Recursive structure of digital lines and planes in the context of image analysis

Xavier Provençal
Laboratoire de Mathématiques
Université de Savoie

Journées Montoires
September 25, 2014, Nancy

Outline

1. Image analysis
2. Recursive structure of digital lines
3. Recursive structure of digital planes

Outline

1. Image analysis (Why do we care)
2. Recursive structure of digital lines (Everything is just great in 2D)
3. Recursive structure of digital planes (How about 3D ?)

Part I

Image analysis

Image analysis

Definition (Wikipedia)

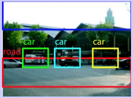
Image analysis is the extraction of meaningful information from images; mainly from digital images by means of digital image processing techniques.

Image analysis

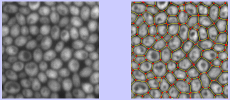
Definition (Wikipedia)

Image analysis is the extraction of meaningful information from images; mainly from digital images by means of digital image processing techniques.

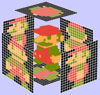
Object recognition



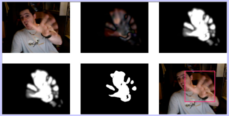
Segmentation



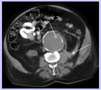
3D reconstruction



Motion detection/tracking



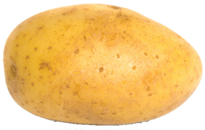
Medical imaging



Optical character recognition

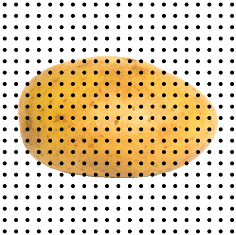
Digitalization

Digitization : $\mathrm{Dig}(P) = P \cap \mathbb{Z}^d$.



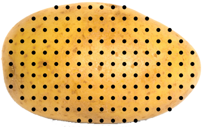
Digitalization

Digitization : $\mathrm{Dig}(P) = P \cap \mathbb{Z}^d$.



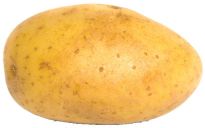
Digitalization

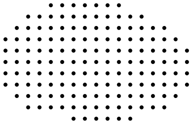
Digitization : $\mathrm{Dig}(P) = P \cap \mathbb{Z}^d$.

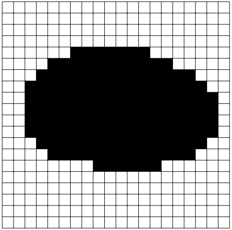


Digitalization

Digitization : $\mathrm{Dig}(P) = P \cap \mathbb{Z}^d$.

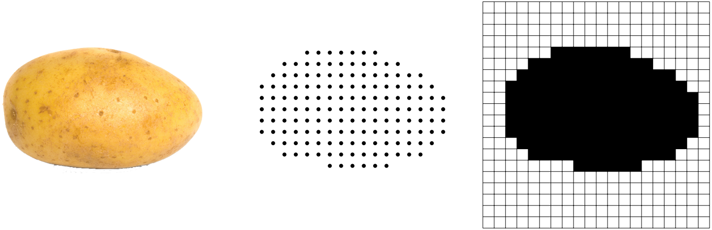






Digitalization

Digitization : $\mathrm{Dig}(P) = P \cap \mathbb{Z}^d$.

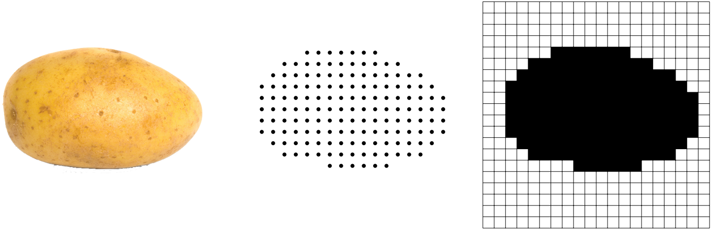


Given $\mathrm{Dig}(P)$, what can we say about P ?

- Convexity ?
- Area ?
- Perimeter ?
- Curvature ?
- ...

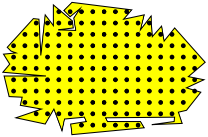
Digitalization

Digitization : $\mathrm{Dig}(P) = P \cap \mathbb{Z}^d$.



Given $\mathrm{Dig}(P)$, what can we say about P ?

- Convexity ?
- Area ?
- Perimeter ?
- Curvature ?
- ...

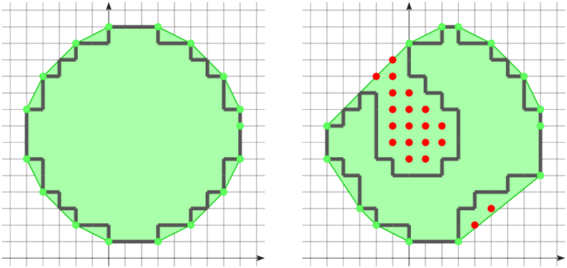


Digital convexity

Definition

A digital set $D \subset Z^d$ is digitally convex if

- $\mathrm{Dig}(\mathrm{Conv}(D)) = D$.



Definitions and characterizations :

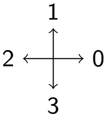
- [Minsky and Papert 1969]
- [Sklansky 1970]
- [Kim, Rosenfeld 1981]
- [Hübler, Klette, Voss 1981]
- [Chassery 1983]
- ...
- [Brlek, Lachaud, P., Reutenauer 2009]

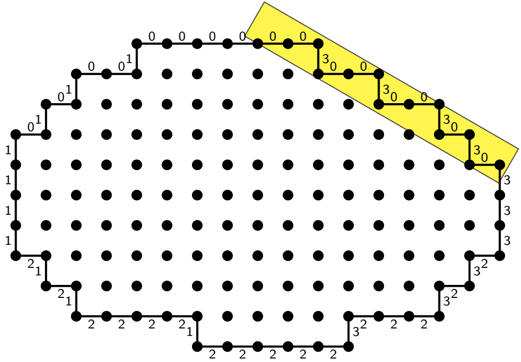
DSS on the boundary of a shape

Definition

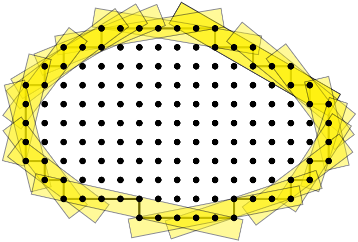
A Digital Straight Segment (DSS) is, equivalently :

- Finite and connected part of a Digital Straight Line.
- A finite factor of a Sturmian word.
- A finite 1-balanced word.





Tangential cover



Definition ([Feschet, Tougne 99])

The tangential cover of a discrete shape is the sequence of all maximal DSS on its boundary.

Theorem ([Debled-Rennesson, Reveilles 1995][Lachaud, Vialard, de Vieilleville 2007])

The computation of the tangential cover take a time in $\mathcal{O}(n)$ where n is the number of points on the boundary of the shape.

Applications of the tangential cover include :

- Convexity test
  [Debled-Rennesson, Reiter-Doerksen 04]
- Tangent estimation
  [Feschet, Tougne 99], [Lachaud, de Vieilleville 07]
- Length estimation
  [Lachaud, de Vieilleville 07]
- Curvature estimation
  [Lachaud, Kerautret, Naegel 08]
- Automatic noise detection
  [Lachaud, Kerautret 12]

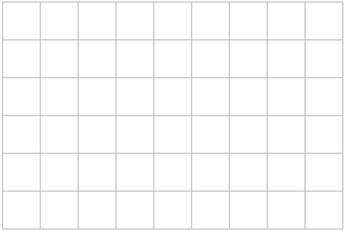
Part II

Recursive structure of digital lines

Christoffel words

Definition ([Christoffel 1875])

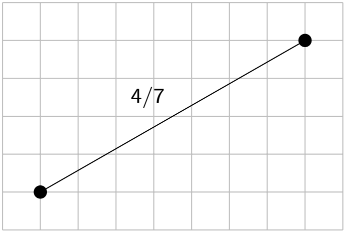
A **Christoffel word** codes digital path right below a segments between two consecutive integer points



Christoffel words

Definition ([Christoffel 1875])

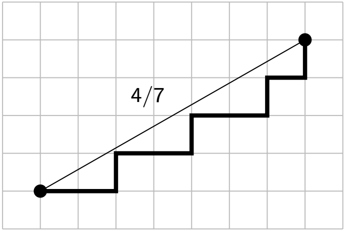
A **Christoffel word** codes digital path right below a segments between two consecutive integer points



Christoffel words

Definition ([Christoffel 1875])

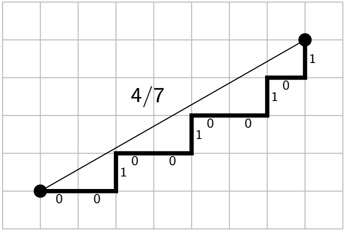
A **Christoffel word** codes digital path right below a segments between two consecutive integer points



Christoffel words

Definition ([Christoffel 1875])

A **Christoffel word** codes digital path right below a segments between two consecutive integer points

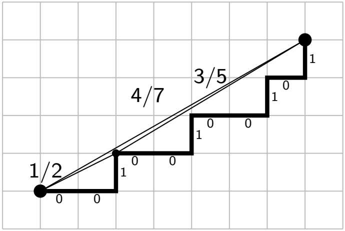


$w = 00100100101$ is the Christoffel word of **slope** $4/7$.

Christoffel words

Definition ([Christoffel 1875])

A **Christoffel word** codes digital path right below a segments between two consecutive integer points



$w = 001 \cdot 00100101$ is the Christoffel word of **slope** $4/7$.

Theorem ([Borel, Laubie 93])

Any Christoffel word, other than 0 *and* 1*, can be written in a unique way as a product of two Christoffel words.*

This is called the **standard factorization**, noted $w = (u, v)$.

Christoffel Tree

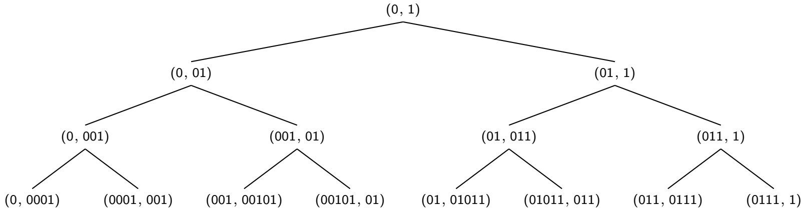
If (u, v) is a standard factorization, then (u, uv) and (uv, v) are standard factorizations of Christoffel words.

Christoffel Tree

If (u, v) is a standard factorization, then (u, uv) and (uv, v) are standard factorizations of Christoffel words.

The **Christoffel Tree** is the tree obtained, starting from $(0, 1)$, using the rule :

$$(u, v) \to (u, uv) \quad (uv, v)$$



Christoffel Tree

If (u, v) is a standard factorization, then (u, uv) and (uv, v) are standard factorizations of Christoffel words.

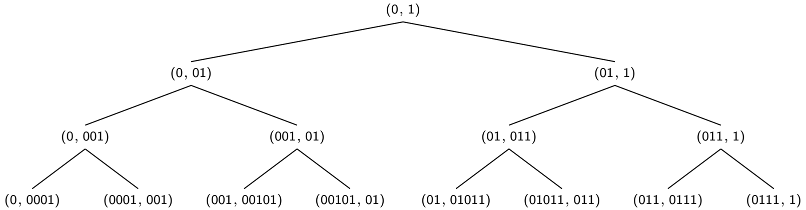
The **Christoffel Tree** is the tree obtained, starting from $(0, 1)$, using the rule :

$$(u, v)$$

$$(u, uv) \qquad\qquad (uv, v)$$

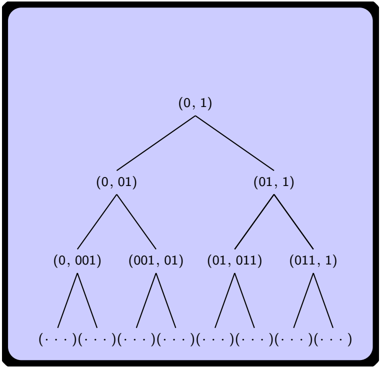
Theorem

Every Christoffel word appears exactly once in the Christoffel Tree.

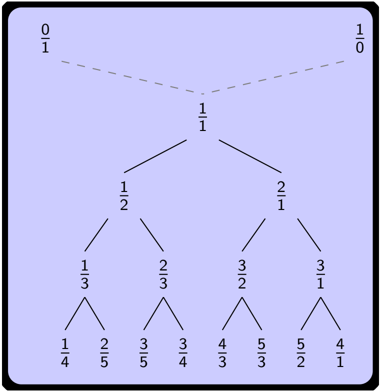


Stern-Brocot Tree

Christoffel tree



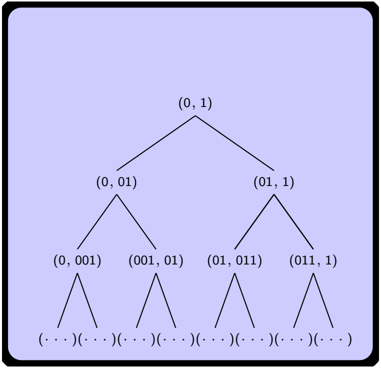
Stern-Brocot tree.



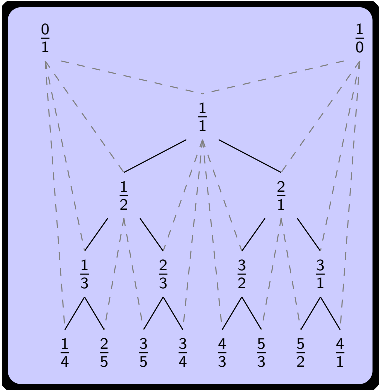
Every irreducible fraction appears exactly once in the Stern-Brocot tree.

Stern-Brocot Tree

Christoffel tree



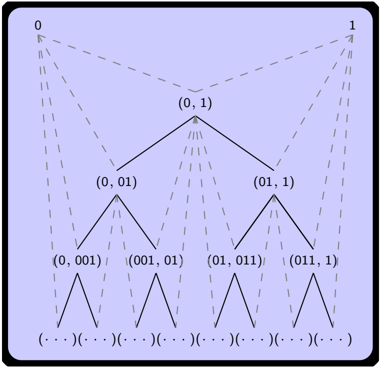
Stern-Brocot tree.



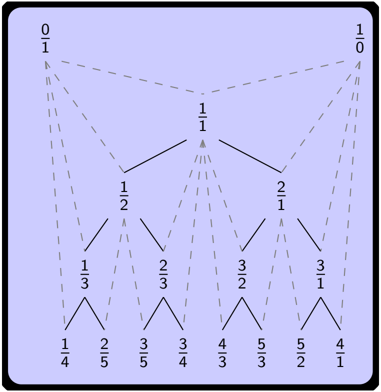
Every irreducible fraction appears exactly once in the Stern-Brocot tree.

Stern-Brocot Tree

Christoffel tree



Stern-Brocot tree.



Every irreducible fraction appears exactly once in the Stern-Brocot tree.

Recursive formula

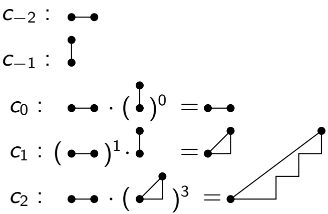
Theorem ([Berstel 92])

The Christoffel word c_n of slope $[z_0; z_1, \ldots, z_n]$ is given recursively by :

$$c_n = \begin{cases} c_{2m-2}\, c_{2m-1}^{z_{2m}} & \text{if } n = 2m, \\ c_{2m}^{z_{2m+1}}\, c_{2m-1} & \text{if } n = 2m+1. \end{cases} \qquad \text{where } c_{-1} = 1, \text{ and } c_{-2} = 0,$$

Example : $3/4 = [0; 1, 3]$,

$$\begin{aligned} c_{-2} &= 0, \\ c_{-1} &= 1, \\ c_0 &= c_{-2} \cdot c_{-1}^0 = 0 \cdot (1)^0 = 0, \\ c_1 &= c_0^1 \cdot c_{-1} = (0)^1 \cdot 1 = 01, \\ c_2 &= c_0 \cdot c_1^3 = 0 \cdot (01)^3 = 0010101, \end{aligned}$$



Nested prefixes

Corollary

A Christoffel word that admits $w = (u, v)$ as a proper prefix, has a prefix of the form : $w^k v = (w, w^{k-1}v)$.

Identifying the longest prefix that is a Christoffel word :

Nested prefixes

Corollary

A Christoffel word that admits $w = (u, v)$ as a proper prefix, has a prefix of the form : $w^k v = (w, w^{k-1}v)$.

Identifying the longest prefix that is a Christoffel word :

Nested prefixes

Corollary

A Christoffel word that admits $w = (u, v)$ as a proper prefix, has a prefix of the form : $w^k v = (w, w^{k-1} v)$.

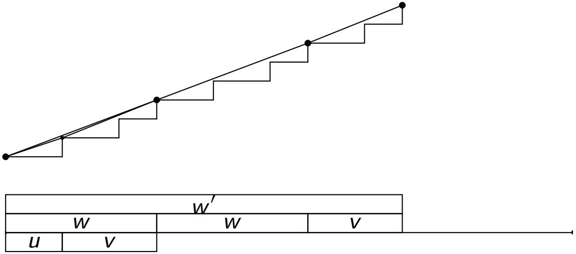
Identifying the longest prefix that is a Christoffel word :

Nested prefixes

Corollary

A Christoffel word that admits $w = (u, v)$ as a proper prefix, has a prefix of the form : $w^k v = (w, w^{k-1}v)$.

Identifying the longest prefix that is a Christoffel word :

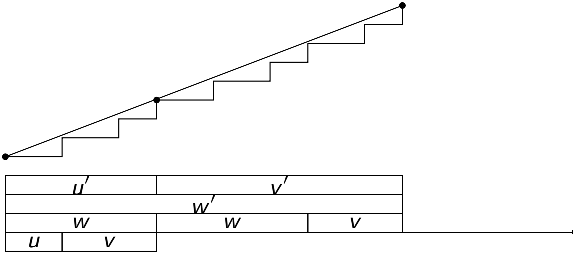


Nested prefixes

Corollary

A Christoffel word that admits $w = (u, v)$ as a proper prefix, has a prefix of the form : $w^k v = (w, w^{k-1}v)$.

Identifying the longest prefix that is a Christoffel word :

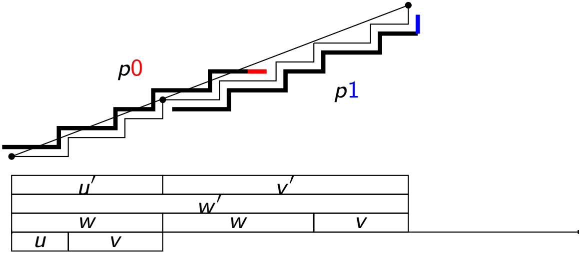


Nested prefixes

Corollary

A Christoffel word that admits $w = (u, v)$ as a proper prefix, has a prefix of the form : $w^k v = (w, w^{k-1}v)$.

Identifying the longest prefix that is a Christoffel word :



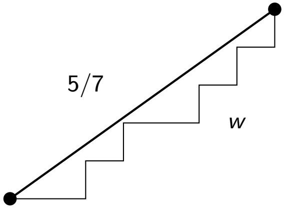
Corollary

A Christoffel word $w = (u, v)$ has a prefix $p0$ such that $v = p1$.

Lexicographic order

Property

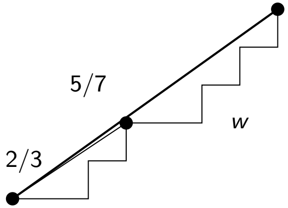
Lexicographic order on Christoffel words correspond to the order on the slope



Lexicographic order

Property

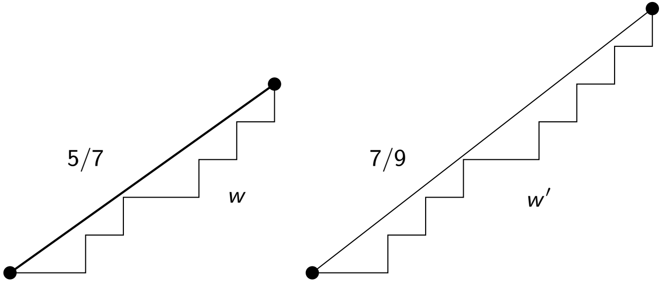
Lexicographic order on Christoffel words correspond to the order on the slope



Lexicographic order

Property

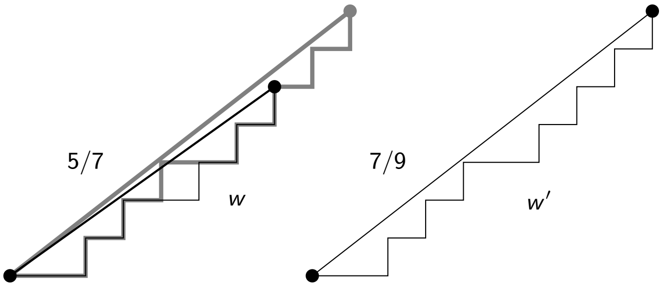
Lexicographic order on Christoffel words correspond to the order on the slope



Lexicographic order

Property

Lexicographic order on Christoffel words correspond to the order on the slope



Lyndon words

Definition ([Lyndon 54])
A w is a *Lyndon word* iff for every proper suffix s of w,

$$w <_{\mathbf{Lex}} s$$

Examples :

1. $aabab$ is Lyndon since $aabab <_{\mathbf{Lex}} \{abab, bab, ab, b\}$,
2. $aabaab$ is not Lyndon, let $aab <_{\mathbf{Lex}} w$.

Lyndon words

Definition ([Lyndon 54])

A w is a *Lyndon word* iff for every proper suffix s of w,

$$w <_{\mathbf{Lex}} s$$

Examples :

1. $aabab$ is Lyndon since $aabab <_{\mathbf{Lex}} \{abab, bab, ab, b\}$,
2. $aabaab$ is not Lyndon, let $aab <_{\mathbf{Lex}} w$.

Theorem ([Chen, Fox, Lyndon 58])

Every word has a unique factorization as non-increasing Lyndon words

Example :

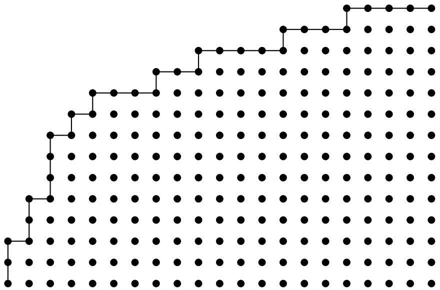
$$\begin{aligned} & 1\,1\,0\,1\,1\,0\,1\,1\,0\,0\,1\,0\,0\,1\,1\,0\,0\,0 \\ = \; & 1 \cdot 1 \cdot 0\,1\,1 \cdot 0\,1\,1 \cdot 0\,0\,1\,0\,0\,1\,1 \cdot 0 \cdot 0 \cdot 0 \\ = \; & (1)^2 \cdot (0\,1\,1)^2 \cdot (0\,0\,1\,0\,0\,1\,1)^1 \cdot (0)^3. \end{aligned}$$

Combinatorial view of convexity

Theorem ([Brlek, Lachaud, P., Reutenauer 09])
The north-west part of a digital shape is convex iff its Lyndon factorization contains only Christoffel words.

Sketch of the proof :

- Uniqueness of the Lyndon factorization.
- No integer points between a Christoffel word and its convex hull.



$$1101101110101000100100001000010000$$

$$= (1)^2 \cdot 0110111 \cdot (01)^2 \cdot 001001 \cdot 000010001 \cdot (0)^4$$

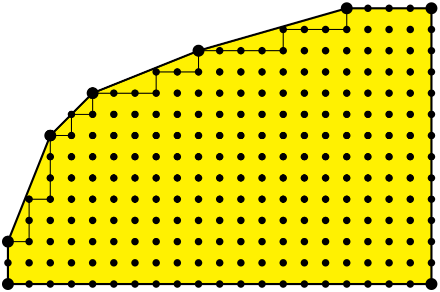
Combinatorial view of convexity

Theorem ([Brlek, Lachaud, P., Reutenauer 09])

The north-west part of a digital shape is convex iff its Lyndon factorization contains only Christoffel words.

Sketch of the proof :

- Uniqueness of the Lyndon factorization.
- No integer points between a Christoffel word and its convex hull.



$$1101101110101000100100001000010000$$

$$=(1)^2 \cdot 0110111 \cdot (01)^2 \cdot 001001 \cdot 000010001 \cdot (0)^4$$

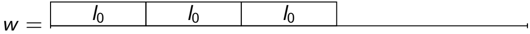
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$w =$

Duval algorithm

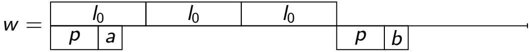
Recursive computation of the First Lyndon Prefix (FLF),



1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.

Duval algorithm

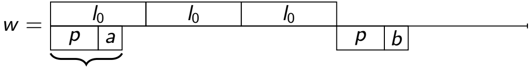
Recursive computation of the First Lyndon Prefix (FLF),



1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),



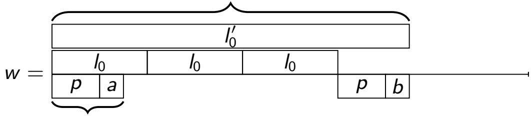
$a > b \implies$ Lyndon fact. starts by l_0^k

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF,

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



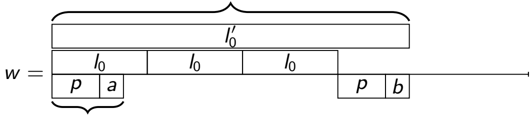
$a > b \implies$ Lyndon fact. starts by l_0^k

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

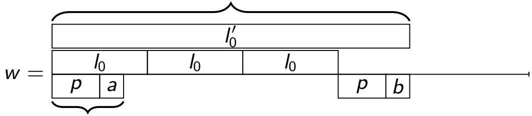
$$\underbrace{0}_{l_0}\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

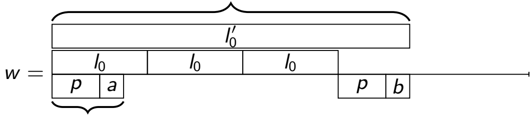
$$\downarrow\ \downarrow$$
$$\underbrace{0}_{l_0}\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

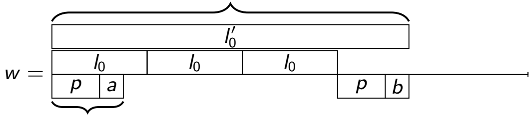
$$\begin{array}{ccccccccccccccccccccccc} \downarrow & & \downarrow \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ l_0 & l_0 \end{array}$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

$$\begin{array}{cccccccccccccccccccccccc} \downarrow & & & \downarrow \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}$$

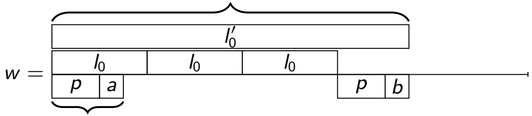
$\underbrace{0\ 0\ 1}_{l_0}$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

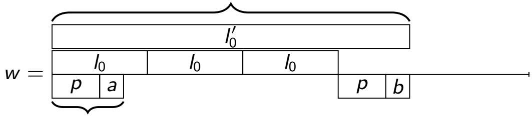
$$\underbrace{0\ 0\ 1}_{l_0}\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

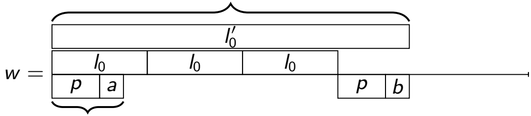
$$\underbrace{0\ 0\ 1}_{l_0}\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

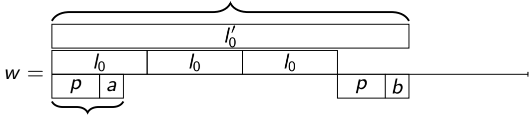
$$\underbrace{0\ 0\ 1}_{l_0}\ \underbrace{0\ 0\ 1}_{l_0}\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$$b > a \implies l_0^k pb \text{ is a Lyndon prefix}$$



$$a > b \implies \text{Lyndon fact. starts by } l_0^k$$

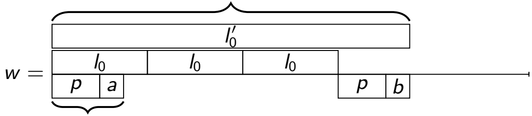
$$\underbrace{0\ \overset{\downarrow}{0}\ 1}_{l_0}\ \underbrace{0\ 0\ 1}_{l_0}\ 0\ \overset{\downarrow}{1}\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

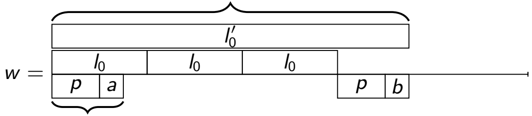
$$\underbrace{\overset{\downarrow}{0}\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ \overset{\downarrow}{0}\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

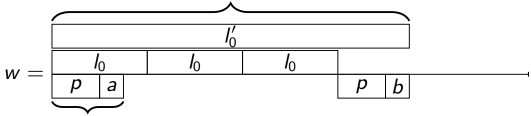
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

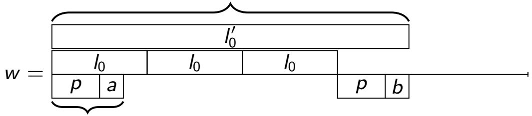
$$\underbrace{0\ 0\ \overset{\downarrow}{1}\ 0\ 0\ 1\ 0\ 1}_{l_0}\ 0\ 0\ \overset{\downarrow}{1}\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

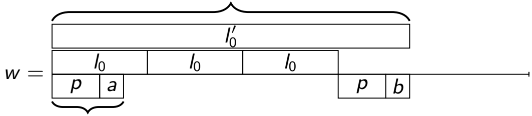
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

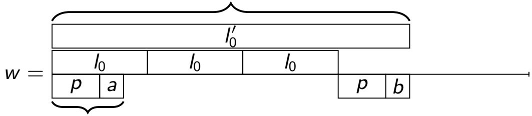
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$$b > a \implies l_0^k pb \text{ is a Lyndon prefix}$$



$$a > b \implies \text{Lyndon fact. starts by } l_0^k$$

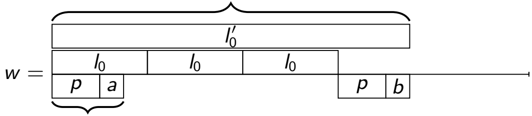
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

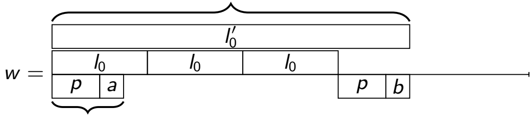
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ \overset{\downarrow}{0}\ 1}_{l_0}\ 0\ 0\ 1\ 0\ 0\ 1\ \overset{\downarrow}{0}\ 0\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

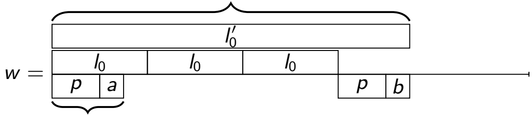
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ \overset{\downarrow}{1}}_{l_0}\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ \overset{\downarrow}{0}\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

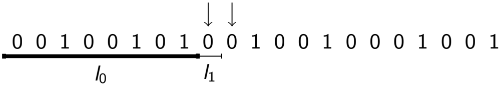
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

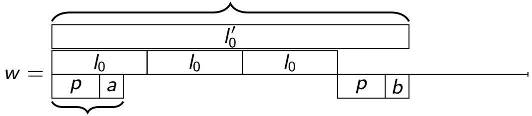


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

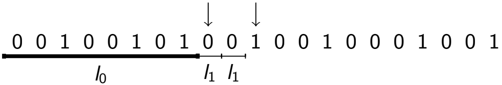
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

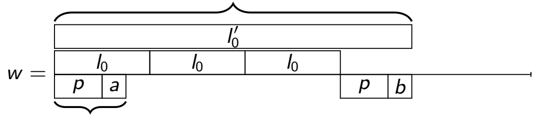


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

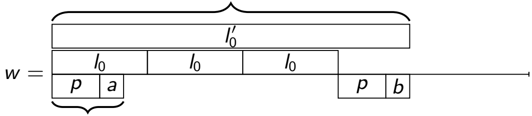
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ \underbrace{\overset{\downarrow}{0}\ 0\ 1}_{l_1}\ \overset{\downarrow}{0}\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

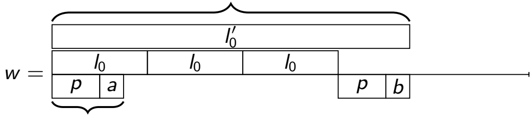
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ \underbrace{0\ 0\ 1}_{l_1}\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

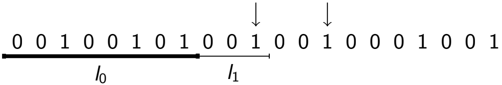
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

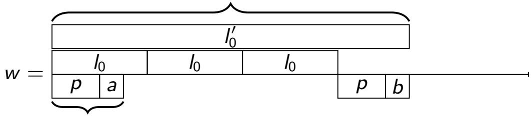


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

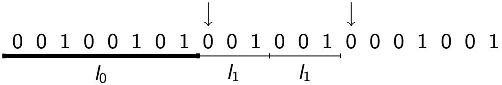
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

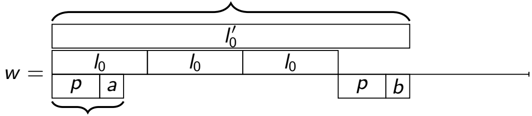


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

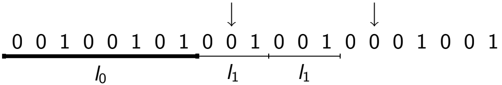
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

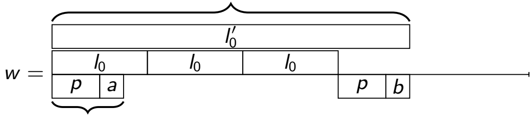


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

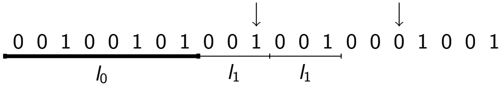
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

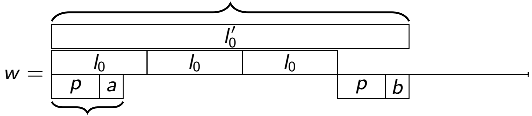


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

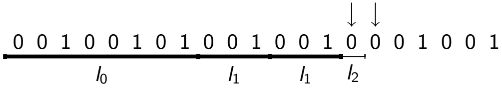
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

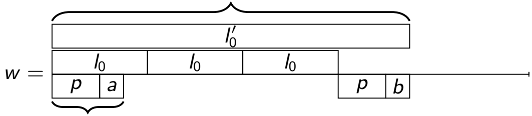


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

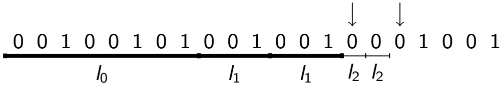
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

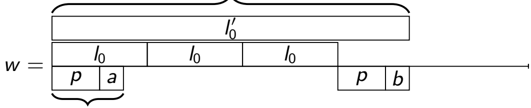


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

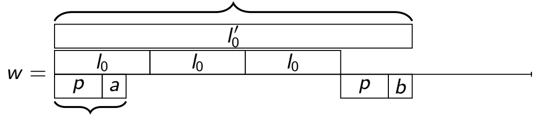
$$\underbrace{0\,0\,1\,0\,0\,1\,0\,1}_{l_0}\;\underbrace{0\,0\,1}_{l_1}\;\underbrace{0\,0\,1}_{l_1}\;\underbrace{\overset{\downarrow}{0}}_{l_2}\;\underbrace{0}_{l_2}\;\underbrace{0}_{l_2}\;\overset{\downarrow}{1}\,0\,0\,1$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

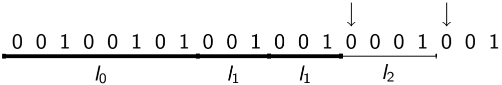
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

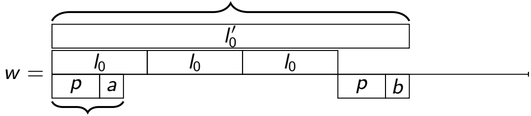


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

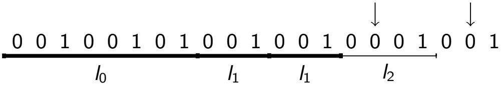
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

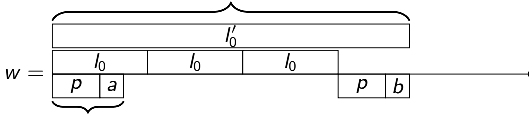


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

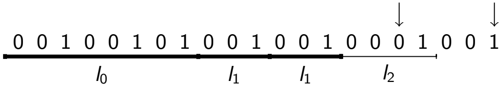
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

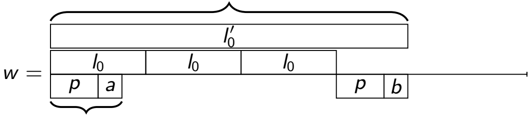


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

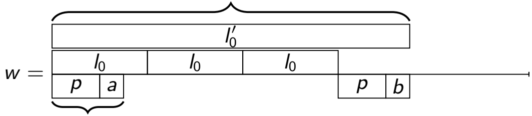
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ \underbrace{0\ 0\ 1}_{l_1}\ \underbrace{0\ 0\ 1}_{l_1}\ \underbrace{0\ 0\ 0\ 1\ 0\ 0\ 1}_{l_2}$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

$b > a \implies l_0^k pb$ is a Lyndon prefix



$a > b \implies$ Lyndon fact. starts by l_0^k

$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ \underbrace{0\ 0\ 1}_{l_1}\ \underbrace{0\ 0\ 1}_{l_1}\ \underbrace{0\ 0\ 0\ 1\ 0\ 0\ 1}_{l_2}$$

1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

When comparing two different letters, let $l_0 = (u, v)$:

- if $|p| = |v| - 1$ then $a = 0$ and $b = 1$ and l_0' is a Christoffel word.
- if $|p| \neq |v| - 1$ and $a = 1$ and $b = 0$ then l_0 is the first edge of the convex hull.
- if $|p| \neq |v| - 1$ and $a = 0$ and $b = 1$ then Shape is not convex.

Duval++

Convexity test

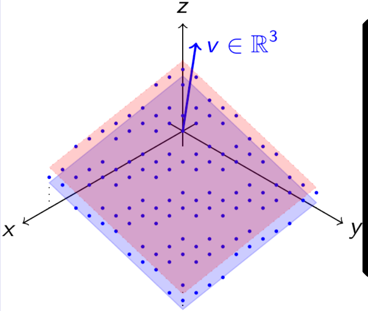
The Duval++ algorithm was indroduced to compute a first order reconstruction of a digital shape [Lachaud, P. 11].

Toolbox based on Duval++ for the computation of the tangential cover are available as `OneBalancedWordComputer` in the DGtal library.

Part III

Recursive structure of digital planes

Arithmetic digital planes



Definition ([Reveillès 91],[Forchhammer 89])

Digital plane with normal vector $\boldsymbol{v} \in \mathbb{R}^3 \setminus \{0\}$, shift $\mu \in \mathbb{R}$ and thickness θ.

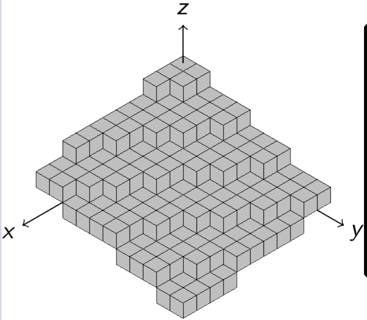
$$\mathcal{P}(\boldsymbol{v}, \mu, \theta) = \{\boldsymbol{x} \in \mathbb{Z}^3 \mid 0 \leq \langle \boldsymbol{v}, \boldsymbol{x} \rangle + \mu < \theta\}$$

In the case where $\theta = \|v\|_1$ then $\mathcal{P}$ is called standard.

(we only consider $\mu = 0$)

- Many definitions and characterizations : [Stojmenovič, Tosič 69], [Kim 84], [Kaufman 87], [Veelaert 93], [Debled-Rennesson 95], [Andrès, Acharya, Sibata 97], . . . [Labbé, Reutenauer 14]
- Recurrent structure : [Vuillon 96], [Arnoux, Berthé, Siegel 04] [Brimkov, Barneva 05], [Berthé 11].
- Topology : [Jamet, Toutant 09], [Domenjoud, Jamet, Toutant 09]

Arithmetic digital planes



Definition ([Reveillès 91],[Forchhammer 89])

Digital plane with normal vector $\boldsymbol{v} \in \mathbb{R}^3 \setminus \{0\}$, shift $\mu \in \mathbb{R}$ and thickness θ.

$$\mathcal{P}(\boldsymbol{v}, \mu, \theta) = \{\boldsymbol{x} \in \mathbb{Z}^3 \mid 0 \leq \langle \boldsymbol{v}, \boldsymbol{x} \rangle + \mu < \theta\}$$

In the case where $\theta = \|v\|_1$ then $\mathcal{P}$ is called standard.

(we only consider $\mu = 0$)

- Many definitions and characterizations : [Stojmenovič, Tosič 69], [Kim 84], [Kaufman 87], [Veelaert 93], [Debled-Rennesson 95], [Andrès, Acharya, Sibata 97], . . . [Labbé, Reutenauer 14]
- Recurrent structure : [Vuillon 96], [Arnoux, Berthé, Siegel 04] [Brimkov, Barneva 05], [Berthé 11].
- Topology : [Jamet, Toutant 09], [Domenjoud, Jamet, Toutant 09]

Continued fractions

- The recursive structure of DSS given by the continued fraction development of its slope.

Continued fractions

- The recursive structure of DSS given by the continued fraction development of its slope.

- Natural question : “Can we do the same in 3D ?”

Continued fractions

- The recursive structure of DSS given by the continued fraction development of its slope.
- Natural question : "Can we do the same in 3D ?"
- Answer is : "Yes. . . but its more complicated."

3D continued fraction algorithms

Euclide algorithm : given two number subtract the smaller to the larger.
$(5,12) \to (5,7) \to (5,2) \to (3,2) \to (1,2) \to (1,1) \to (1,0)$

3D continued fraction algorithms

Euclide algorithm : given two number subtract the smaller to the larger.
$(5,12) \to (5,7) \to (5,2) \to (3,2) \to (1,2) \to (1,1) \to (1,0)$

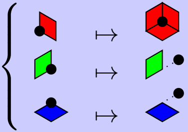
Given three numbers :

- Brun : subtract the second largest to the largest.
  $(3,7,5) \to (3,2,5) \to (3,2,2) \to (1,2,2) \to (1,2,0) \to (1,1,0) \to (1,0,0)$.
- Selmer : subtract the smallest to the largest.
  $(3,7,5) \to (3,4,5) \to (3,4,2) \to (3,2,2) \to (1,2,2) \to (1,2,0) \to (1,1,0) \to (1,0,0)$.
- Poincaré : subtract the smallest to the mid and the mid to the largest.
  $(3,7,5) \to (3,2,2) \to (1,2,0) \to (1,1,0) \to (1,0,0)$.
- Arnoux-Rauzy : subtract the sum of the two smallest to the largest (not always possible).
  $(3,7,5) \to \mathrm{impossible}$.
- Fully subtractive : subtract the smallest to the two others.
  $(3,7,5) \to (3,4,2) \to (1,2,2) \to (1,1,1) \to (0,0,1)$.
- ...

From CF to digital plane

Given a morphism $\sigma : \{1,2,3\} \mapsto \{1,2,3\}^*$,

$E_1^*(\sigma)$:



[Ito, Ohtsuki 93], [Ito, Ohtsuki 94], [Arnoux, Ito 01]

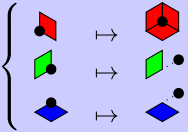
CF algo. $\longrightarrow$ Matrix $\longrightarrow$ Word morphism

$$\begin{array}{c}(a,b,c)\\ \downarrow \\ (a,b-a,c-a)\end{array} \longrightarrow \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \longrightarrow \sigma = \begin{cases} 1 \mapsto 1 \\ 2 \mapsto 12 \\ 3 \mapsto 13 \end{cases}$$

From CF to digital plane

Given a morphism $\sigma : \{1,2,3\} \mapsto \{1,2,3\}^*$,

$E_1^*(\sigma)$:



[Ito, Ohtsuki 93], [Ito, Ohtsuki 94], [Arnoux, Ito 01]

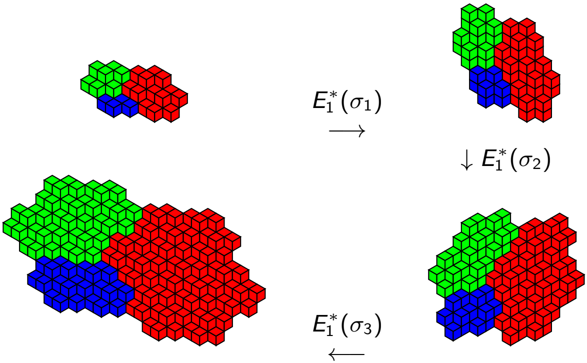
CF algo. $\longrightarrow$ Matrix $\longrightarrow$ Word morphism

$$\begin{matrix} (a,b,c) \\ \downarrow \\ (a, b-a, c-a) \end{matrix} \longrightarrow \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \longrightarrow \sigma = \begin{cases} 1 \mapsto 1 \\ 2 \mapsto 12 \\ 3 \mapsto 13 \end{cases}$$

Theorem ([Arnoux, Ito 01])

Let D_v be the visible faces of the voxels of $\mathcal{P}(v,0)$, then if σ is primitive and unimodular,

$$E_1^*(\sigma)(D_v) = D_{M_\sigma^T v}.$$



- Powerfull framework for the study of digital planes
  - Generation : [Arnoux, Berthé, Ito 02] [Fernique 09], [Berthé, Bourdon, Jolivet, Siegel 13], [Furukado, Ito, Yasutomi 13]
  - Characterization : [Arnoux, Berthé, Fernique, Jamet 07], [Berthé, Fernique 11]
  - Topology :[Berthé, Lacasse, Paquin, P. 13], [Berthé, Jolivet, Siegel 14]

- Still some work to do before practical use.

A construction guided by Fully Subtractive

The fully subtractive CF algorithm :

$$\mathbf{FS}\left((a,b,c)\right) = \begin{cases} (\quad a, \quad b-a, c-a) & \text{if } a = \min(a,b,c), \\ (a-b, \quad b, \quad c-b) & \text{if } b = \min(a,b,c), \\ (a-c, b-c, \quad c \quad) & \text{if } c = \min(a,b,c). \end{cases}$$

Given a vector v, the execution of the algorithm produces a sequence of vectors $(v_n)_{n\geq 0}$ defined by :

- $v_0 = v$,
- for $n \geq 1$, $v_n = \mathbf{FS}(v_{n-1})$.

A construction guided by Fully Subtractive

The fully subtractive CF algorithm :

$$\mathbf{FS}\left((a,b,c)\right) = \begin{cases} (\ \ a, \quad b-a, c-a) & \text{if } a = \min(a,b,c), \\ (a-b, \quad b, \quad c-b) & \text{if } b = \min(a,b,c), \\ (a-c, b-c, \quad c \ \ ) & \text{if } c = \min(a,b,c). \end{cases}$$

Given a vector v, the execution of the algorithm produces a sequence of vectors $(v_n)_{n\geq 0}$ defined by :

- $v_0 = v$,
- for $n \geq 1$, $v_n = \mathbf{FS}(v_{n-1})$.

If at one step, $2\|v_n\|_\infty > \|v_n\|_1$ (i.e. one coordinate is bigger then the sum of the two others), then the **FS** algorithm "fails".

Examples :

v_0	$(1,\pi,10)$	$(1,\pi,20)$
v_1	$(1,\pi-1,9)$	$(1,\pi-1,19)$
v_2	$(1,\pi-2,8)$	$(1,\pi-2,18)$
v_3	$(1,\pi-3,8)$	$(1,\pi-3,18)$
v_4	$(4-\pi,\pi-3,11-\pi)$	$(4-\pi,\pi-3,21-\pi)$
v_5	$(7-2\pi,\pi-3,14-2\pi)$	$(7-2\pi,\pi-3,24-2\pi)$
$\vdots$	$\vdots$	$\vdots$

Construction guided by Fully Subtractive

Definition
Let $\mathcal{K}$ be the set of vectors such that $2\|v_n\|_\infty < \|v\|_1$ for all $n \geq 0$.

$$v \in \mathcal{K} \implies \lim_{n\to\infty} v_n = 0.$$

Construction guided by Fully Subtractive

Definition ([Domenjoud,Vuillon 12],[Berthé, Jamet, Jolivet, P. 2013])

For all $n \geq 0$, let :

- M_n be the matrix such that $v_{n+1} = M_n v_n$.
- δ_n be the index of the smallest coordinate of v_n.
- $\theta_n = \langle v_n, e_{\delta_n} \rangle$. (the quantity that is subtracted to the two other coordinates of v_n).
- $T_n = M_0^T M_1^T \cdots M_{n-1}^T e_{\delta_n}$,

- $\theta_n = \langle v_n, e_{\delta_n} \rangle = \langle M_{n-1} \cdots M_0 v, e_{\delta_n} \rangle = \langle v, \underbrace{M_0^T \cdots M_{n-1}^T e_{\delta_n}}_{T_n} \rangle$.
- $\displaystyle\sum_{n \geq 0} \theta_n = \frac{\|v\|_1}{2}$.
- For all finite $I \subset \mathbb{N}$, let $\mathcal{T}_I = \sum_{i \in I} T_i$, we have :
  - $0 \leq \langle v, \mathcal{T}_I \rangle < \dfrac{\|v\|_1}{2}$,
  - $\mathcal{T}_I \in \mathcal{P}(v, 0, \dfrac{\|v\|_1}{2})$.

Construction guided by Fully Subtractive

Definition ([Berthé, Domenjoud, Jamet, P. 13])

- Let $P_0 = \{(0,0,0)\}$,
- For all $n \geq 1$ let $P_n = P_{n-1} \cup (T_n + P_{n-1})$

Theorem ([Domenjoud, P., Vuillon 14])

If $v \in \mathcal{K}$, *then* $P_\infty = \mathcal{P}(v, 0, \frac{\|v\|_1}{2})$

Construction guided by Fully Subtractive

Definition ([Berthé, Domenjoud, Jamet, P. 13])

- Let $P_0 = \{(0,0,0)\}$,
- For all $n \geq 1$ let $P_n = P_{n-1} \cup (T_n + P_{n-1})$

Theorem ([Domenjoud, P., Vuillon 14])

If $v \in \mathcal{K}$, then $P_\infty = \mathcal{P}(v, 0, \frac{\|v\|_1}{2})$

Examples :

- $v = (\beta, 2\beta + \beta^2, 1) \in \mathcal{K}$ où β est la racine réelle de $x^3 + 2x^2 + 2x - 1$.

Construction guided by Fully Subtractive

Definition ([Berthé, Domenjoud, Jamet, P. 13])

- Let $P_0 = \{(0,0,0)\}$,
- For all $n \geq 1$ let $P_n = P_{n-1} \cup (T_n + P_{n-1})$

Theorem ([Domenjoud, P., Vuillon 14])

If $v \in \mathcal{K}$, then $P_\infty = \mathcal{P}(v, 0, \frac{\|v\|_1}{2})$

Examples :

- $v = (\beta, 2\beta + \beta^2, 1) \in \mathcal{K}$ où β est la racine réelle de $x^3 + 2x^2 + 2x - 1$.

$T_0 = (1, 0, 0)$

Construction guided by Fully Subtractive

Definition ([Berthé, Domenjoud, Jamet, P. 13])

- Let $P_0 = \{(0,0,0)\}$,
- For all $n \geq 1$ let $P_n = P_{n-1} \cup (T_n + P_{n-1})$

Theorem ([Domenjoud, P., Vuillon 14])

If $v \in \mathcal{K}$, then $P_\infty = \mathcal{P}(v, 0, \frac{\|v\|_1}{2})$

Examples :

- $v = (\beta, 2\beta + \beta^2, 1) \in \mathcal{K}$ où β est la racine réelle de $x^3 + 2x^2 + 2x - 1$.

$T_0 = (1,0,0)$
$T_1 = (-1,1,0)$

Construction guided by Fully Subtractive

Definition ([Berthé, Domenjoud, Jamet, P. 13])

- Let $P_0 = \{(0,0,0)\}$,
- For all $n \geq 1$ let $P_n = P_{n-1} \cup (T_n + P_{n-1})$

Theorem ([Domenjoud, P., Vuillon 14])

If $v \in \mathcal{K}$, then $P_\infty = \mathcal{P}(v, 0, \frac{\|v\|_1}{2})$

Examples :

- $v = (\beta, 2\beta + \beta^2, 1) \in \mathcal{K}$ où β est la racine réelle de $x^3 + 2x^2 + 2x - 1$.

$T_0 = (1,0,0)$
$T_1 = (-1,1,0)$
$T_2 = (-1,1,0)$

Construction guided by Fully Subtractive

Definition ([Berthé, Domenjoud, Jamet, P. 13])

- Let $P_0 = \{(0,0,0)\}$,
- For all $n \geq 1$ let $P_n = P_{n-1} \cup (T_n + P_{n-1})$

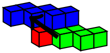
Theorem ([Domenjoud, P., Vuillon 14])

If $v \in \mathcal{K}$, then $P_\infty = \mathcal{P}(v, 0, \frac{\|v\|_1}{2})$

Examples :

- $v = (\beta, 2\beta + \beta^2, 1) \in \mathcal{K}$ où β est la racine réelle de $x^3 + 2x^2 + 2x - 1$.

$T_0 = (1,0,0)$
$T_1 = (-1,1,0)$
$T_2 = (-1,1,0)$
$T_3 = (1,-2,1)$



Construction guided by Fully Subtractive

Definition ([Berthé, Domenjoud, Jamet, P. 13])

- Let $P_0 = \{(0,0,0)\}$,
- For all $n \geq 1$ let $P_n = P_{n-1} \cup (T_n + P_{n-1})$

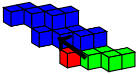
Theorem ([Domenjoud, P., Vuillon 14])

If $v \in \mathcal{K}$, then $P_\infty = \mathcal{P}(v, 0, \frac{\|v\|_1}{2})$

Examples :

- $v = (\beta, 2\beta + \beta^2, 1) \in \mathcal{K}$ où β est la racine réelle de $x^3 + 2x^2 + 2x - 1$.

$T_0 = (1,0,0)$
$T_1 = (-1,1,0)$
$T_2 = (-1,1,0)$
$T_3 = (1,-2,1)$
$T_4 = (1,-2,1)$



Construction guided by Fully Subtractive

Definition ([Berthé, Domenjoud, Jamet, P. 13])

- Let $P_0 = \{(0,0,0)\}$,
- For all $n \geq 1$ let $P_n = P_{n-1} \cup (T_n + P_{n-1})$

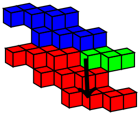
Theorem ([Domenjoud, P., Vuillon 14])

If $v \in \mathcal{K}$, then $P_\infty = \mathcal{P}(v, 0, \frac{\|v\|_1}{2})$

Examples :

- $v = (\beta, 2\beta + \beta^2, 1) \in \mathcal{K}$ où β est la racine réelle de $x^3 + 2x^2 + 2x - 1$.

$T_0 = (1,0,0)$
$T_1 = (-1,1,0)$
$T_2 = (-1,1,0)$
$T_3 = (1,-2,1)$
$T_4 = (1,-2,1)$
$T_5 = (1,2,-2)$



Construction guided by Fully Subtractive

Definition ([Berthé, Domenjoud, Jamet, P. 13])

- Let $P_0 = \{(0,0,0)\}$,
- For all $n \geq 1$ let $P_n = P_{n-1} \cup (T_n + P_{n-1})$

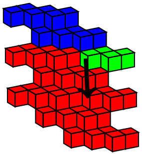
Theorem ([Domenjoud, P., Vuillon 14])

If $v \in \mathcal{K}$, *then* $P_\infty = \mathcal{P}(v, 0, \frac{\|v\|_1}{2})$

Examples :

- $v = (\beta, 2\beta + \beta^2, 1) \in \mathcal{K}$ où β est la racine réelle de $x^3 + 2x^2 + 2x - 1$.

$T_0 = (1,0,0)$
$T_1 = (-1,1,0)$
$T_2 = (-1,1,0)$
$T_3 = (1,-2,1)$
$T_4 = (1,-2,1)$
$T_5 = (1,2,-2)$
$T_6 = (1,2,-2)$



Construction guided by Fully Subtractive

Definition ([Berthé, Domenjoud, Jamet, P. 13])

- Let $P_0 = \{(0,0,0)\}$,
- For all $n \geq 1$ let $P_n = P_{n-1} \cup (T_n + P_{n-1})$

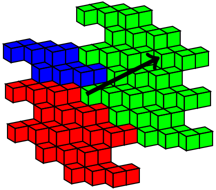
Theorem ([Domenjoud, P., Vuillon 14])

If $v \in \mathcal{K}$, then $P_\infty = \mathcal{P}(v, 0, \frac{\|v\|_1}{2})$

Examples :

- $v = (\beta, 2\beta + \beta^2, 1) \in \mathcal{K}$ où β est la racine réelle de $x^3 + 2x^2 + 2x - 1$.

$T_0 = (1,0,0)$
$T_1 = (-1,1,0)$
$T_2 = (-1,1,0)$
$T_3 = (1,-2,1)$
$T_4 = (1,-2,1)$
$T_5 = (1,2,-2)$
$T_6 = (1,2,-2)$
$T_7 = (-5,1,2)$



Construction guided by Fully Subtractive

Definition ([Berthé, Domenjoud, Jamet, P. 13])

- Let $P_0 = \{(0,0,0)\}$,
- For all $n \geq 1$ let $P_n = P_{n-1} \cup (T_n + P_{n-1})$

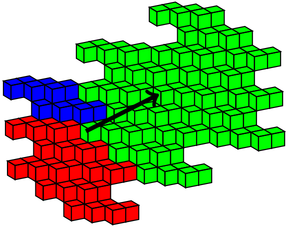
Theorem ([Domenjoud, P., Vuillon 14])

If $v \in \mathcal{K}$, then $P_\infty = \mathcal{P}(v, 0, \frac{\|v\|_1}{2})$

Examples :

- $v = (\beta, 2\beta + \beta^2, 1) \in \mathcal{K}$ où β est la racine réelle de $x^3 + 2x^2 + 2x - 1$.

$T_0 = (1,0,0)$
$T_1 = (-1,1,0)$
$T_2 = (-1,1,0)$
$T_3 = (1,-2,1)$
$T_4 = (1,-2,1)$
$T_5 = (1,2,-2)$
$T_6 = (1,2,-2)$
$T_7 = (-5,1,2)$
$T_8 = (-5,1,2)$



Construction guided by Fully Subtractive

Definition ([Berthé, Domenjoud, Jamet, P. 13])

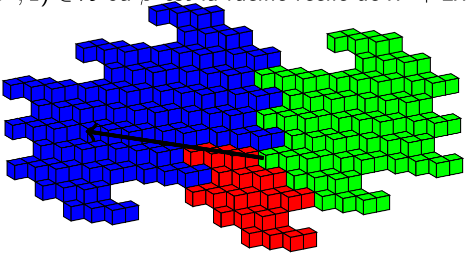
- Let $P_0 = \{(0,0,0)\}$,
- For all $n \geq 1$ let $P_n = P_{n-1} \cup (T_n + P_{n-1})$

Theorem ([Domenjoud, P., Vuillon 14])
If $v \in \mathcal{K}$, *then* $P_\infty = \mathcal{P}(v, 0, \frac{\|v\|_1}{2})$

Examples :

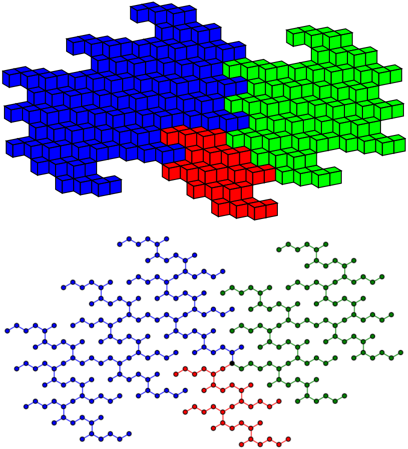
- $v = (\beta, 2\beta + \beta^2, 1) \in \mathcal{K}$ où β est la racine réelle de $x^3 + 2x^2 + 2x - 1$.

$T_0 = (1,0,0)$
$T_1 = (-1,1,0)$
$T_2 = (-1,1,0)$
$T_3 = (1,-2,1)$
$T_4 = (1,-2,1)$
$T_5 = (1,2,-2)$
$T_6 = (1,2,-2)$
$T_7 = (-5,1,2)$
$T_8 = (-5,1,2)$
$T_9 = (9,-8,1)$



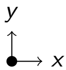
Tree structure

Theorem ([Domenjoud, Vuillon 12])
The adjacency graph of P_n has a tree rooted in $\vec{0}$.



Reinterpretation of the Christoffel tree

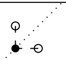
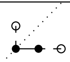
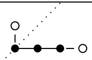
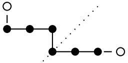
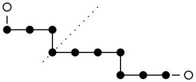
How to draw the Christoffel word with normal vector $(3, 8)$?



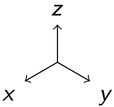
- Exclude the first and the last steps, we'll add them back at the end.
- At each step, replace one side by the whole pattern.

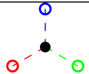
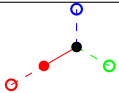
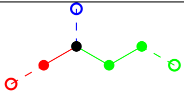
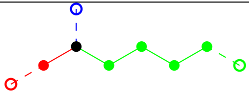
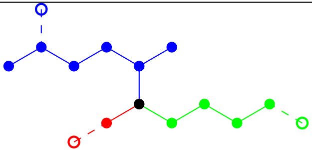
(u, v)

(u, uv) $\quad$ (uv, v)

Pattern	slope	Euclid algorithm
	$\frac{1}{1}$	$(\underline{3}, 8)$
	$\frac{1}{2}$	$(\underline{3}, 5)$
	$\frac{1}{3}$	$(3, \underline{2})$
	$\frac{2}{5}$	$(\underline{1}, 2)$
	$\frac{3}{8}$	$(1, 1)$

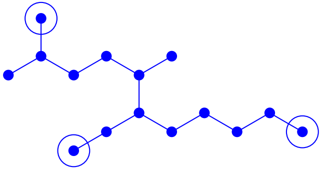
Unified view for patches of discrete planes



Pattern	Normal vector	Fully subtractive algorithm
	$(1,1,1)$	$(\underline{6},8,11)$
	$(1,2,2)$	$(6,\underline{2},5)$
	$(2,3,4)$	$(4,\underline{2},3)$
	$(3,4,6)$	$(2,2,\underline{1})$
	$(6,8,11)$	$(1,1,1)$

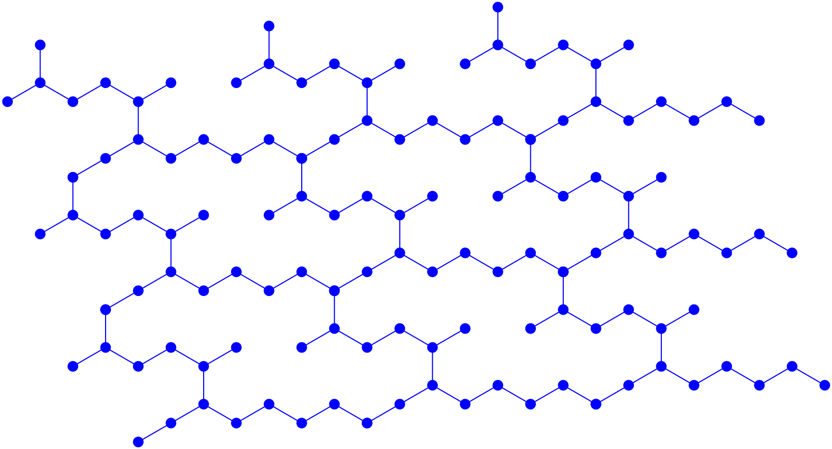
From pattern to digital plane

$(6, 8, 11)$



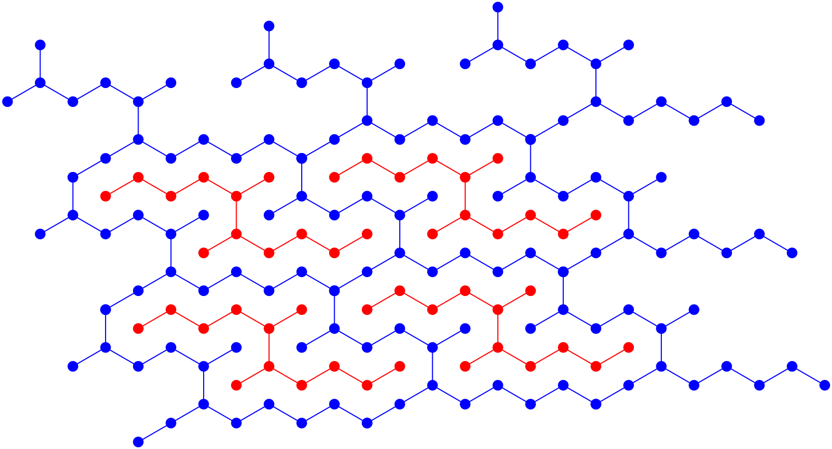
From pattern to digital plane

$(6, 8, 11)$



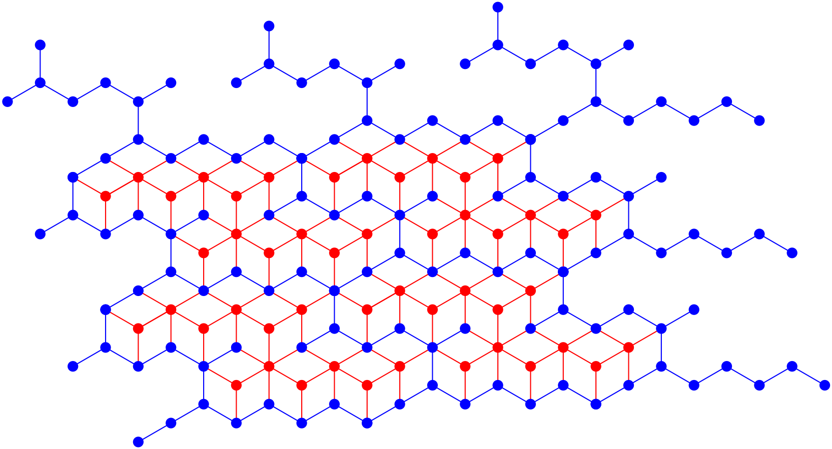
From pattern to digital plane

$(6, 8, 11)$



From pattern to digital plane

$(6, 8, 11)$



Mission accomplished ?

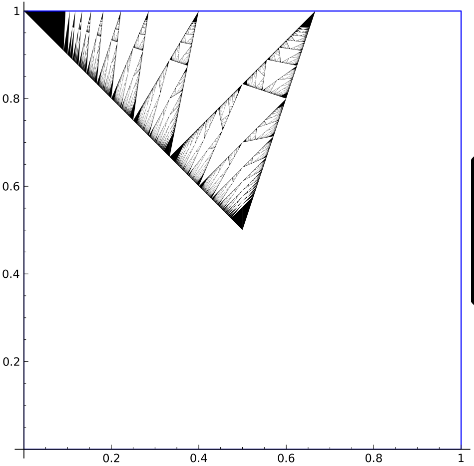
Mission accomplished ?

- No !

Mission accomplished ?

- No !

Theorem ([Kraaikamp, Meester 95])
The set $\mathcal{K}$ is a mesure-zero set.



Construction deals only with :

- vectors of $\mathcal{K}$,
- integer vectors such that FS reaches $(1, 1, 1)$.

$\{(x/z, y/z) \mid x \leq y \leq z$ and our construction deals with $(x, y, z)\}$

Merci pour votre attention

Fin