On the Generation of L-Convex Polyominoes

Paolo Massazza[1]

[1]Dipartimento di Scienze Teoriche e Applicate
Università degli Studi dell'Insubria
Varese Italy

GASCom 2012
Bordeaux, June 26th 2012

Abstract

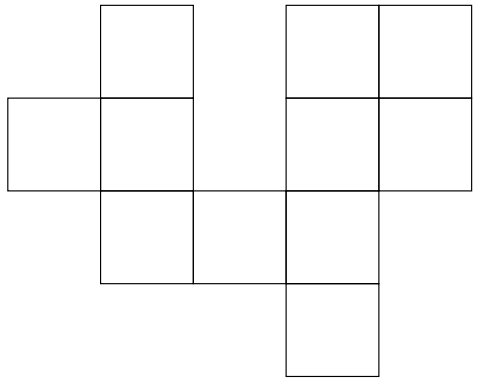
We consider the problem of the exhaustive generation of L-convex polyominoes by efficient algorithms.

We show a bijection between the set of L-convex polyominoes of size n and a suitable set of pairs of integer sequences.

This lets us design an algorithm which generates all L-convex polyominoes of size n in constant amortized time using $O(\sqrt{n})$ space

Polyominoes

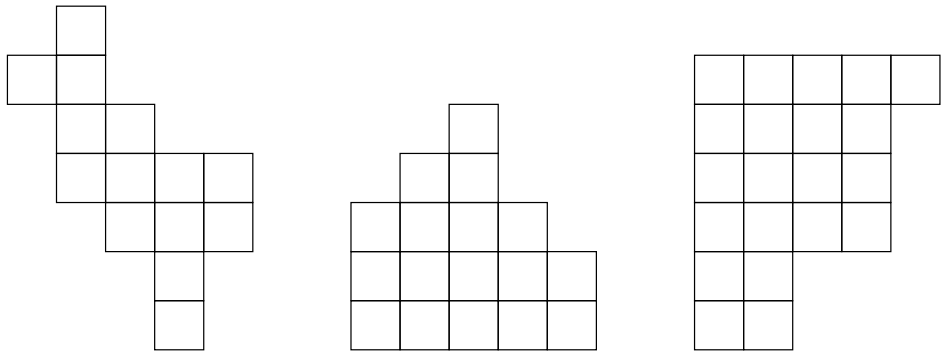
A polyomino is a finite connected set of edge-to-edge adjacent 1×1 cells in the $2D$ plane



- introduced by Golomb (1954)
- widely studied in
  - enumerative combinatorics
  - bijective combinatorics
  - two dimensional language theory
  - tiling theory

Convex Polyominoes

A convex polyomino is a polyomino such that the intersection with an infinite row or column is a connected set of cells



(a) (b) (c)

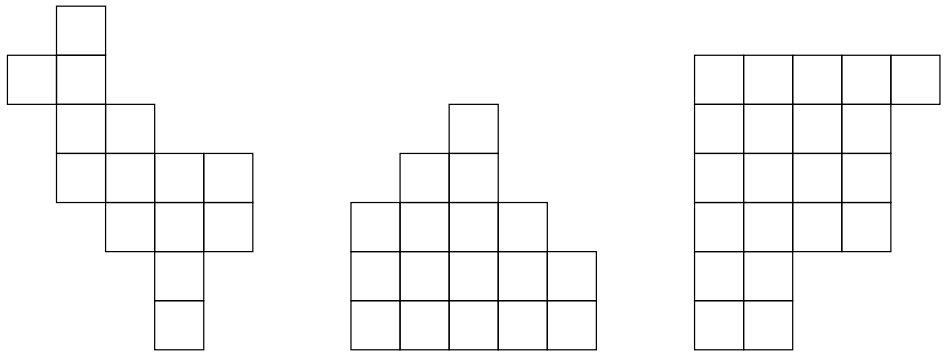
(a) convex polyomino

(b) stack polyomino

(c) Ferrer diagram

Convex Polyominoes

A convex polyomino is a polyomino such that the intersection with an infinite row or column is a connected set of cells



(a) (b) (c)

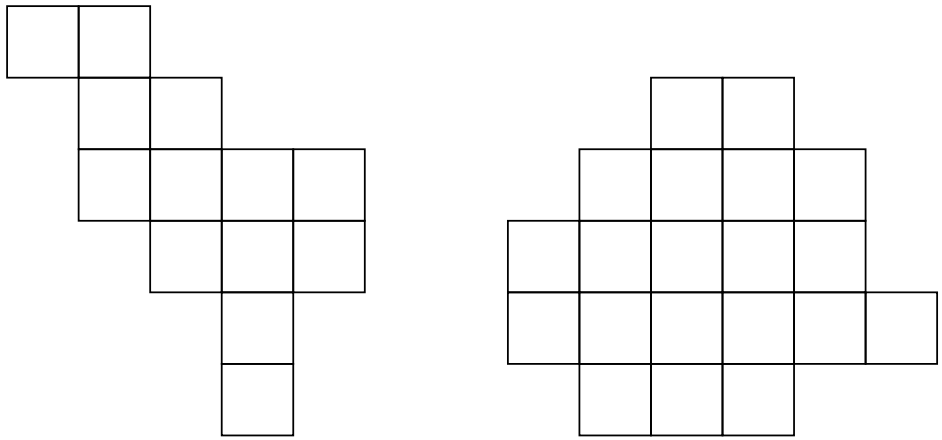
(a) convex polyomino

(b) stack polyomino

(c) Ferrer diagram

L-Convex Polyominoes

An L-convex polyomino is a convex polyomino such that there is a path with at most one change of direction between any two cells



not L–convex L–convex

- introduced by Castiglione and Restivo (2003)
- enumerated w.r.t. rows and columns by Castiglione et al. (2005)
- studied as 2D languages by Brocchi et al. (2009)

L-convex Polyominoes: the problem

Problem (exhaustive generation)

Given n, generate $LPol(n)$

We split the problem in two subproblems:

1. find a bijection between LPol(n) and a set T of suitable pairs of integer sequences
2. design a CAT algorithm for T

Convex Polyominoes: representation

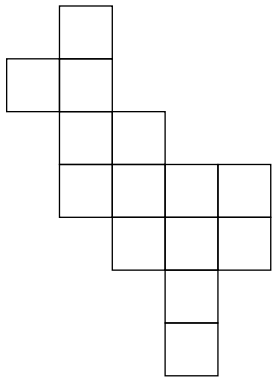
A convex polyomino is univocally determined by

- the vertical projection

$$\pi(P) = (\pi_1, \ldots, \pi_l), \quad \pi_i = \sharp\{j|(i,j) \in P\}$$

- the position vector

$$\sigma(P) = (\sigma_1, \ldots, \sigma_l), \quad \sigma_i = \min\{j|(i,j) \in P\}$$



π(P)=(1,4,3,4,2)

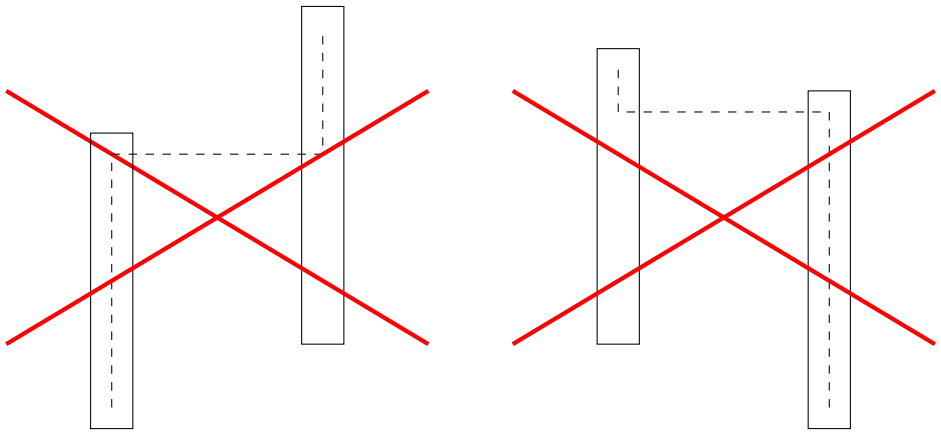
σ(P)=(6,4,3,1,3)

L-convex Polyominoes: the bijection

Theorem

Let $v, p \in \mathbb{N}^*$, *with* $|v| = |p|$, $\min(p) = 1$. *Then* (v, p) *individuates* $P \in LPol(n)$ *with* $\pi(P) = v$ *and* $\sigma(P) = p$, *if and only if*

1. v *is an unimodal integer sequence of* n;
2. *for all* i, j *with* $1 \leq i < j \leq |v|$, *either*
   $p_j \leq p_i < p_i + v_i - 1 \leq p_j + v_j - 1$ *or*
   $p_i \leq p_j < p_j + v_j - 1 \leq p_i + v_i - 1$.



A Recipe for generating LPol(n): ingredients

Let $s = (s_1, \ldots, s_l) \in \mathbb{N}^l$ and $\sum s_i = n$. Then

- $s \in \mathrm{LP}(n)$ iff $s_i \geq s_{i+1}$;
- $s \in \mathrm{US}(n)$ iff $s_1 \leq \ldots \leq s_i \geq \ldots \geq s_l$.

Definition

Let $v \in \mathrm{US}(n)$ and $l = |v|$. The set of sequences in $\mathbb{N}^l$ which are LPol-feasible with v is

$$F_v = \{p \in \mathbb{N}^l \,|\, \min(p) = 1, (v, p) \text{ identifies } P \in \mathrm{LPol}(n)\}$$

Recipe

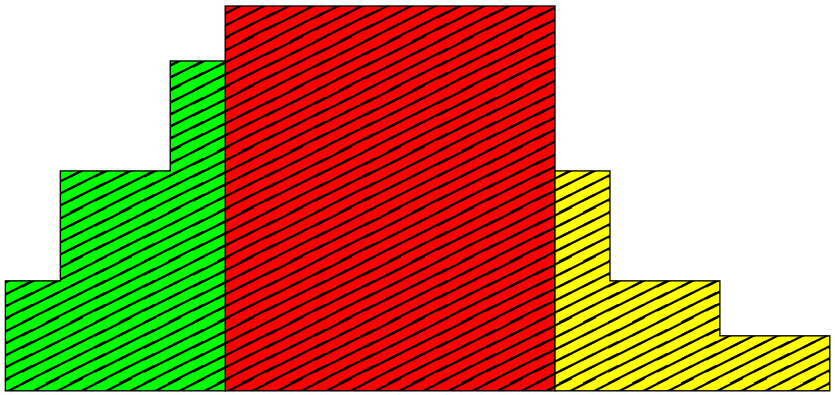
We obtain a CAT algorithm for LPol(n) by combining:

- A CAT algorithm for US(n) (stack polyominoes)
- A CAT algorithm for F_v

A CAT algorithm for US(n)

By decomponing a stack polyomino t into 2 Ferrer diagrams and 1 rectangle

$$t = t_{<i} \cdot t_i^{[k]} \cdot t_{\geq i+k}, \quad t_{i-1}, t_{i+k} < t_i$$



we get a CAT algorithm which generates US(n) by combining

- A CAT iterator for rectangles (trivial)
- A CAT iterator for Ferrer diagrams of bounded height

An iterator for $\mathrm{LP}^h(m)$

The idea...

- Let $\mathrm{LP}^h(m)$ be equipped with $<_{\mathrm{nlex}}$,

$$t^{(1)} = \min{}_{<_{\mathrm{nlex}}}(\mathrm{LP}^h(m)) = h^{[a]} \cdot b$$

- define an iterator Next : $\mathrm{LP}^h(m) \mapsto \mathrm{LP}^h(m) \cup \{\perp\}$, s.t.

$$\forall e > 0, \quad \mathrm{Next}(t^{(e)}) \to t^{(e+1)};$$

- run through the ordered sequence

$$t^{(1)} = h^{[a]} \cdot b <_{\mathrm{nlex}} t^{(2)} <_{\mathrm{nlex}} \cdots <_{\mathrm{nlex}} t^{(p)} = 1^{[m]}$$

by calling Next $|\mathrm{LP}^h(m)| - 1$ times.

An iterator for $\mathrm{LP}^h(m)$

Fact

There is an implementation of Next which runs in time $O(1)$

Proof (outline).

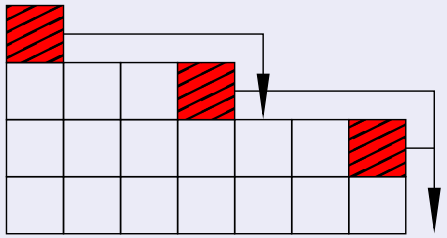
An iterator for $\mathrm{LP}^h(m)$

Fact

There is an implementation of Next which runs in time $O(1)$

Proof (outline).

1) Define a (granular) discrete dynamical system with state space $\mathrm{LP}^h(m)$



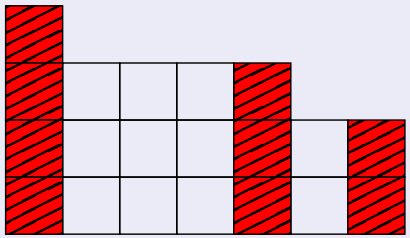
An iterator for $\mathrm{LP}^h(m)$

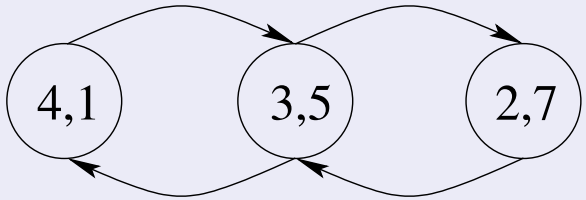
Fact

There is an implementation of Next which runs in time $O(1)$

Proof (outline).

2) Represent $t \in \mathrm{LP}^h(m)$ by a suitable doubly-linked list





An iterator for $\mathrm{LP}^h(m)$

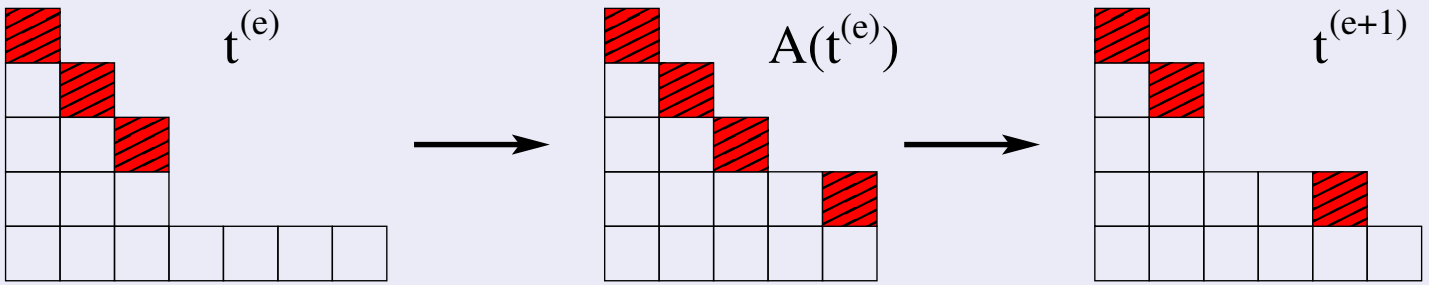
Fact

There is an implementation of Next which runs in time $O(1)$

Proof (outline).

3) Follow the technique for generating Ice piles (Massazza, Radicioni 2010)

$$\mathrm{A}(t^{(e)}) \stackrel{i_e}{\Rightarrow} t^{(e+1)}, \qquad \text{with } i_e = \text{ rigthmost move in } t^{(e)}$$



A CAT algorithm for F_v

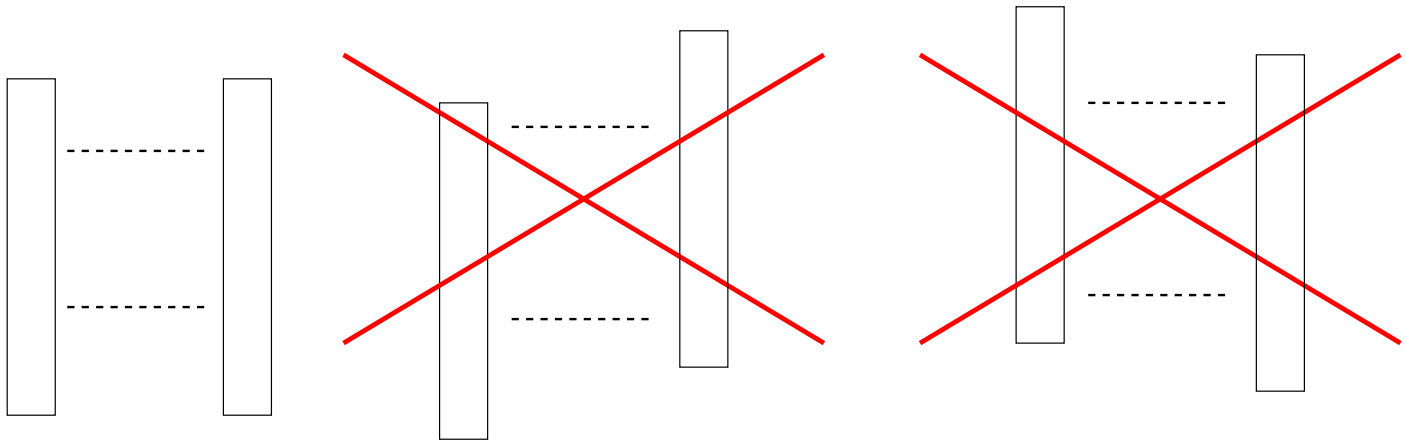
Consider v in decreasing order, $v_{i_1} \geq v_{i_2} \geq \ldots \geq v_{i_l}$, and note that

A CAT algorithm for F_v

Consider v in decreasing order, $v_{i_1} \geq v_{i_2} \geq \ldots \geq v_{i_l}$, and note that

1) columns with the same height must have the same position

$$v_{i_q} = v_{i_r} \quad \Rightarrow \quad p_{i_q} = p_{i_r}$$

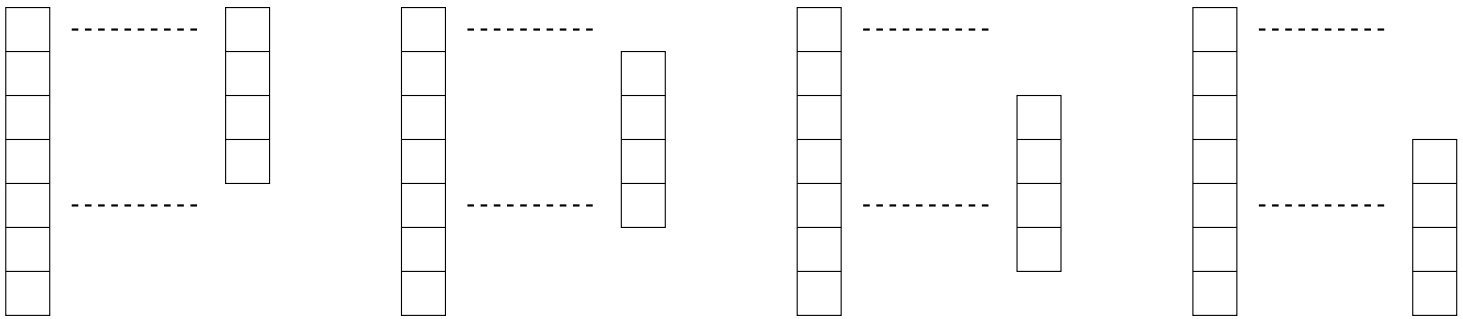


A CAT algorithm for F_v

Consider v in decreasing order, $v_{i_1} \geq v_{i_2} \geq \ldots \geq v_{i_l}$, and note that

2) the position of a column of height $v_{i_{r+1}}$ depends only on the position of a column of height v_{i_r}

$$p_{i_r} \leq p_{i_{r+1}} \leq p_{i_r} + v_{i_r} - v_{i_{r+1}}$$



A CAT algorithm for F_v

Let r different values occur in v, $v_{i_1} > v_{i_2} > \ldots > v_{i_r}$. We can easily define a recursive algorithm for F_v.

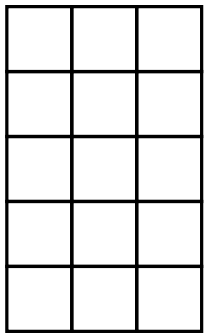
```
PROCEDURE POSITION(j)
if j = 1 then
   p_{i_1} := 1;
   if j < r then
      POSITION(j + 1);
   end if
else
   for m:=p_{i_{j-1}} + v_{i_{j-1}} - v_{i_j} downto p_{i_{j-1}} do
      p_{i_j} :=m;
      if j < r then
         POSITION(j + 1);
      end if
   end for
end if
```

A CAT algorithm for F_v: example

Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)

A CAT algorithm for F_v: example

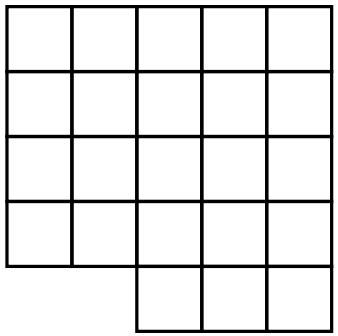
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$1^{[3]}$

A CAT algorithm for F_v: example

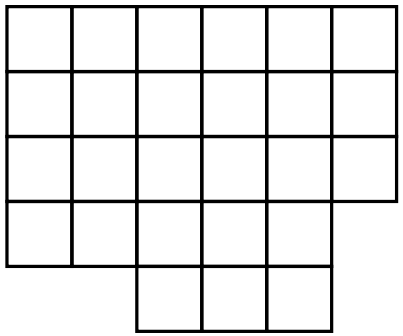
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$2^{[2]} \cdot 1^{[3]}$

A CAT algorithm for F_v: example

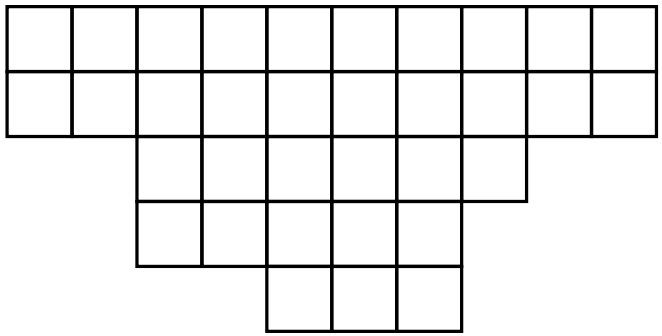
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$2^{[2]} \cdot 1^{[3]} \cdot 3$

A CAT algorithm for F_v: example

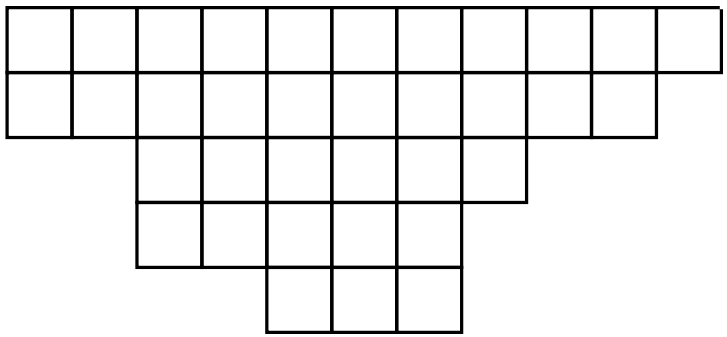
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$4^{[2]} \cdot 2^{[2]} \cdot 1^{[3]} \cdot 3 \cdot 4^{[2]}$

A CAT algorithm for F_v: example

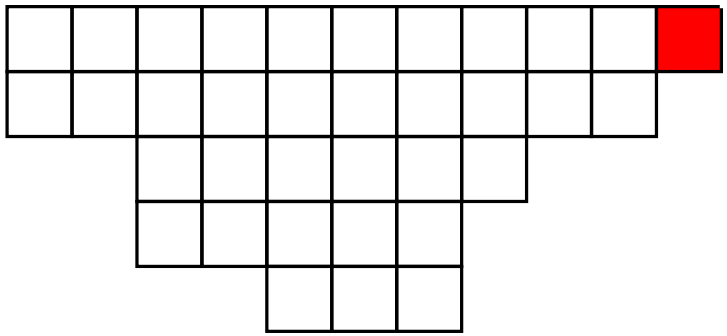
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$4^{[2]} \cdot 2^{[2]} \cdot 1^{[3]} \cdot 3 \cdot 4^{[2]} \cdot 5$

A CAT algorithm for F_v: example

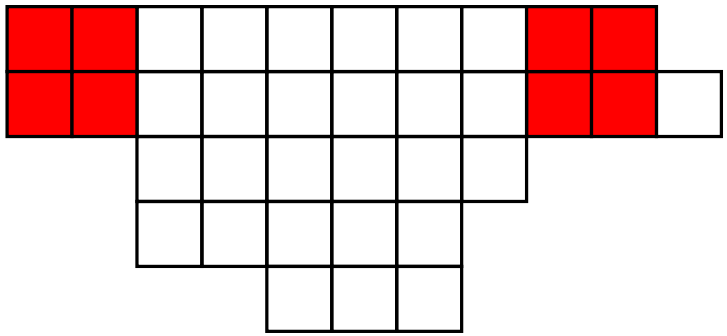
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$4^{[2]} \cdot 2^{[2]} \cdot 1^{[3]} \cdot 3 \cdot 4^{[2]} \cdot 5$

A CAT algorithm for F_v: example

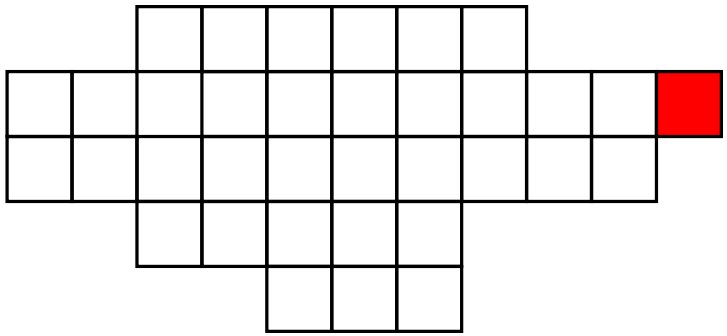
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$4^{[2]} \cdot 2^{[2]} \cdot 1^{[3]} \cdot 3 \cdot 4^{[2]} \cdot 4$

A CAT algorithm for F_v: example

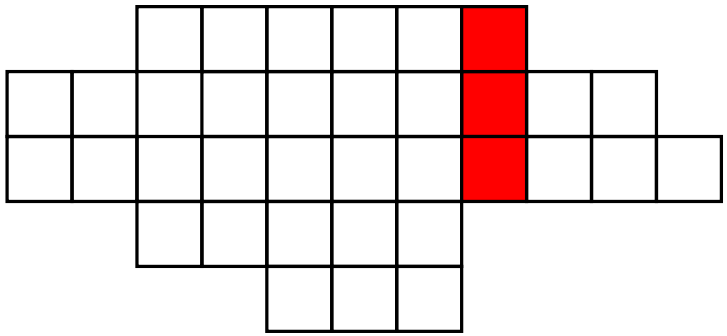
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$3^{[2]} \cdot 2^{[2]} \cdot 1^{[3]} \cdot 3 \cdot 3^{[2]} \cdot 4$

A CAT algorithm for F_v: example

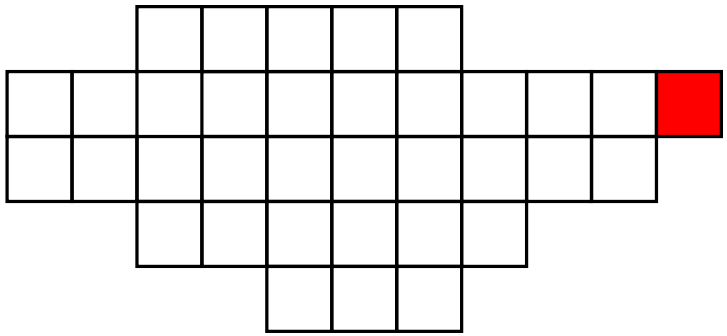
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$3^{[2]} \cdot 2^{[2]} \cdot 1^{[3]} \cdot 3 \cdot 3^{[2]} \cdot 3$

A CAT algorithm for F_v: example

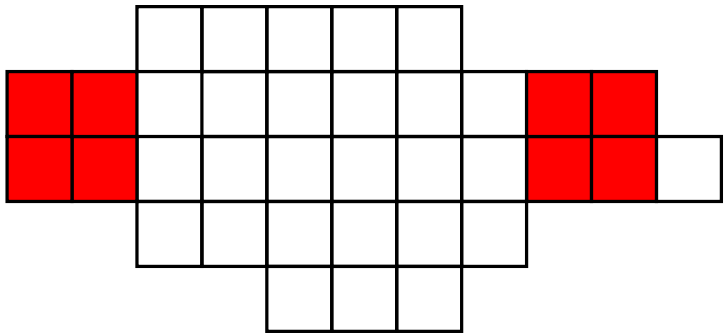
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$3^{[2]} \cdot 2^{[2]} \cdot 1^{[3]} \cdot 2 \cdot 3^{[2]} \cdot 4$

A CAT algorithm for F_v: example

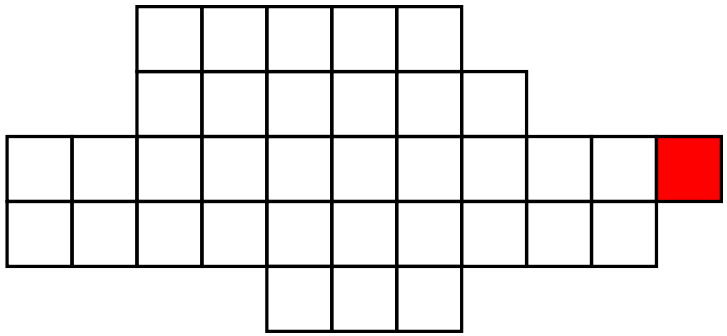
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$3^{[2]} \cdot 2^{[2]} \cdot 1^{[3]} \cdot 2 \cdot 3^{[2]} \cdot 3$

A CAT algorithm for F_v: example

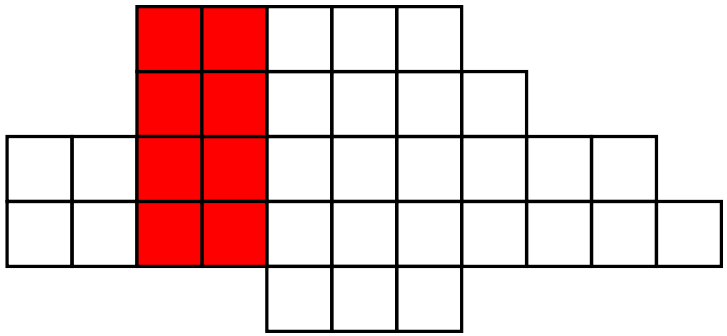
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$2^{[2]} \cdot 2^{[2]} \cdot 1^{[3]} \cdot 2 \cdot 2^{[2]} \cdot 3$

A CAT algorithm for F_v: example

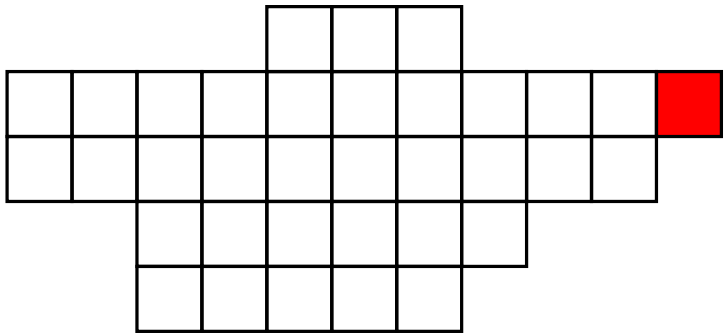
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$2^{[2]} \cdot 2^{[2]} \cdot 1^{[3]} \cdot 2 \cdot 2^{[2]} \cdot 2$

A CAT algorithm for F_v: example

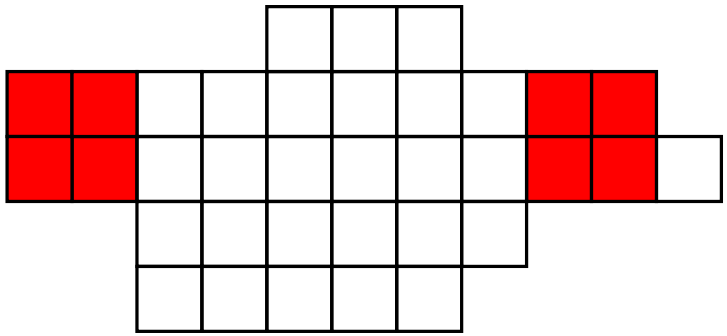
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$3^{[2]} \cdot 1^{[2]} \cdot 1^{[3]} \cdot 2 \cdot 3^{[2]} \cdot 4$

A CAT algorithm for F_v: example

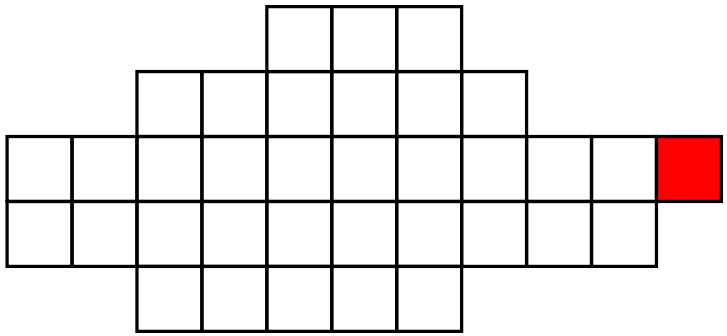
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$3^{[2]} \cdot 1^{[2]} \cdot 1^{[3]} \cdot 2 \cdot 3^{[2]} \cdot 3$

A CAT algorithm for F_v: example

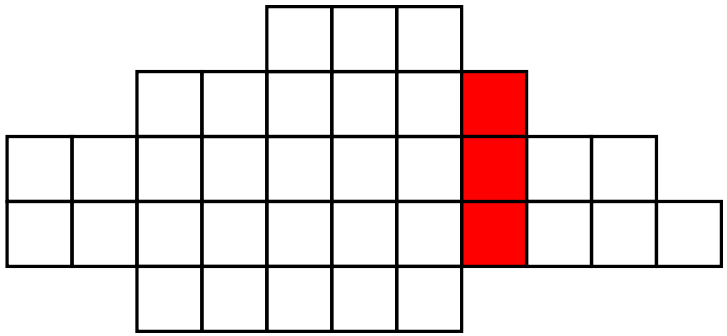
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$2^{[2]} \cdot 1^{[2]} \cdot 1^{[3]} \cdot 2 \cdot 2^{[2]} \cdot 3$

A CAT algorithm for F_v: example

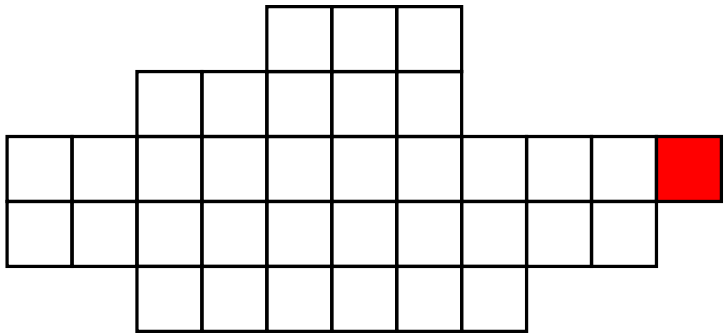
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$2^{[2]} \cdot 1^{[2]} \cdot 1^{[3]} \cdot 2 \cdot 2^{[2]} \cdot 2$

A CAT algorithm for F_v: example

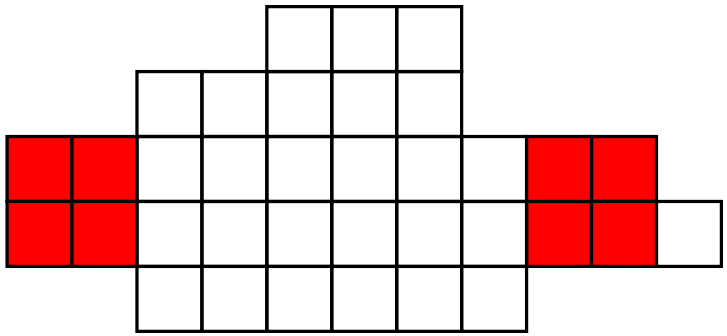
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$2^{[2]} \cdot 1^{[2]} \cdot 1^{[3]} \cdot 1 \cdot 2^{[2]} \cdot 3$

A CAT algorithm for F_v: example

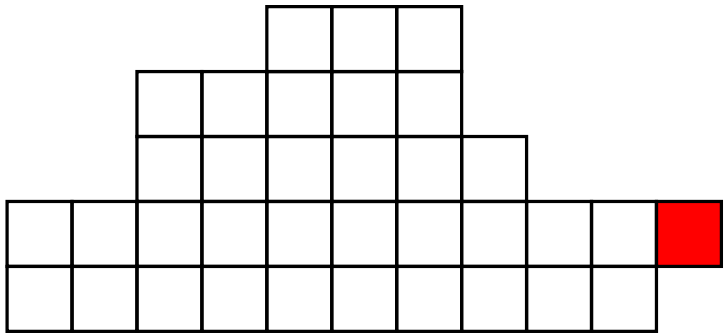
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$2^{[2]} \cdot 1^{[2]} \cdot 1^{[3]} \cdot 1 \cdot 2^{[2]} \cdot 2$

A CAT algorithm for F_v: example

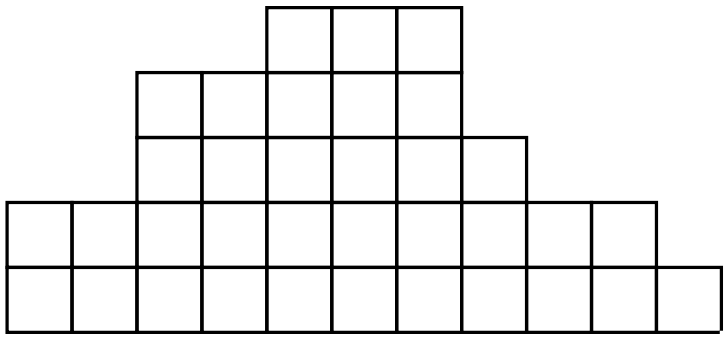
Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$1^{[2]} \cdot 1^{[2]} \cdot 1^{[3]} \cdot 1 \cdot 1^{[2]} \cdot 2$

A CAT algorithm for F_v: example

Let $v = 2^{[2]} \cdot 4^{[2]} \cdot 5^{[3]} \cdot 2^{[2]} \cdot 1$ and call POSITION(1)



$1^{[2]} \cdot 1^{[2]} \cdot 1^{[3]} \cdot 1 \cdot 1^{[2]} \cdot 1$

A CAT algorithm for F_v

Lemma

Procedure POSITION *runs in time* $O(|F_v|)$

Indeed, the execution tree of POSITION is a tree of height $r-1$ where

- internal nodes have degree at least 2
- all leaves are at level $r-1$
- each leaf corresponds to a different $P \in \mathrm{LPol}(n)$ with $\pi(P) = v$

The algorithm

```
Procedure GenLPol(n)
for h:=n downto 2 do
  for w:=⌊n/h⌋ downto 1 do
    R.height := h; R.width := w;
    m:= n − hw;
    LFerrer(m, h − 1);
  end for
end for
R.height := 1; R.width := n;
```

```
Procedure LFerrer(n, h)
for m:=n downto 0 do
  t:=Init(m, h);
  RFerrer(n − m, h);
  while Rmost(t) ≠ ⊥ do
    t:=Next(t);
    RFerrer(n − m, h);
  end while
end for
```

```
Procedure RFerrer(n, h)
t′:=Init(n, h);
Position(1);
while Rmost(t′) ≠ ⊥ do
  t′:=Next(t′);
  Position(1);
end while
```

Theorem

Procedure GenLPol*(n) runs in time O(|LPol(n)|)*

Conclusions

The "granular approach" has lead to CAT algorithms for

- Ice/Sand piles (2008, 2010)
- Symmetric Ice/Sand piles (2010)
- Plane partitions (2011)
- Parallelogram polyominoes (2011)
- L-convex polyominoes

Work in progress

adapt the method to

- Z-convex polyominoes
- convex polyominoes