several subclasses have been introduced in literature. For instance, the class of convex polyominoes and its subclasses have been studied under different points of view [6, 1, 4, 5]. Recently, some efficient algorithms for the exhaustive generation by area of some subclasses of convex polyominoes have been presented [14, 12, 3, 2].

The problem of efficiently generating non-convex polyominoes is particularly difficult. Indeed, no Constant Amortized Time (CAT) algorithm using space $O(n)$ is known for the generation of $\mathsf{Pol}(n)$. In order to answer this question, we tackled some special cases in the hope of detecting all the conceptual obstacles and trying to remove them. Indeed, we are interested in the class $\mathsf{Pol}_k(n)$ containing all polyominoes P of area n such that, for any $i > 0$, the first i columns of P are made of at most k polyominoes. A CAT algorithm for the exhaustive generation of $\mathsf{Pol}_1(n)$ (called partially directed animals in [15]), has been proposed in [7]. This paper provides a CAT algorithm that generates $\mathsf{Pol}_2(n)$ using space $O(n)$. Remark that $\mathsf{Pol}_2(n)$ is a strictly new class which neither coincides with the class of partially directed animals with k sources (see Figure 1) nor with other known subclasses of polyominoes.

The algorithm is based on an idea that was already used for other exhaustive generation algorithms (for instance, [12, 11, 13]) and it is inspired by discrete



a) b)

Fig. 1. a) a 2-polyomino corresponding to a partially directed animal with $k+1$ sources (k is the number of repetitions of the gray pattern). b) a partially directed animal with 3 sources which is not a 2-polyomino.

dynamical systems theory. Indeed, it proceeds to generate a polyomino column by column, and each column is uniquely decomposed into at most four parts, namely the skeleton and three regions. Each region corresponds to the state of a suitable dynamical system. Proving that the dynamical system is surjective and that it has one orbit grants exhaustive generation. Granting that the next state can be computed from the previous one in constant amortized time and in linear space leads to the result. This last step is accomplished through a careful choice of the data structure used to represent a polyomino.

Because of the lack of space, most of the proofs and the pseudo-code have been omitted. They will appear in the long version of the paper.

2 Preliminaries

The *area* $\mathrm{A}(P)$ of a polyomino P is the number of its cells. A polyomino can be seen as a (finite) sequence of columns. A *column* consists of a sequence of *vertical segments* separated by empty unit squares. A *vertical segment* is a sequence of unit cells $q_1, q_2, \ldots, q_k$ that are in the same column and such that q_i is edge-adjacent to q_{i+1}, for $1 \leq i < k$. The *position* of a cell is its y-coordinate. The position of the top (resp., bottom) cell of a segment s is denoted by $\mathrm{Top}(s)$ (resp., $\mathrm{Bot}(s)$). We represent a segment s of a column by means of the pair $(\mathrm{A}(s), \mathrm{Top}(s))$. Segments belonging to the same column are numbered from the top to the bottom, thus a column with p segments is simply a sequence of disjoint segments $\mathbf{c} = (s_1, \ldots, s_p)$, with $\mathrm{Top}(s_i) < \mathrm{Top}(s_{i-1}) - \mathrm{A}(s_{i-1}) - 1$ for $1 < i \leq p$. The *area* of $\mathbf{c}$ is $\mathrm{A}(\mathbf{c}) = \sum_{i=1}^{p} \mathrm{A}(s_i)$. Furthermore, the position of $\mathbf{c}$ is the position of its first segment, $\mathrm{Top}(\mathbf{c}) = \mathrm{Top}(s_1)$. Given a segment s and an integer j such that $\mathrm{Top}(s) > j \geq \mathrm{Bot}(s)$, we denote by $s_{>j}$ (resp., $s_{\leq j}$) the subsegment consisting of the cells of s with position greater than j (resp., smaller than or equal to j). The part of a column $\mathbf{c}$ that is above a position j is $\mathbf{c}_{>j}$ ($\mathbf{c}_{\geq j}$, $\mathbf{c}_{\leq j}$ and $\mathbf{c}_{<j}$ are defined similarly). Segments can be ordered with respect to their position and their area.

Definition 1 ($<$ on segments). *Let u and v be two segments. Then, one has $u < v$ if and only if $\mathrm{Top}(u) > \mathrm{Top}(v)$ or $\mathrm{Top}(u) = \mathrm{Top}(v)$ and $A(u) > A(v)$.*

A total order on columns, denoted by $\prec$, can be easily obtained by extending $<$.

Definition 2 ($\prec$ on columns). *Let $\mathbf{b} = (s_1, \ldots, s_p)$ and $\mathbf{c} = (t_1, \ldots, t_q)$ be two columns. Then, one has $\mathbf{b} \prec \mathbf{c}$ if and only if either $A(\mathbf{b}) > A(\mathbf{c})$, or $A(\mathbf{b}) = A(\mathbf{c})$ and there exists m with $1 \leq m \leq \min(p, q)$ such that $s_j = t_j$ for all $j < m$ and $s_m < t_m$.*

Consider a polyomino $P = (\mathbf{c}_1, \ldots, \mathbf{c}_q)$. We assume that the position of the bottom cell of the last segment of $\mathbf{c}_1$ is 0. We indicate by $P_{\leq i}$ (resp., P_i) the sequence of the first i columns of P from the left (resp., the i-th column of P), also called the *i-prefix* of P. We also write $P_{<i}$ for $P_{\leq i-1}$. The area of P is $\mathrm{A}(P) = \sum_{i=1}^{w} \mathrm{A}(P_i)$ where w is the number of columns of P. From here on, $\mathsf{Pol}(n)$ denotes the set of the polyominoes of area n.

Definition 3. ***(The class*** $\mathsf{Pol}_k(n)$***)*** *Let $k > 0$. The class of k-polyominoes of area n, $\mathsf{Pol}_k(n)$, contains all $P \in \mathsf{Pol}(n)$ such that, for all i with $1 \leq i < w$, $P_{\leq i}$ consists of at most k polyominoes, where w is the number of columns of P.*

The class $\mathsf{Pol}_1(n)$ corresponds to the class of partially directed animals [17, 15, 16], for which a CAT generation algorithm has been provided in [7]. Here, we are interested into an efficient exhaustive generation algorithm for the class $\mathsf{Pol}_2(n)$.

Two segments s and t are *adjacent* if they belong to adjacent columns and there exists at least one cell of s that is edge-adjacent to a cell of t. The set $\mathrm{LAdj}(s)$ of *left-adjacencies* of a segment s contains the positions of the cells of s that are edge-adjacent to some cells of segments in P_{i-1}. A segment s is *l-detached* if $\mathrm{LAdj}(s) = \emptyset$, whereas it is *l-adjacent* if $\mathrm{LAdj}(s) \neq \emptyset$. A column that does not contain l-detached segments is called *l-detached-free*. The segment of P_i immediately above (resp., below) s is $s^{\uparrow}$ (resp., $s_{\downarrow}$). If s is l-detached then ${}_{\downarrow}s$ (resp., ${}^{\uparrow}s$) indicates the first (resp., last) segment v in P_{i-1} such that $\mathrm{Top}(v) < \mathrm{Bot}(s)$ (resp., $\mathrm{Bot}(v) > \mathrm{Top}(s)$). Otherwise, if s is l-adjacent then ${}^{\uparrow}s$ (resp., ${}_{\downarrow}s$) indicates the segment in P_{i-1} that has a cell in the position $\min(\mathrm{LAdj}(s))$ (resp., $\max(\mathrm{LAdj}(s))$). For any segment s of P_i, $\mathrm{Con}(s)$ is the largest polyomino comprised in $P_{\leq i}$ that contains s. Let $P_{\leq i}$ consist of two polyominoes U and V. Then, $U^{\uparrow}$ (resp., $V^{\uparrow}$) indicates the *upper extreme* of U (resp., V) in P_i, that is, the first segment $u \in U$ (resp., $v \in V$) of P_i such that $u^{\uparrow} \in V$ or $u_{\downarrow} \in V$ (resp., $v^{\uparrow} \in U$ or $v_{\downarrow} \in U$). Analogously, $U_{\downarrow}$ (resp., $V_{\downarrow}$) denotes the *lower extreme* of U (resp., V) in P_i that is, the last segment $u \in U$ (resp., $v \in V$) of P_i such that $u^{\uparrow} \in V$ or $u_{\downarrow} \in V$ (resp., $v^{\uparrow} \in U$ or $v_{\downarrow} \in U$). If $U_{\downarrow} = U^{\uparrow}$ then U has only one extreme in P_i, denoted by $U^{\updownarrow}$. We say that U and V are *independent* if both U and V have only one extreme in P_i ($U^{\updownarrow}$ is above $V^{\updownarrow}$ or vice versa), see Cases $b), c)$ and $d)$ of Fig. 2. Similarly, U *includes* V if $V^{\uparrow}$ is immediately below $U^{\uparrow}$ and $V_{\downarrow}$ is immediately above $U_{\downarrow}$, see Cases $f)$ and $g)$ of Fig. 2. Thus, U and V are *enclosing* if U includes V or vice versa.

A segment s of P_i is a *bridge* if $P_{<i}$ consists of two polyominoes $\bar{U}$ and $\bar{V}$ and s is l-adjacent to two segments v and $v_{\downarrow}$, which are extremes of $\bar{U}$ and $\bar{V}$, respectively. The *distance* $\mathrm{Dist}(s)$ of an extreme s is the number of empty unit squares that separate s from the nearest segment $t \in \{s_{\downarrow}, s^{\uparrow}\}$ with $\mathrm{Con}(t) \neq \mathrm{Con}(s)$. In other words, $\mathrm{Dist}(s) + 2$ is the area of the smallest bridge that can join s and t. The *vertical distance* of two segments s, t with $\mathrm{Bot}(s) \geq \mathrm{Top}(t)$ is $d(s,t) = \mathrm{Bot}(s) - \mathrm{Top}(t) - 1$. If $P_{\leq i}$ consists of two distinct polyominoes U and V, we also set $\mathrm{Dist}(U,V) = \min\{\mathrm{Dist}(s) | s \text{ is an extreme of } U\}$.

Notation: from here on, $P_{\leq i}$ (resp., $P_{<i}$) denotes an i-prefix (resp., $(i-1)$-prefix) of a polyomino in $\mathsf{Pol}_2(n)$. A bar over a segment indicates that it is a bridge, $\bar{s}$.

Given a sequence S of $i-1$ columns and a column $\mathbf{c}$, the operation $|$ of *column concatenation* (left associative) produces a new sequence $S' = S|\mathbf{c}$ with i columns. A column $\mathbf{c}$ is *compatible* with P_{i-1}, denoted by $\mathbf{c} \,\square\, P_{i-1}$, if and only if $P_{<i}|\mathbf{c}$ is the i-prefix of a polyomino in $\mathsf{Pol}_2(n)$. Obviously, one has $\mathbf{c} \,\square\, P_{i-1}$ if and only if $P_{<i}|\mathbf{c} \in \mathsf{Pol}_2(n)$ or there exists a column $\mathbf{d}$ such that $P_{<i}|\mathbf{c}|\mathbf{d} \in \mathsf{Pol}_2(n)$. We denote by $\mathrm{Comp}(P_{<i}, k, n)$ the set of columns of area k that are compatible with P_{i-1}. Bridges and l-adjacent extremes are called *special segments*. They determine a unique decomposition of $\mathbf{b}$ into *regions*, and they form the so-called *skeleton* of $\mathbf{b}$. The skeleton of a column $\mathbf{b} \in \mathrm{Comp}(P_{<i}, k, n)$, denoted by $\mathrm{Ske}(\mathbf{b})$, is either the column consisting of the bridges in $\mathbf{b}$ (if any), or the column containing all l-adjacent extremes in $\mathbf{b}$ (if $\mathbf{b}$ does not contain a bridge and $P_{<i} \notin \mathsf{Pol}$).

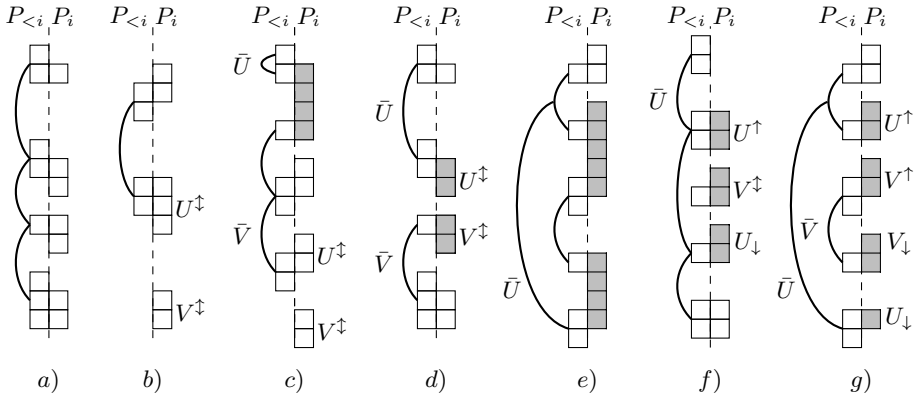


Fig. 2. Columns, compatibility and skeletons (represented by gray cells).

Definition 4. *Let* $\mathbf{b} \in \mathrm{Comp}(P_{<i}, k, n)$. *Then, the skeleton of* $\mathbf{b}$ *is the column*

$$\mathrm{Ske}(\mathbf{b}) = \begin{cases} () & \text{if } P_{<i} \in \mathsf{Pol} \\ (\bar{s}) & \text{if } \mathbf{b} \text{ contains only one bridge } \bar{s} \\ (\bar{s}, \bar{t}) & \text{if } \mathbf{b} \text{ contains two bridges } \bar{s}, \bar{t} \\ (s, s_{\downarrow}) & \text{if } P_{<i}|\mathbf{b} = U \cup V, s = U^{\updownarrow}, s_{\downarrow} = V^{\updownarrow} \\ (s, s_{\downarrow}, s_{\downarrow\downarrow}) & \text{if } P_{<i}|\mathbf{b} = U \cup V, s = U^{\uparrow}, s_{\downarrow} = V^{\updownarrow}, s_{\downarrow\downarrow} = U^{\downarrow} \\ (s, s_{\downarrow}, u, u_{\downarrow}) & \text{if } P_{<i}|\mathbf{b} = U \cup V, s = U^{\uparrow}, s_{\downarrow} = V^{\uparrow}, u = V^{\downarrow}, u_{\downarrow} = U^{\downarrow} \end{cases}$$

where $U^{\uparrow}, U^{\downarrow}, V^{\uparrow}, V^{\downarrow}, V^{\updownarrow}, U^{\updownarrow}$ *are l-adjacent extremes of* $\mathbf{b}$ *(with no bridges).*

Notice that if $\mathbf{b}$ contains a bridge t then it does not contain an l-adjacent extreme s with $\mathrm{Con}(t) \neq \mathrm{Con}(s)$ (otherwise $P_{<i}$ consists of at least three polyominoes). For any $\mathbf{b} \in \mathrm{Comp}(P_{<i}, k, n)$ such that $\mathrm{Ske}(\mathbf{b}) \neq ()$ one has $\mathrm{Ske}(\mathbf{b}) \in \mathrm{Comp}(P_{<i}, h, n)$ for a suitable integer $h \leq k$.

Definition 5. *The northern (resp., central, southern) region of* $\mathbf{b} = (s_1, \ldots, s_q)$ *is denoted by* $\mathbf{b}^{\uparrow}$ *(resp.,* $\mathbf{b}^{\updownarrow}$*,* $\mathbf{b}_{\downarrow}$*) and it is defined as follows (Ske*$(\mathbf{b})$ *to the left):*

(): $\mathbf{b}^{\uparrow} = \mathbf{b}$*;*
$(\bar{s}_m)$**:** $\mathbf{b}^{\uparrow} = (s_1, \ldots, s_{m-1})$ *and* $\mathbf{b}_{\downarrow} = (s_{m+1}, \ldots, s_q)$*;*
$(\bar{s}_m, \bar{s}_p)$**:** $\mathbf{b}^{\uparrow} = (s_1, \ldots, s_{m-1})$, $\mathbf{b}^{\updownarrow} = (s_{m+1}, \ldots, s_{p-1})$, $\mathbf{b}_{\downarrow} = (s_{p+1}, \ldots, s_q)$*;*
(s_m, s_{m+1})**:** $\mathbf{b}^{\uparrow} = (s_1, \ldots, s_{m-1})$, $\mathbf{b}_{\downarrow} = (s_{m+2}, \ldots, s_q)$*;*
(s_m, s_{m+1}, s_{m+2})**:** $\mathbf{b}^{\uparrow} = (s_1, \ldots, s_m)$, $\mathbf{b}_{\downarrow} = (s_{m+3}, \ldots, s_q)$*;*
$(s_m, s_{m+1}, s_p, s_{p+1})$**:** $\mathbf{b}^{\uparrow} = (s_1, \ldots, s_{m-1})$, $\mathbf{b}^{\updownarrow} = (s_{m+2}, \ldots, s_{p-1})$, $\mathbf{b}_{\downarrow} = (s_{p+2}, \ldots, s_q)$*;*

Remark that if s and t are l-adjacent segments belonging to the same region then $\mathrm{Con}(s) = \mathrm{Con}(t)$. The notion of skeleton induces an equivalence relation on $\mathrm{Comp}(P_{<i}, k, n)$.

Definition 6 ($\diamond$). *Let* $\mathbf{b}, \mathbf{c} \in \mathit{Comp}(P_{<i}, k, n)$*. Then, one has* $\mathbf{b} \diamond \mathbf{c}$ *if and only if* $\mathit{Ske}(\mathbf{b}) = \mathit{Ske}(\mathbf{c})$*.*

The set $\mathrm{Sk}(P_{<i}, k, n) = \{\mathbf{c} \in \mathrm{Comp}(P_{<i}, j, n) | j \leq k \wedge \mathrm{Ske}(\mathbf{c}) = \mathbf{c}\}$ contains the skeletons of area at most k that are compatible with P_{i-1}. Obviously, one has

$$\mathrm{Comp}(P_{<i}, k, n) = \bigcup_{\mathbf{b} \in \mathrm{Sk}(P_{<i}, j, n)} [\mathbf{b}]_{\diamond}.$$

Notice that $\mathrm{Sk}(P_{<i}, j, n) = \emptyset$ if $\mathrm{Sk}(P_{<i}, j+1, n) \setminus \mathrm{Sk}(P_{<i}, j, n) = \emptyset$. We define a total order $\lhd$ on $\mathrm{Sk}(P_{<i}, k, n)$ as follows.

Definition 7 ($\lhd$ on $\mathbf{Sk}(P_{<i}, k, n)$). *Let* $\mathbf{b}, \mathbf{c} \in \mathit{Sk}(P_{<i}, k, n)$*. Then, one has* $\mathbf{b} \lhd \mathbf{c}$ *if and only if one of the following conditions holds:*

- $|\mathbf{b}| < |\mathbf{c}|$*, (*i.e. $\mathbf{b}$ *contains less segments than* $\mathbf{c}$*);*
- $\mathbf{b}$ *contains a bridge whereas* $\mathbf{c}$ *does not;*
- $\mathbf{b} = (\bar{s}), \mathbf{c} = (\bar{t})$ *and* $\bar{s} < \bar{t}$*;*
- $\mathbf{b} = (s, t)$ *(resp.,* $\mathbf{b} = (\bar{s}, \bar{t})$*),* $\mathbf{c} = (v, w)$ *(resp.,* $\mathbf{c} = (\bar{v}, \bar{w})$*) and either* $s < v$ *(resp.,* $\bar{s} < \bar{v}$*), or* $s = v$ *(resp.,* $\bar{s} = \bar{v}$*) and* $t < w$ *(resp.,* $\bar{t} < \bar{w}$*);*
- $\mathbf{b} = (s, t, u)$*,* $\mathbf{c} = (v, w, z)$ *and either* $s < v$*, or* $s = v$ *and* $t < w$*, or* $s = v$*,* $t = w$ *and* $u < z$*;*
- $\mathbf{b} = (q, r, s, t)$*,* $\mathbf{c} = (u, v, w, z)$ *and either* $q < u$*, or* $q = u$ *and* $r < v$*, or* $q = u$*,* $r = v$ *and* $s < w$*, or* $q = u$*,* $r = v$*,* $s = w$ *and* $t < z$*;*

In Sect. 4, polyominoes of $\mathsf{Pol}_2(n)$ are generated column by column. All the columns that are compatible with P_{i-1} are generated according to the following ordering.

Definition 8 ($<$ on columns). *Let* $\mathbf{b}, \mathbf{c} \in \mathit{Comp}(P_{<i}, k, n)$*. Then, one has* $\mathbf{b} < \mathbf{c}$ *iff* $\mathit{Ske}(\mathbf{b}) \lhd \mathit{Ske}(\mathbf{c})$ *or* $\mathit{Ske}(\mathbf{b}) = \mathit{Ske}(\mathbf{c})$ *and one of the following holds:*

(Ske(b) is null) $\mathbf{b} \prec \mathbf{c}$*;*
(Ske(b) is $(\bar{s})$ **or** (s, t) **or** (s, t, u)**)** *either* $\mathbf{b}^{\uparrow} \prec \mathbf{c}^{\uparrow}$*, or* $\mathbf{b}^{\uparrow} = \mathbf{c}^{\uparrow} \wedge \mathbf{b}_{\downarrow} \prec \mathbf{c}_{\downarrow}$*;*
(Ske(b) is $(\bar{s}, \bar{t})$ **or** (s, t, u, v)**)** *either* $\mathbf{b}^{\uparrow} \prec \mathbf{c}^{\uparrow}$*, or* $\mathbf{b}^{\uparrow} = \mathbf{c}^{\uparrow} \wedge \mathbf{b}^{\updownarrow} \prec \mathbf{c}^{\updownarrow}$*, or* $\mathbf{b}^{\uparrow} = \mathbf{c}^{\uparrow} \wedge \mathbf{b}^{\updownarrow} = \mathbf{c}^{\updownarrow} \wedge \mathbf{b}_{\downarrow} \prec \mathbf{c}_{\downarrow}$*;*

3 Dynamical systems for regions

Given a column $\mathbf{b}$, its skeleton $\mathrm{Ske}(\mathbf{b})$ uniquely identifies the regions $\mathbf{b}^{\uparrow}$, $\mathbf{b}_{\downarrow}$ and $\mathbf{b}^{\updownarrow}$. Recall that a region of $\mathbf{b}$ is a particular subsequence of $\mathbf{b}$ consisting of consecutive segments delimited by special segments in $\mathrm{Ske}(\mathbf{b})$. In each region, we are going to use specific dynamical systems such that their current state represents a list of segments that when completed with the skeleton produces a compatible column. All compatible columns can be built in this way. Given an i-prefix $P_{<i}|\mathbf{b}$, with $h = \mathrm{A}(\mathbf{b})$ $(1 \leq h \leq n - \mathrm{A}(P_{<i}))$, and a segment s of a region $\mathbf{R}$ of $\mathbf{b}$, a *move* at $j \in [\mathrm{Bot}(s)\,, \mathrm{Top}(s)]$ is an operation which rearranges the cells of $\mathbf{R}$ while respecting a constraint (expressed by a value $\delta \geq 0$) regarding the creation of l-detached segments. More precisely, if $\delta = 0$ then no new l-detached segment can be created (because an l-detached segment is already present in $\mathbf{b}$, or the number of remaining cells is not sufficient to create a bridge in the next column, that is, $n - \mathrm{A}(P_{<i}|\mathbf{b}) < 3$, or $\mathrm{Ske}(\mathbf{b})$ contains at least two l-adjacent extremes), whereas $\delta > 0$ allows the creation of a new l-detached segment with distance at most δ only if $\mathbf{b}$ is l-detached free (moreover, $\mathrm{Ske}(\mathbf{b})$ should not contain l-adjacent extremes and $\delta + 2 = n - \mathrm{A}(P_{<i}|\mathbf{b})$). Two types of moves are devised, namely *shift* and *split*, and both output a region $\mathbf{R}'$ such that $\mathbf{R} \prec \mathbf{R}'$. Notice that, since $\mathrm{Ske}(\mathbf{b}) \in \mathrm{Comp}(P_{<i}, j, n)$ (with $j \leq h$), any rearrangement $\mathbf{R}'$ of the cells of $\mathbf{R}$ such that

1. $\mathbf{R}'$ is l-detached free if $\delta = 0$,
2. no cell of $\mathbf{R}'$ becomes edge-adjacent to a segment of $\mathrm{Ske}(\mathbf{b})$,
3. if $\delta > 0$ then $\mathbf{R}'$ contains at most one l-detached segment s' and $\mathrm{Dist}(s') \leq \delta$

produces a column $\mathbf{b}' \in \mathrm{Comp}(P_{<i}, h, n)$. A shift move may occur only in the position $j = \mathrm{Top}(s)$ of a segment s. It has the effect of shifting the segment s by e positions downwards, where e is the smallest integer such that the resulting region satisfies Conditions 1–3 above, see Fig. 2 (left). Possibly, s and $s_{\downarrow}$ are joined (if $s_{\downarrow} \notin \mathrm{Ske}(\mathbf{b})$). If such an e does not exist, then the move is undefined.

A split move may occur only in a position $j \in [\mathrm{Bot}(s)\,, \mathrm{Top}(s))$ of a segment s. In this case, s is split into $s_{>j}$ and $s_{\leq j}$. The segment $s_{>j}$ stays in place whereas $s_{\leq j}$ is shifted e positions downwards, where e is the smallest integer such that the resulting region $\mathbf{R}'$ satisfies Conditions 1–3 above, see Fig. 2 (center). Possibly, $s_{\leq j}$ and $s_{\downarrow}$ are joined (if $s_{\downarrow} \notin \mathrm{Ske}(\mathbf{b})$), see Fig. 2 (right).

Denote by $\mathrm{Mcr}(P_{<i}|\mathbf{b}, \mathbf{R}, \delta)$ the set of the positions where a move occurs on a segment of $\mathbf{R}$. The following lemma relates the non-emptyness of $\mathrm{Mcr}(P_{<i}|\mathbf{b}, \mathbf{R}, \delta)$ to the number of segments in $\mathbf{R}$.

Lemma 1. *Let $\mathbf{R}$ be a region of $\mathbf{b}$. If $\mathbf{R}$ contains at least three segments, then $Mcr(P_{<i}|\mathbf{b}, \mathbf{R}, \delta) \neq \emptyset$.*

Proof. (outline) Let s, $s_{\downarrow}$ and $s_{\downarrow\downarrow}$ be the last three segments of $\mathbf{R}$. If $\mathbf{R}$ does not contain an l-detached segment then a shift move on $s_{\downarrow}$ is always defined. Otherwise, if s (resp., $s_{\downarrow}$) is l-detached then a shift move on s (resp., $s_{\downarrow}$) is defined. Lastly, if $s_{\downarrow\downarrow}$ is l-detached then s can always be shifted. □

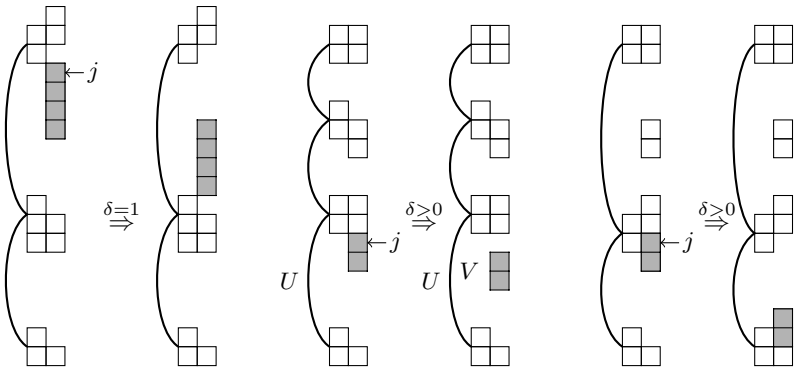


Fig. 3. Examples of shift (left) and split moves (center and right).

Remark that in general several moves are possible in $\mathbf{R}$. However, we are interested in the one occurring in the lowest position. This leads to the following:

Definition 9 (Grand ancestor). *Let* $\mathbf{R}$ *be a region of* $\mathbf{b}$ *and suppose that* $j = \min(Mcr(P_{<i}|\mathbf{b}, \mathbf{R}, \delta))$ *is defined. The* grand ancestor $\mathsf{GA}(P_{<i}|\mathbf{b}, \mathbf{R}, \delta)$ *of* $\mathbf{R}$ *is a region* $\mathbf{G}$ *delimited as* $\mathbf{R}$ *and such that* $A(\mathbf{G}) = A(\mathbf{R})$, $\mathbf{G}$ *is identical to* $\mathbf{R}$ *in the positions above* $j-1$, $\mathbf{G}$ *admits a move at* j *and for all other regions* $\mathbf{R}'$ *satisfying the previous properties, it holds* $\mathbf{G} \prec \mathbf{R}'$ *(possibly,* $\mathbf{G} = \mathbf{R}'$*).*

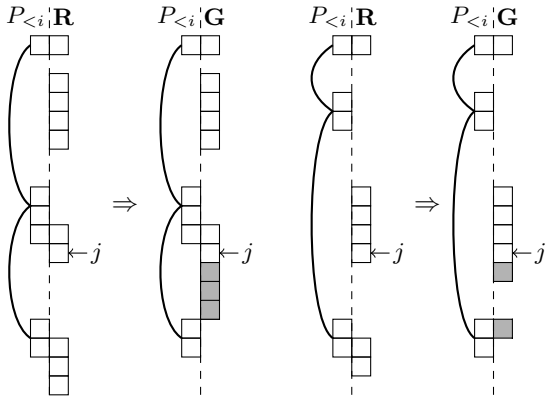


Fig. 4. Examples of grand ancestor (gray cells are those redeployed, $\delta = 4$).

The lowest position where a move may occur in a segment s is characterized in the following Lemma.

Lemma 2. *Let* s *be a segment of a region* $\mathbf{R}$ *of* $\mathbf{b}$ *and let* j *be the lowest position of* s *such that* $j \in Mcr(P_{<i}|\mathbf{b}, \mathbf{R}, \delta)$*. Then,*

$$j = \begin{cases} Bot(s) \ \textit{or}\ 1 + \min(LAdj(s)) \ \textit{or}\ Top(s) & \textit{if } \delta = 0, \\ Bot(s) \ \textit{or}\ 1 + Bot(s) & \textit{if } \delta > 0 \textit{ and } s \textit{ is l-adjacent}, \\ Bot(s) \ \textit{or}\ Top(s) & \textit{if } \delta > 0 \textit{ and } s \textit{ is l-detached}, \end{cases}$$

We point out that the data structure described in Sect. 4.1 allows (in time $O(1)$) both to find the lowest position j where a move may occur in a segment s, and to

make the move at j. Furthermore, the grand ancestor can be effectively computed by exploiting Lemmas 1 and 2. Indeed, one can easily prove the following:

Lemma 3. *Given a region* $\mathbf{R} = (s_1, \ldots, s_m)$ *of* $\mathbf{b}$*, let* s_h *be the segment with a cell in position* $j = \min(Mcr(P_{<i}|\mathbf{b}, \mathbf{R}, \delta))$*. Then,* $\mathsf{GA}(P_{<i}|\mathbf{b}, R, \delta) = (s_1, \ldots, s_{h-1}) \cdot (s', s'_{\downarrow})$ *where* $Top(s') = Top(s_h)$ *and*

$$\begin{cases} A(s') = A(s_h) + w - 1, A\Big(s'_{\downarrow}\Big) = 1, Top\Big(s'_{\downarrow}\Big) = Top({\downarrow}s_h) & \text{if } G, \\ A(s') = A(s_h) + w, s'_{\downarrow} = \epsilon & \text{otherwise,} \end{cases}$$

where $w = \sum_{k=h+1}^{m} A(s_k)$ *and* G *holds if and only if* s_h *is l-detached,* $d(s_{h-1}, s_h) > \delta$ *and* $d(s_h, v) - w > \delta$*, where* v *is the segment below* s_m *(if any).*

Given two regions $\mathbf{R}$ and $\mathbf{R}'$, write $\mathbf{R} \Rightarrow \mathbf{R}'$ if and only if $\mathbf{R}'$ is obtained by making a move in $\mathbf{R}$. Moreover, $\mathbf{R} \overset{j}{\Rightarrow} \mathbf{R}'$ if $j \in \mathrm{Mcr}(P_{<i}|\mathbf{b}, \mathbf{R}, \delta)$ and the move at the position j of $\mathbf{R}$ produces the region $\mathbf{R}'$.

At this point we are ready to define a discrete dynamical system $f_{P_{<i}|\mathbf{b},\delta}$ that takes in input a region $\mathbf{R}$ of $\mathbf{b}$ and changes its content, producing a region $\mathbf{R}'$ such that $\mathbf{R} \prec \mathbf{R}'$ unless the input region contains no moves.

Definition 10 (The dynamical system over regions). *Given a region* $\mathbf{R}$ *of* $\mathbf{b}$*, let* $j = \min(Mcr(P_{<i}|\mathbf{b}, \mathbf{R}, \delta))$ *and define*

$$f_{P_{<i}|\mathbf{b},\delta}(\mathbf{R}) = \begin{cases} \mathbf{R}' & \text{if } j \neq \perp \text{ and } \mathsf{GA}(P_{<i}|\mathbf{b}, \mathbf{R}, \delta) \overset{j}{\Rightarrow} \mathbf{R}' \\ \mathbf{R} & \text{otherwise.} \end{cases}$$

Let $\mathbf{R} = (s_1, \ldots, s_m)$ be a region of a column $\mathbf{b}$. Then, $RC(\mathrm{Ske}(\mathbf{b}), \mathbf{R}, \delta)$ denotes the set of all sequences $\mathbf{G}$ of segments, with $\mathrm{A}(\mathbf{G}) = \mathrm{A}(\mathbf{R})$, such that one can replace $\mathbf{R}$ with $\mathbf{G}$ obtaining a different column $\mathbf{b}'$ with $\mathbf{b}' \square P_{i-1}$. Remark that $RC(\mathrm{Ske}(\mathbf{b}), \mathbf{R}, \delta)$ is finite and totally ordered by $\prec$. We call *initial region* its minimum element, denoted $\mathbf{R}_{\min}$. Indeed, $\mathbf{R}_{\min}$ is delimited by u and v, where $u = s_1^{\uparrow}$ and $v^{\uparrow} = s_m$ (they both can be null), and it contains at most two segments s' and $s'_{\downarrow}$ with

$$\begin{array}{l} \mathrm{Top}(s') = \mathrm{Bot}(u) - 2 \text{ if } u \neq \epsilon \wedge (\mathrm{Bot}({\downarrow}u) < \mathrm{Bot}(u) - 1 \vee w > d(u, ({\downarrow}u)_{\downarrow}) \vee \delta > 0) \\ \mathrm{Bot}(s') = \mathrm{Top}(({\downarrow}u)_{\downarrow}) \text{ if } u \neq \epsilon \wedge (w \leq d(u, ({\downarrow}u)_{\downarrow}) \wedge \mathrm{Bot}({\downarrow}u) \geq \mathrm{Bot}(u) - 1 \wedge \delta = 0) \\ \mathrm{Bot}(s') = f \qquad \text{if } u = \epsilon \wedge \delta = 0 \\ \mathrm{Bot}(s') = f + \delta + 1 \text{ if } u = \epsilon \wedge \delta > 0 \end{array} \tag{1}$$

where $w = \sum_{k=1}^{m} \mathrm{A}(s_k)$ and f is the position of the first segment of P_{i-1}. Moreover, $\mathrm{A}(s') = w$ and $s'_{\downarrow} = \epsilon$ unless $\delta > 0$ and $u = \epsilon$, in which case one has $\mathrm{A}(s') = w - 1$, $\mathrm{A}\Big(s'_{\downarrow}\Big) = 1$ and $\mathrm{Top}\Big(s'_{\downarrow}\Big) = \mathrm{Top}(z)$. The following lemma proves that $f_{P_{<i}|\mathbf{b},\delta}$ starts in an initial region and eventually ends.

Lemma 4. *Consider an i-prefix* $P_{<i}|\mathbf{b}$ *of a polyomino in* $\mathsf{Pol}_2(n)$ *and a region* $\mathbf{R}$ *of* $\mathbf{b}$*,* $\mathbf{b} = \mathbf{b}' \cdot \mathbf{R} \cdot \mathbf{b}''$*. Then, one has*

1. *for all* $\mathbf{a} \in RC(Ske(\mathbf{b}), \mathbf{R}, \delta)$ *such that* $Mcr(P_{<i}|\mathbf{b}' \cdot \mathbf{a} \cdot \mathbf{b}'', \mathbf{a}, \delta) \neq \emptyset$ *it holds* $\mathbf{a} \prec f_{P_{<i}|\mathbf{b},\delta}(\mathbf{a})$*;*
2. $RC(Ske(\mathbf{b}), \mathbf{R}, \delta) = \bigcup_{n\in\mathbb{N}} f^n_{P_{<i}|\mathbf{b},\delta}(\mathbf{R}_{min})$*, where* $f^n_{P_{<i}|\mathbf{b},\delta}(\mathbf{R}_{min})$ *corresponds to* $f_{P_{<i}|\mathbf{b},\delta}(f^{n-1}_{P_{<i}|\mathbf{b},\delta}(\mathbf{R}_{min}))$*, with* $f^0_{P_{<i}|\mathbf{b},\delta}(\mathbf{R}_{min}) = \mathbf{R}_{min}$*.*

In other words, the previous lemma grants that if one wants to exhaustively generate all possible regions, it is enough to effectively simulate the dynamical system $f_{P_{<i}|\mathbf{b},\delta}$. There will be neither missing regions nor repetitions.

4 Exhaustive generation of $\mathsf{Pol}_2(n)$

An inductive approach is used. Given $i, k \in \mathbb{N}$ and a skeleton $\mathbf{b} \in \mathrm{Sk}(P_{<i}, k, n)$ (where $P_{<i}$ is the current prefix), suppose that all the polyominoes $Q \in \mathsf{Pol}_2(n)$ with $Q_{\leq i} = P_{<i}|\mathbf{c}$ have been already generated, for all columns $\mathbf{c}$ such that $\mathrm{A}(\mathbf{c}) = k$ and $\mathrm{Ske}(\mathbf{c}) = \mathbf{b}'$, where $\mathbf{b}' \in \mathrm{Sk}(P_{<i}, k, n)$ and $\mathbf{b}' \lhd \mathbf{b}$. We proceed by generating all the columns $\mathbf{c} \in \mathrm{Comp}(P_{<i}, k, n) \cap [\mathbf{b}]_\diamond$ and then, for each $\mathbf{c}$, we (recursively) generate all the polyominoes $Q \in \mathsf{Pol}_2(n)$ with $Q_{\leq i} = P_{<i}|\mathbf{c}$. Thus, the problem is reduced to design an efficient algorithm for generating the set $\mathrm{Comp}(P_{<i}, k, n)$. To this aim, we propose an algorithm that works in two steps:

1. an outer loop that generates all skeletons $\mathbf{b}$ in $\mathrm{Sk}(P_{<i}, k, n)$;
2. an inner loop that, for each $\mathbf{b}$, generates all colums $\mathbf{c} \in \mathrm{Comp}(P_{<i}, k, n) \cap [\mathbf{b}]_\diamond$.

The algorithm is a smart combination of two main iterators (functions that run through ordered sequences), one for skeletons and one for columns. The pseudocode of all iterators and functions will appear in the long version of this paper. Here we briefly describe how they works.

The iterator NextSkel$(P_{<i}, k, \mathbf{b}, \delta)$ takes $P_{<i}$, an integer k, a skeleton $\mathbf{b} \in \mathrm{Sk}(P_{<i}, k, \delta)$ and δ, and returns the smallest skeleton larger than $\mathbf{b}$ in $\mathrm{Sk}(P_{<i}, k, \delta)$. The function is based on Def. 7 and exploits different iterators for different types of skeletons. More precisely, the first skeleton $\mathbf{b}_{\text{init}}$ is either null (if $P_{<i} \in \mathsf{Pol}$), or it consists of a suitable bridge $\mathbf{b}_{\text{init}} = (\bar{s})$ with $\mathrm{A}(\bar{s}) = k$ (if $P_{<i}$ consists of two polyominoes U and V with $\mathrm{Dist}(U, V) + 2 < k$), or it consists of two suitable extremes, $\mathbf{b}_{\text{init}} = (s, t)$ with $\mathrm{A}(s) = k - 1$ and $\mathrm{A}(t) = 1$ (if $\mathrm{Dist}(U, V) + 2 > k$). If $\mathbf{b}_{\text{init}} = ()$ then, for any $k \geq 0$, one has $\mathrm{Sk}(P_{<i}, k, \delta) = \{()\}$ and we are done. Otherwise, the successor $\mathbf{b}'$ of a non-null skeleton $\mathbf{b}$ depends on the number of its segments and on their types - either bridges or extremes, see Def. 4. Indeed, if $\mathbf{b} = (\bar{s})$ then $\mathbf{b}'$ is obtained by using an iterator NextBridge that works as follows. If $\mathbf{b}$ is not the last skeleton consisting of one bridge, then $\mathbf{b}'$ is obtained either by shifting $\bar{s}$ (if after a shift by e positions downwards $\bar{s}$ remains a bridge – e is always 1 if P_{i-1} contains only two extremes), or by replacing $\bar{s}$ with a smaller bridge $\bar{t}$, with $\mathrm{A}(\bar{t}) = \mathrm{A}(\bar{s}) - 1$, placed in the highest possible position (*i.e.* $\bar{t}$ is l-adjacent to the first two extremes e and $e_\downarrow$ in P_{i-1} with $d(e, e_\downarrow) + 2 \leq \mathrm{A}(\bar{t})$). Otherwise, $\mathbf{b}$ is the last element in the ordered sequence of skeletons consisting of one bridge, and $\mathbf{b}'$ should be the smallest skeleton consisting of two bridges of total area k. A function InitBridges is used for this purpose. Otherwise, one

can not place two bridges of total area k in column i (because the number of cells k is not sufficient or P_{i-1} has only two extremes or it has three extremes e_1, e_2 and e_3 with $\mathrm{A}(e_2) = 2$), and then $\mathbf{b}'$ should be the smallest skeleton consisting of two l-adjacent extremes of total area k. Similarly, if $\mathbf{b} = (s, t)$ then $\mathbf{b}'$ is obtained either by using an iterator NextTwoExtremes (if $\mathbf{b}$ is not the greatest skeleton consisting of two extremes), or a function InitThreeExtremes.

NextTwoExtremes either shifts t, or it shifts s and places t in the highest possible position below s (if t can not be shifted), or it constructs the smallest skeleton comprising two extremes s' and t' with $\mathrm{A}(s') = \mathrm{A}(s) - 1$ and $\mathrm{A}(t') = \mathrm{A}(t) + 1$ (if neither t nor s can be shifted), or it constructs the smallest skeleton of area $k-1$ that consists of two extremes (if $\mathrm{A}(s') = 1$). Otherwise, $\mathbf{b}$ is greatest skeleton consisting of two extremes and $\mathbf{b}'$ is the smallest skeleton with three l-adjacent extremes of total area k. The remaining cases $\mathbf{b} = (\bar{s}, \bar{t})$, $\mathbf{b} = (s, t, u)$ and $\mathbf{b} = (s, t, u, v)$ are treated similarly.

In addition to the previous iterators, an iterator Next is used to run through a sequence of regions. $\textsc{Next}(\mathbf{R}, \delta)$ takes a region $\mathbf{R}$ and returns a region $\mathbf{R}'$ obtained according to the dynamical system provided in Def. 10. Furthermore, let $\textsc{Init}(\mathrm{Ske}(\mathbf{b}), r, m, \delta)$ be the function that takes a skeleton, the type r of the region (northern, central, southern), an integer m and δ, and returns the smallest region of type r and area m, see the characterization (1) for $\mathbf{R}_{\min}$.

The iterator $\textsc{NextCol}(P_{<i}, \mathbf{c}, \delta, n)$ takes a column $\mathbf{c} \in \mathrm{Comp}(P_{<i}, k, n)$ and δ, and returns the smallest column $\mathbf{c}'$ in $\mathrm{Comp}(P_{<i}, k, n)$ such that $\mathbf{c} < \mathbf{c}'$ and $\mathrm{Ske}(\mathbf{c}) = \mathrm{Ske}(\mathbf{c}')$. NextCol works by distinguishing six cases, depending on $\mathrm{Ske}(\mathbf{c})$. Let $\mathrm{DF}(\mathbf{b}) = 1$ if the column $\mathbf{b}$ is l-detached free, 0 otherwise. The simplest case is when $\mathrm{Ske}(\mathbf{c})$ is null, hence $\mathbf{c} = \mathbf{c}^{\uparrow}$. Here, the next column is simply $\textsc{Next}(\mathbf{c}, \delta)$. Now, consider the case $\mathrm{Ske}(\mathbf{c}) = (\bar{s})$ where $\mathbf{c}$ has at most two regions (the case $\mathrm{Ske}(\mathbf{c}) = (s, t, u)$ is treated in the same way). One has $\mathbf{c} = \mathbf{c}^{\uparrow} \cdot (\bar{s}) \cdot \mathbf{c}_{\downarrow}$. So, if $\mathbf{c}^{\uparrow} = \epsilon$ then $\mathbf{c}'$ is either $(\bar{s}) \cdot \mathbf{b}$ where $\mathbf{b} = \textsc{Next}(\mathbf{c}_{\downarrow}, \delta)$ (if $\mathbf{b} \neq \perp$), or $\perp$ (if $\mathbf{b} = \perp$). Otherwise, one has $\mathbf{c}^{\uparrow} \neq \epsilon$ and $\mathbf{c}'$ is either $\mathbf{c}^{\uparrow} \cdot (\bar{s}) \cdot \mathbf{e}'$ (if $\mathbf{e}' = \textsc{Next}(\mathbf{c}_{\downarrow}, \mathrm{DF}(\mathbf{c}^{\uparrow}) \cdot \delta) \neq \perp$) or $\mathbf{e} \cdot (\bar{s}) \cdot \mathbf{e}''$ where $\mathbf{e} = \textsc{Next}(\mathbf{c}^{\uparrow}, \delta)$) and $\mathbf{e}'' = \textsc{Init}(P_{<i}, \mathrm{south}, \mathrm{A}(\mathbf{c}_{\downarrow}), \mathrm{Ske}(\mathbf{c}), \mathrm{DF}(\mathbf{e}) \cdot \delta)$ (if $e' = \perp$ and $e \neq \perp$), or (if $e = e' = \perp$) $\textsc{Init}(P_{<i}, \mathrm{north}, \mathrm{A}(\mathbf{c}^{\uparrow}) - 1, \mathrm{Ske}(\mathbf{c}), \delta) \cdot (\bar{s}) \cdot \textsc{Init}(P_{<i}, \mathrm{south}, \mathrm{A}(\mathbf{c}_{\downarrow}) + 1, \mathrm{Ske}(\mathbf{c}), 0)$. NextCol works similarly also in the cases where three regions can be present (skeletons of type $(\bar{s}, \bar{t})$ or (s, t, u, v)). Lastly, if $\textsc{NextCol}(P_{<i}, \mathbf{c}, \delta, n) = \perp$ then $\mathbf{c}'$ consists of the smallest skeleton larger than $\mathrm{Ske}(\mathbf{c})$, with $\mathrm{A}(\mathbf{c}') = k$, which is returned by $\textsc{NextSkel}(P_{<i}, k, \mathrm{Ske}(\mathbf{c}), \delta)$. The correctness of all the iterators directly follows from their definition and from the order on skeletons and on columns.

4.1 The data structure

We represent $P_{\leq i}$ by an array of i records, one per column. The record for the i-th column is a tuple $(L, U^{\uparrow}, U^{\downarrow}, V^{\uparrow}, V^{\downarrow}, R_1, R_2, R_3)$. The first field L is a doubly-linked list for the sequence of segments representing the column (as many nodes as segments), then we have four links to the nodes associated with the extremes of the column (at least one is non-null). Lastly, we have three links

to the last node of each region (at least one is non-null). Each node of the list L corresponds to a segment s (represented by $(\mathrm{A}(s), \mathrm{Top}(s))$) and contains five entries (s, l_1, l_2, l_3, l_4), where l_1 is a link to the preceding node (for $s^{\uparrow}$), l_2 is the link to the next node (for $s_{\downarrow}$), l_3 and l_4 are the links to the nodes (in the list representing the preceding column) associated with ${}^{\uparrow}s$ and ${}_{\downarrow}s$, respectively. Given a region $\mathbf{R}$ of P_i, by Lemma 1 the move in the position $\min(\mathrm{Mcr}(P_{\leq i}, \mathbf{R}, \delta))$ regards one of the last three segments of $\mathbf{R}$. Notice that this data structure allows the execution of any move (either Shift or Split) in time $O(1)$ (Lemmas 2 and 3). Figure 5 illustrates the data structure associated with a polyomino.

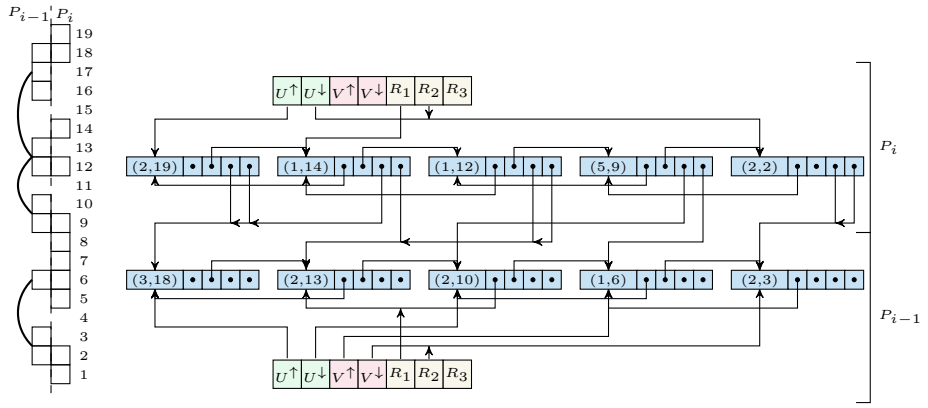


Fig. 5. Two columns (left) and the associated data structure (right).

4.2 Complexity

The execution of the algorithm that generates $\mathsf{Pol}_2(n)$ is represented by a tree $T(n)$ where an internal node v at level i corresponds to a suitable i-prefix $P_{\leq i}$. Furthermore, the children of v correspond to the $(i+1)$-prefixes $P_{\leq i}|\mathbf{c}$, for all $\mathbf{c}$ in $\bigcup_k \mathrm{Comp}(P_{\leq i}, k, n)$. Then, the *complexity* of v, denoted by $C(v)$, is the time taken to generate the set $\bigcup_k \mathrm{Comp}(P_{\leq i}, k, n)$ (as many columns as children of v), with $C(v) = O(1)$ if v is a leaf. Since $f(n) = O(g(n))$ and $g(n) = |\mathsf{Pol}_2(n)|$, where $f(n)$ (resp., $g(n)$) is the number of internal nodes (resp., leaves) of $T(n)$, it follows that the algorithm is CAT if one proves that $C(T(n)) = O(|T(n)|)$, where $C(T(n)) = \sum_{v \in T(n)} C(v)$. Indeed, it holds the following result.

Theorem 1. $C(T(n)) = O(|T(n)|)$.

5 Conclusions

This paper further deepens the new approach to polyominoes generation started in [7], by presenting a new CAT generation algorithm for $\mathsf{Pol}_2(n)$, which takes linear space. We strongly believe that all conceptual obstacles have been detected and that our approach might lead to a CAT algorithm working in linear space for the generation of the full set $\mathsf{Pol}(n)$.

References

1. Bousquet-Mélou, M.: A method for the enumeration of various classes of column-convex polygons. Discrete Math. 154(1-3), 1–25 (1996)
2. Brocchi, S., Castiglione, G., Massazza, P.: On the exhaustive generation of k-convex polyominoes. Theor. Comput. Sci. 664, 54 – 66 (2017)
3. Castiglione, G., Massazza, P.: An efficient algorithm for the generation of Z-convex polyominoes. In: IWCIA 2014. Lecture Notes in Comput. Sci., vol. 8466, pp. 51–61. Springer (2014)
4. Castiglione, G., Restivo, A.: Reconstruction of L-convex polyominoes. Electron. Notes Discrete Math. 12, 290–301 (2003)
5. Del Lungo, A., Duchi, E., Frosini, A., Rinaldi, S.: On the generation and enumeration of some classes of convex polyominoes. Electron. J. Comb. 11(1) (2004)
6. Delest, M.P., Viennot, G.: Algebraic languages and polyominoes enumeration. Theor. Comput. Sci. 34(1-2), 169–206 (1984)
7. Formenti, E., Massazza, P.: From tetris to polyominoes generation. Electronic Notes in Discrete Mathematics 59 (2017)
8. Golomb, S.W.: Checker boards and polyominoes. Amer. Math. Monthly 61, 675–682 (1954)
9. Jensen, I.: Enumerations of lattice animals and trees. Journal of Statistical Physics 102(3), 865–881 (2001)
10. Jensen, I.: Counting polyominoes: A parallel implementation for cluster computing. In: ICCS 2003. Lecture Notes in Comput. Sci., vol. 2659, pp. 203–212. Springer-Verlag (2003)
11. Mantaci, R., Massazza, P.: On the exhaustive generation of plane partitions. Theor. Comput. Sci. 502, 153–164 (Sep 2013)
12. Mantaci, R., Massazza, P.: From linear partitions to parallelogram polyominoes. In: DLT11. Lecture Notes in Comput. Sci., vol. 6795, pp. 350–361. Springer (2011)
13. Mantaci, R., Massazza, P., Yunès, J.B.: An efficient algorithm for generating symmetric ice piles. Theor. Comput. Sci. 629(C), 96–115 (May 2016)
14. Massazza, P.: On the generation of convex polyominoes. Discrete Appl. Math. 183, 78–89 (2015)
15. Privman, V., Barma, M.: Radii of gyration of fully and partially directed lattice animals. Zeitschrift für Physik B Condensed Matter 57(1), 59–63 (1984)
16. Privman, V., Forgacs, G.: Exact solution of the partially directed compact lattice animal model. Journal of Physics A: Mathematical and General 20(8), L543 (1987)
17. Redner, S., Yang, Z.R.: Size and shape of directed lattice animals. Journal of Physics A: Mathematical and General 15(4), L177 (1982)