Recursive structure of digital planes, a combinatorial approach based on continued fractions

Xavier Provençal
Laboratoire de Mathématiques
Université Savoie Mont-Blanc

Journées Informatique et Géométrie 2015
8 octobre 2015, Paris

Outline

1. Recursive Structure of Digital line
2. Construction guided by Euclid
3. Generalization to higher dimensions

Part I

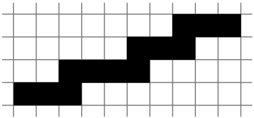
Recursive Structure of Digital line

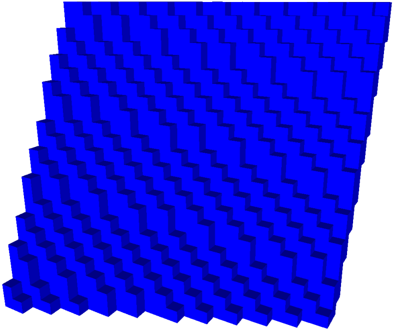
Digital lines and planes

Definition ([Reveillès 91])

The digital hyperplane $\mathcal{P}(v, \mu)$ with **normal vector** $v \in \mathbb{Z}^d$, **shift** $\mu \in \mathbb{R}$ is the subset of $\mathbb{Z}^d$ defined by:

$$\mathcal{P}(v, \mu) = \left\{ x \in \mathbb{Z}^d \mid \mu \leq \langle x, v \rangle < \mu + \|v\|_1 \right\}$$





Digital lines and planes

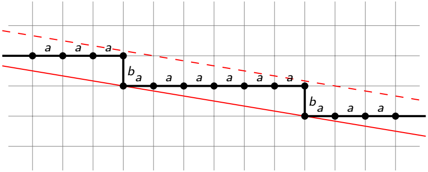
Definition ([Reveillès 91])

The digital hyperplane $\mathcal{P}(v,\mu)$ with **normal vector** $v \in \mathbb{Z}^d$, **shift** $\mu \in \mathbb{R}$ is the subset of $\mathbb{Z}^d$ defined by:

$$\mathcal{P}(v,\mu) = \left\{x \in \mathbb{Z}^d \mid \mu \leq \langle x, v\rangle < \mu + \|v\|_1\right\}$$

$\mathcal{P}((1,6),0)$

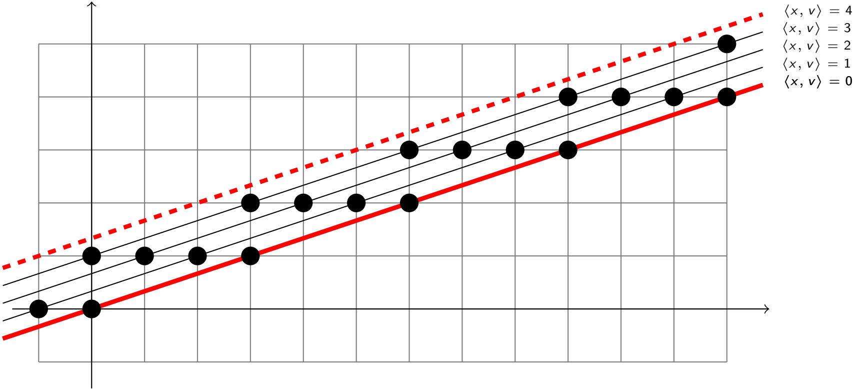
$0 \leq 1x + 6y < 7$



A digital line can be coded on two letters.

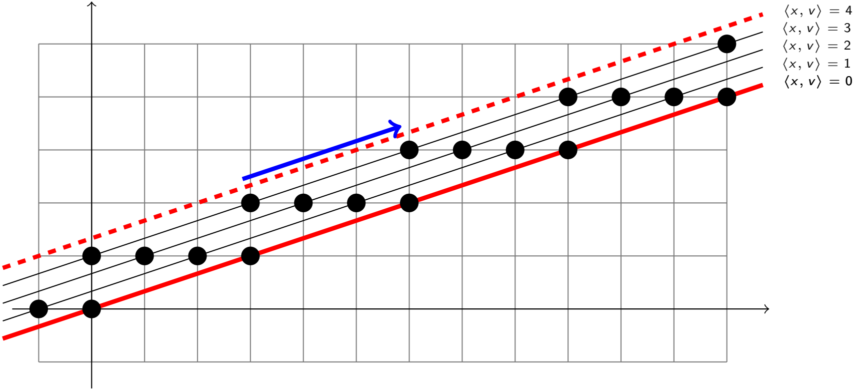
Periodic structure of a digital line

- $\langle x, v \rangle$ is the **height** of x,
- $v = (-3, 1)$,
- $\mathcal{P}(v, 0) = \{x \in \mathbb{Z}^2 \mid 0 \leq \langle x, v \rangle < 4\}$.



Periodic structure of a digital line

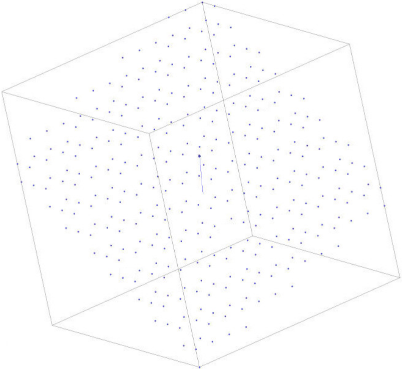
- $\langle x, v\rangle$ is the **height** of x,
- $v = (-3, 1)$,
- $\mathcal{P}(v, 0) = \{x \in \mathbb{Z}^2 \mid 0 \le \langle x, v\rangle < 4\}$.

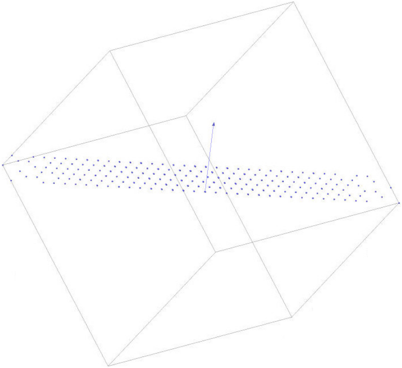


- $\langle x, v\rangle = \langle y, v\rangle \implies y - x$ is a period vector.
- A point of each height from 0 to $\|v\|_1 - 1$ appear in a period.

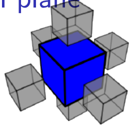
Periodic structure of a digital plane

$$v = (1,2,3), \quad \mathcal{P}(v,0) = \{x \in \mathbb{Z}^3 \mid 0 \leq \langle x, v \rangle < 6\}$$

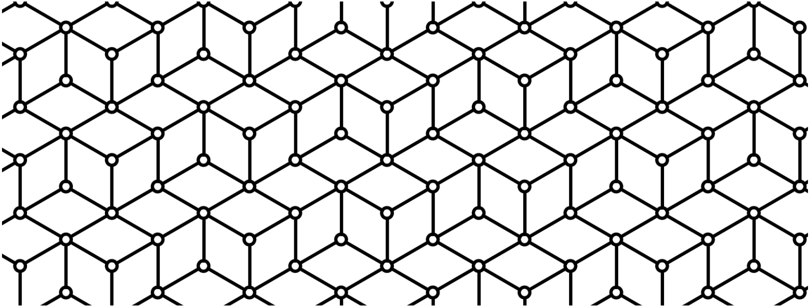




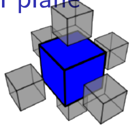
Periodic structure of a digital plane



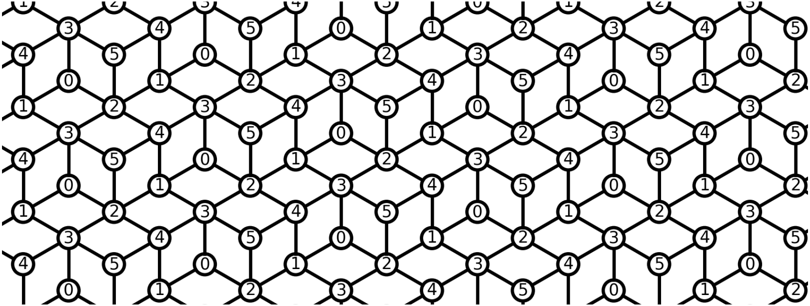
$$v = (1,2,3), \quad \mathcal{P}(v,0) = \{x \in \mathbb{Z}^3 \mid 0 \leq \langle x, v \rangle < 6\}$$



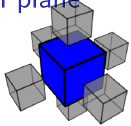
Periodic structure of a digital plane



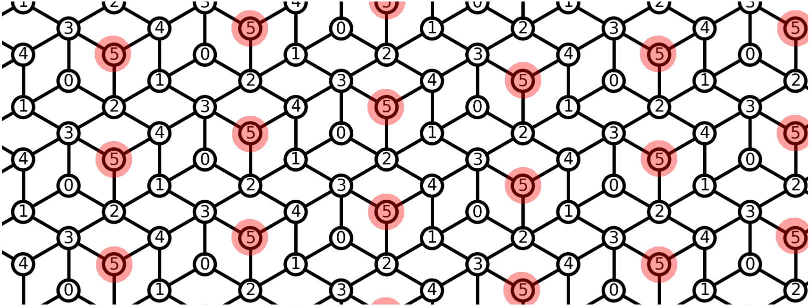
$$v = (1,2,3), \quad \mathcal{P}(v,0) = \{x \in \mathbb{Z}^3 \mid 0 \leq \langle x, v \rangle < 6\}$$



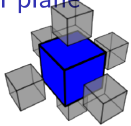
Periodic structure of a digital plane



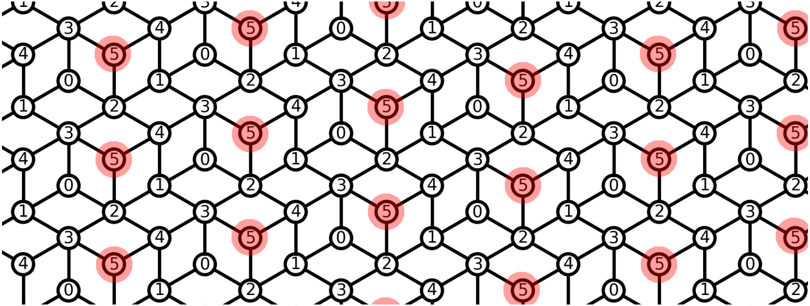
$$v = (1,2,3), \quad \mathcal{P}(v,0) = \{x \in \mathbb{Z}^3 \mid 0 \leq \langle x, v\rangle < 6\}$$



Periodic structure of a digital plane



$$v = (1, 2, 3), \quad \mathcal{P}(v, 0) = \{x \in \mathbb{Z}^3 \mid 0 \leq \langle x, v \rangle < 6\}$$

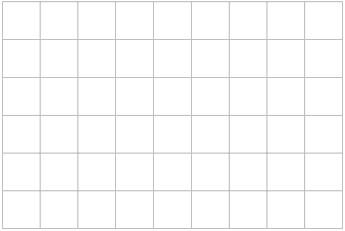


- $\langle x, v \rangle = \langle y, v \rangle \implies y - x$ is a period vector.
- A point of each height from 0 to $\|v\|_1 - 1$ appear in a period.

Christoffel words

Definition ([Christoffel 1875])

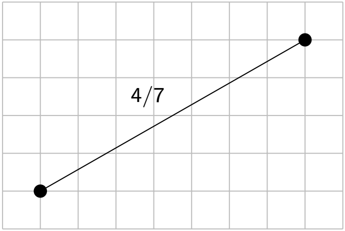
A **Christoffel word** codes digital path right below a segments between two consecutive integer points



Christoffel words

Definition ([Christoffel 1875])

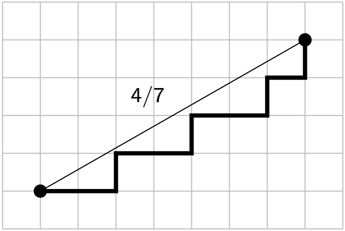
A **Christoffel word** codes digital path right below a segments between two consecutive integer points



Christoffel words

Definition ([Christoffel 1875])

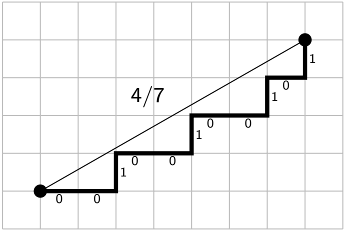
A **Christoffel word** codes digital path right below a segments between two consecutive integer points



Christoffel words

Definition ([Christoffel 1875])

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

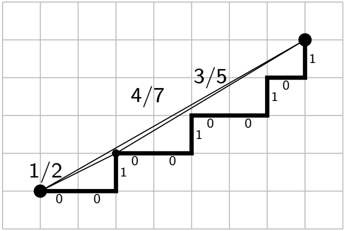


$w = 00100100101$ is the Christoffel word of **slope** $4/7$.

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** $4/7$.

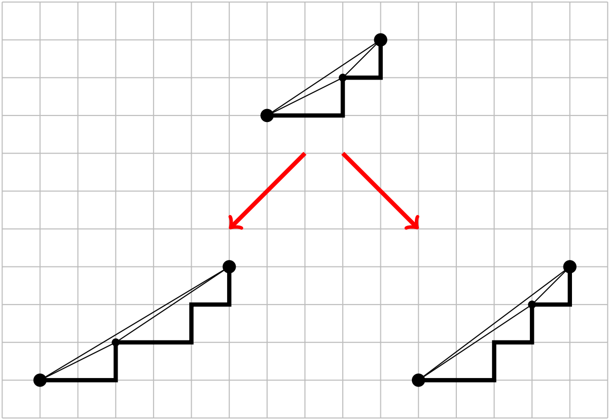
Theorem ([Borel, Laubie 93])

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.



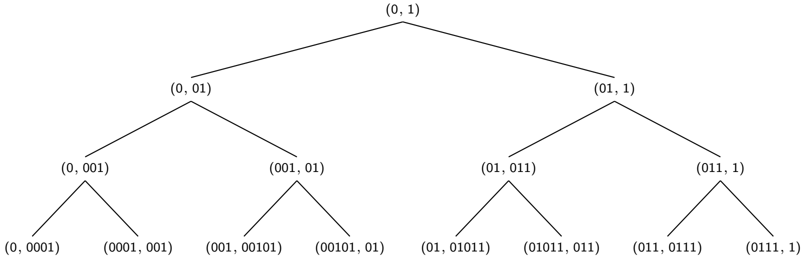
Christoffel Tree

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 :

$$(u, v)$$

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



Christoffel Tree

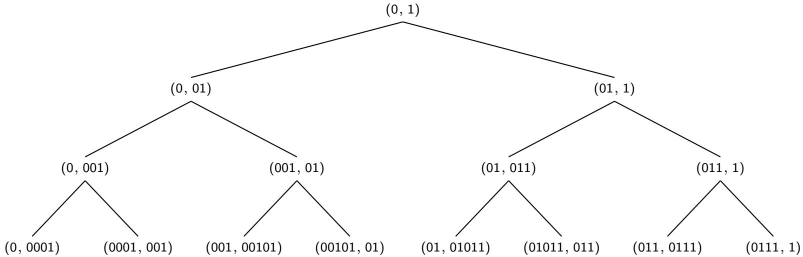
If (u, v) is a standard factorization, then (u, uv) and (uv, v) are standard factorizations of Christoffel words.

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

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

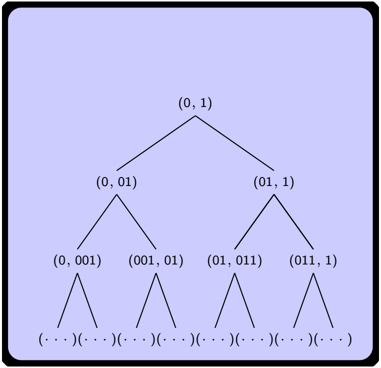
Theorem

Every Christoffel word appears exactly once in the Christoffel Tree.

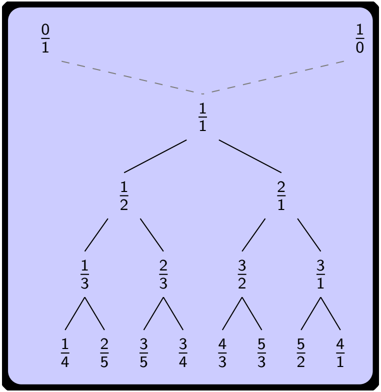


Stern-Brocot Tree

Christoffel tree



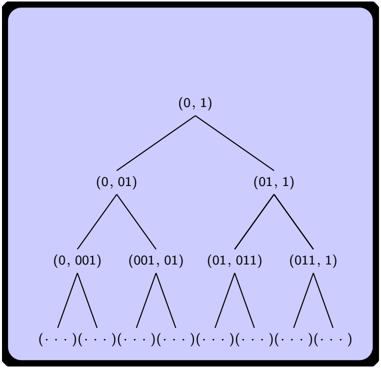
Stern-Brocot tree.



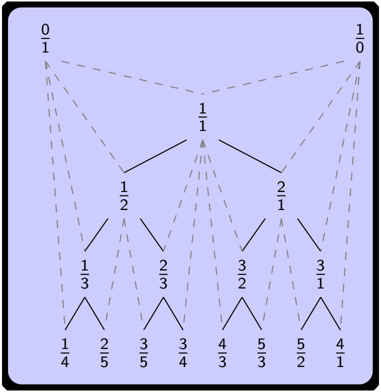
Every irreducible fraction appears exactly once in the Stern-Brocot tree.

Stern-Brocot Tree

Christoffel tree



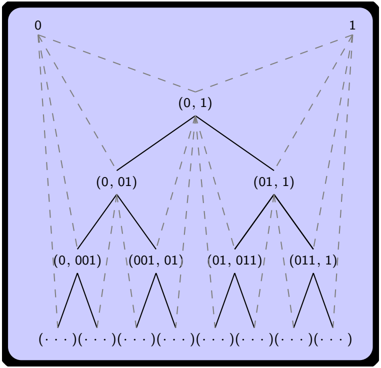
Stern-Brocot tree.



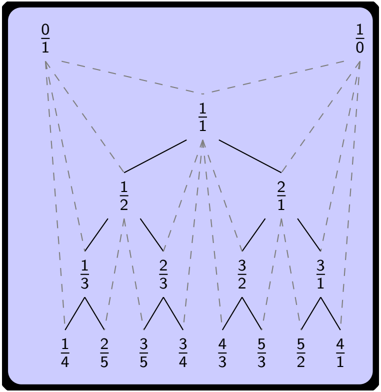
Every irreducible fraction appears exactly once in the Stern-Brocot tree.

Stern-Brocot Tree

Christoffel tree



Stern-Brocot tree.



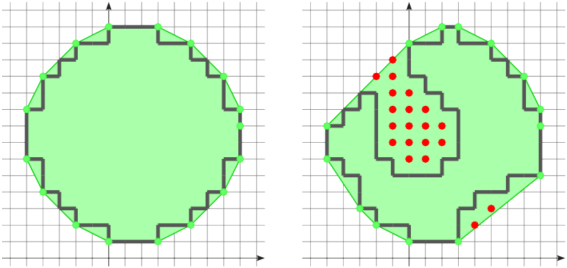
Every irreducible fraction appears exactly once in the Stern-Brocot tree.

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]

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 : $w^k v = (w, w^{k-1} v)$.

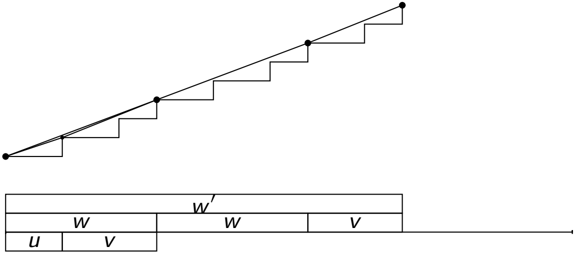
Identifying the longest prefix that is a Christoffel word :

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 :

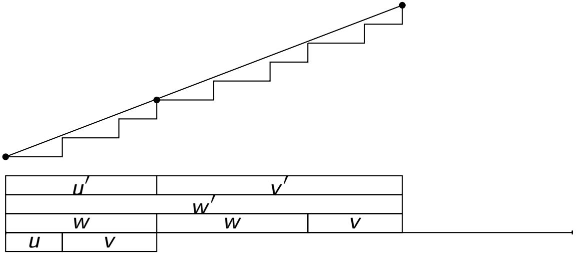


Nested prefixes

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 :

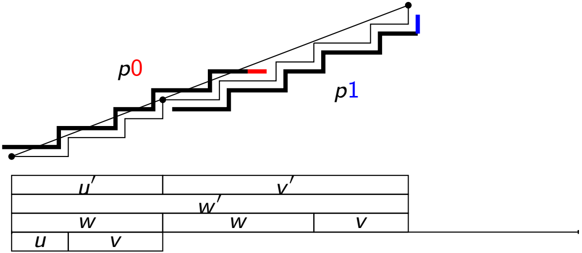


Nested prefixes

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 :



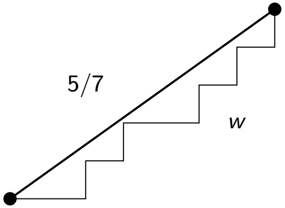
Corollary

Let word $w = (u, v)$ and $v = p1$, then $p0$ is a prefix of w.

Lexicographic order

Property

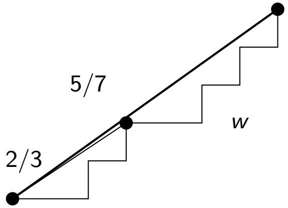
Lexicographic order on Christoffel words correspond to the order on the slope



Lexicographic order

Property

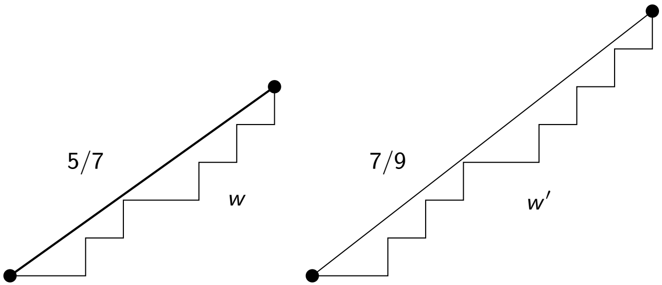
Lexicographic order on Christoffel words correspond to the order on the slope



Lexicographic order

Property

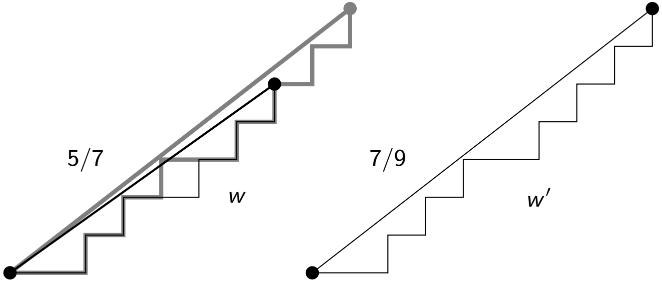
Lexicographic order on Christoffel words correspond to the order on the slope



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. $abaab$ is not Lyndon, since $aab <_{\mathbf{Lex}} abaab$.
3. $aabaab$ is not Lyndon, since $aab <_{\mathbf{Lex}} aabaab$.

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. $abaab$ is not Lyndon, since $aab <_{\mathbf{Lex}} abaab$.
3. $aabaab$ is not Lyndon, since $aab <_{\mathbf{Lex}} aabaab$.

Theorem ([Chen, Fox, Lyndon 58])

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

Example :

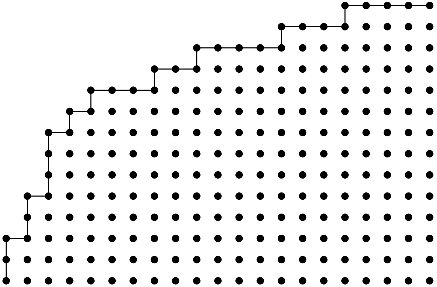
$$\begin{aligned} & 1\,1\,0\,1\,1\,0\,1\,1\,0\,0\,1\,0\,0\,1\,1\,0\,0\,0 \\ = \; & 1 \cdot 1 \cdot 0\,1\,1 \cdot 0\,1\,1 \cdot 0\,0\,1\,0\,0\,1\,1 \cdot 0 \cdot 0 \cdot 0 \\ = \; & (1)^2 \cdot (0\,1\,1)^2 \cdot (0\,0\,1\,0\,0\,1\,1)^1 \cdot (0)^3. \end{aligned}$$

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.



$$1101101110101000100100001000010000$$

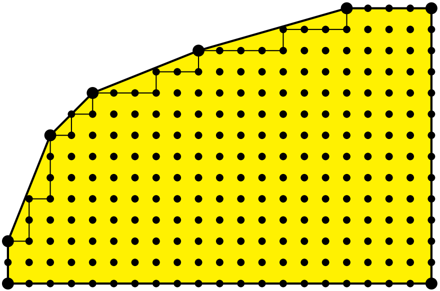
$$=(1)^2 \cdot 0110111 \cdot (01)^2 \cdot 001001 \cdot 000010001 \cdot (0)^4$$

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

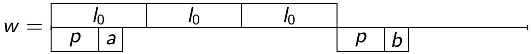
Recursive computation of the First Lyndon Prefix (FLF),

$$w = \boxed{l_0}\boxed{l_0}\boxed{l_0}\rule{4cm}{0.4pt}$$

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

Duval algorithm

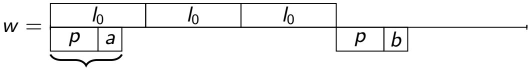
Recursive computation of the First Lyndon Prefix (FLF),



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),



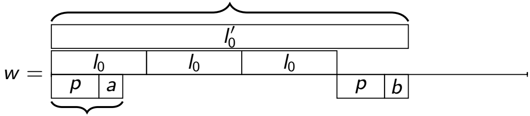
If $a > b$ then the Lyndon fact. starts by l_0^k

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),

if $a < b$ then $l_0^k pb$ is a Lyndon



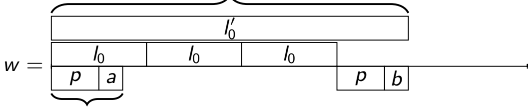
If $a > b$ then the Lyndon fact. starts by l_0^k

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),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

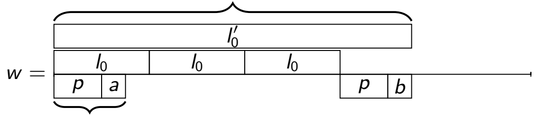
$$\underbrace{0}_{l_0}\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

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),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

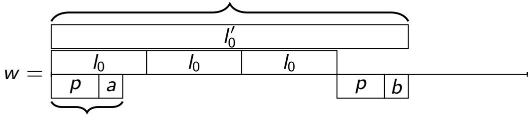
$$\begin{array}{cccccccccccccccccccccc} \downarrow & \downarrow \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ l_0 \end{array}$$

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),

if $a < b$ then $l_0^k pb$ is a Lyndon



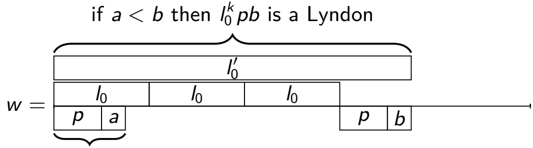
If $a > b$ then the Lyndon fact. starts by l_0^k

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

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),



If $a > b$ then the Lyndon fact. starts by l_0^k

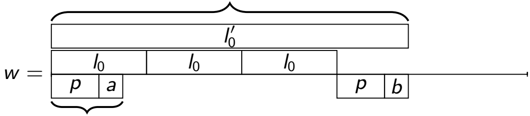
$$\underbrace{\overset{\downarrow}{0}\ 0\ 1}_{l_0}\ \overset{\downarrow}{0}\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

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),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

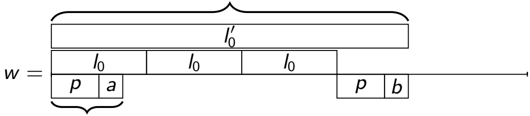
$$\underbrace{0\ 0\ 1}_{l_0}\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

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),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

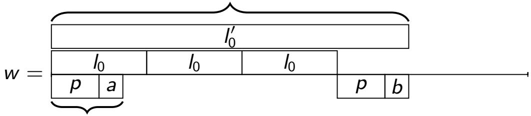
$$\underbrace{0\ 0\ 1}_{l_0}\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

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),

if $a < b$ then $l_0^k pb$ is a Lyndon



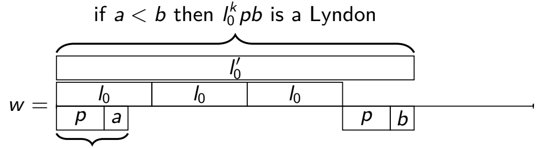
If $a > b$ then the Lyndon fact. starts by l_0^k

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

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),



If $a > b$ then the Lyndon fact. starts by l_0^k

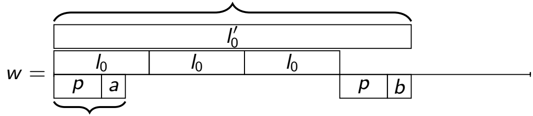
$$\underbrace{0\ 0\ 1}_{l_0}\ \underbrace{0\ 0\ 1}_{l_0}\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

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),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

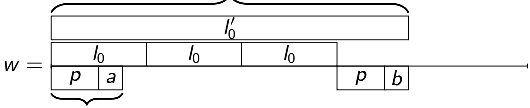
$$\underbrace{\overset{\downarrow}{0}\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ \overset{\downarrow}{0}\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

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),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

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

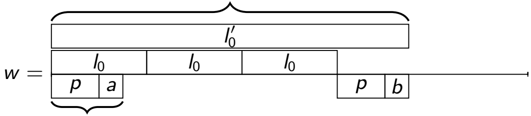
$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}$

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),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

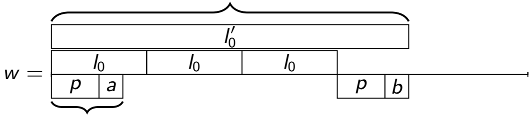
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

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),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

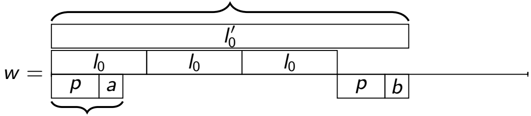
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

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),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

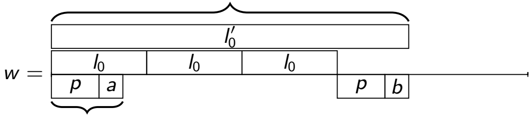
$$\underbrace{0\ 0\ 1\ 0\ \overset{\downarrow}{0}\ 1\ 0\ 1}_{l_0}\ 0\ 0\ 1\ 0\ \overset{\downarrow}{0}\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

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),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

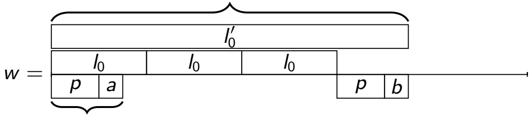
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

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),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

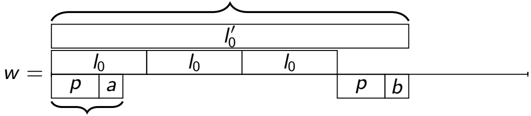
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ \overset{\downarrow}{0}\ 1}_{l_0}\ 0\ 0\ 1\ 0\ 0\ 1\ \overset{\downarrow}{0}\ 0\ 0\ 1\ 0\ 0\ 1$$

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),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

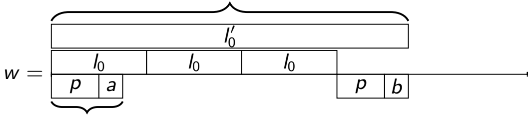
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ \overset{\downarrow}{1}}_{l_0}\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ \overset{\downarrow}{0}\ 0\ 1\ 0\ 0\ 1$$

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.

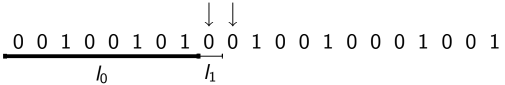
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

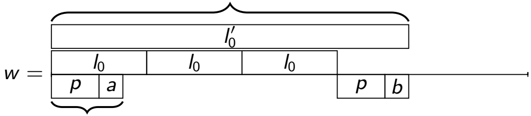


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

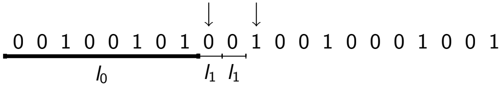
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

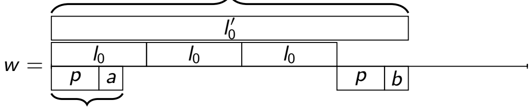


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

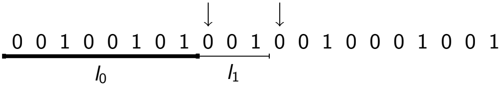
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

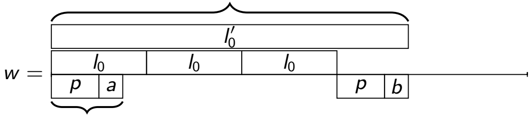


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

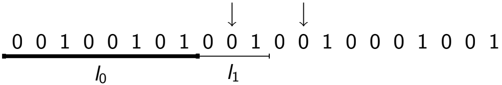
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

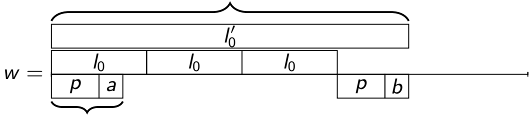


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

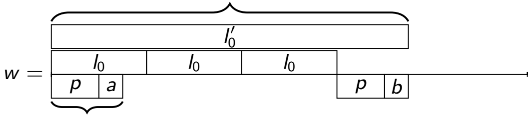
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ \underbrace{0\ 0\ \overset{\downarrow}{1}}_{l_1}\ 0\ 0\ \overset{\downarrow}{1}\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

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.

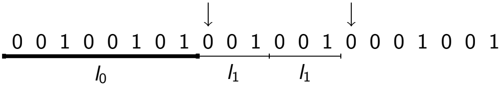
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k



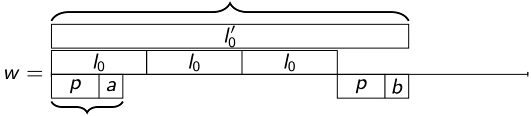
1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Definition
Periodic structure
Christoffel words
Digital convexity test

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

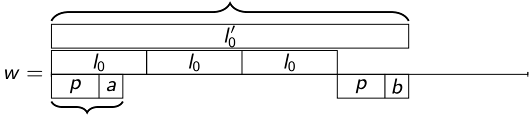
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ \underbrace{0\ 0\ 1}_{l_1}\ \underbrace{0\ 0\ 1}_{l_1}\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

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.

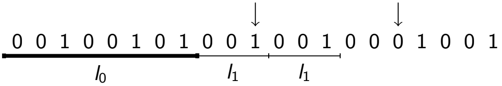
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

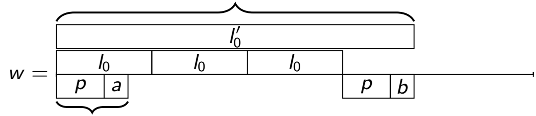


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

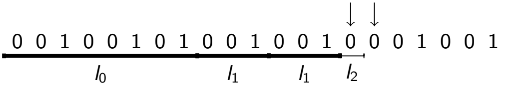
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

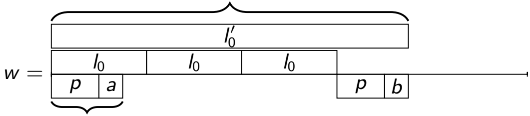


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

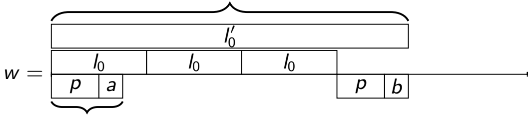
$$\begin{array}{cccccccccccccccccccccc} & & & & & & & & & & & & & & \downarrow & & \downarrow & & & & \\ 0&0&1&0&0&1&0&1&0&0&1&0&0&1&0&0&0&1&0&0&1 \\ \multicolumn{8}{c}{l_0} & \multicolumn{3}{c}{l_1} & \multicolumn{3}{c}{l_1} & l_2 & l_2 & & & & & \end{array}$$

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.

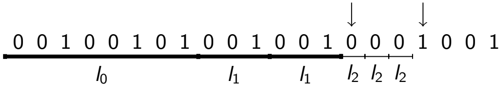
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

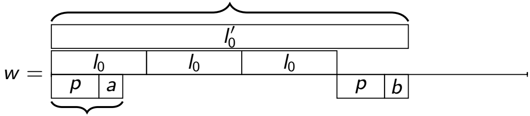


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

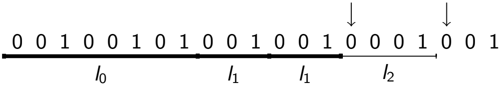
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

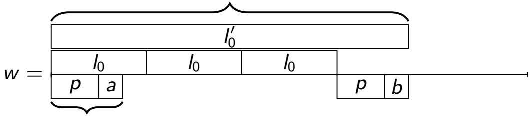


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

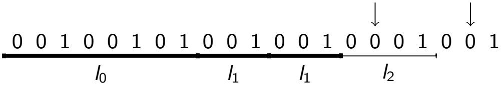
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

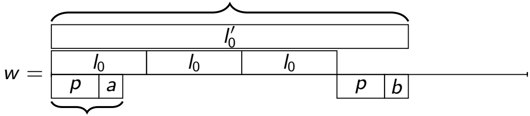


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

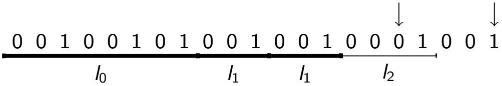
Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

if $a < b$ then $l_0^k pb$ is a Lyndon



If $a > b$ then the Lyndon fact. starts by l_0^k

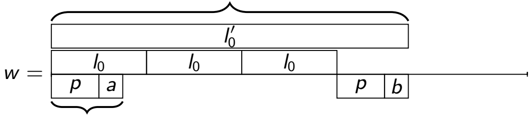


1. Let l_0 be a Lyndon prefix and k be it's number of repetitions.
2. Identify at the first letter that is not that same than in l_0.
3. If its smaller than l_0 is FLF, otherwise, $l_0^k pb$ is a Lyndon word.

Duval algorithm

Recursive computation of the First Lyndon Prefix (FLF),

if $a < b$ then $l_0^k pb$ is a Lyndon



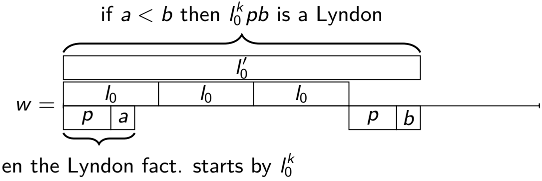
If $a > b$ then the Lyndon fact. starts by l_0^k

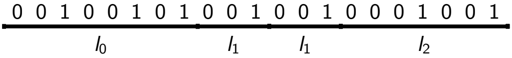
$$\underbrace{0\ 0\ 1\ 0\ 0\ 1\ 0\ 1}_{l_0}\ \underbrace{0\ 0\ 1}_{l_1}\ \underbrace{0\ 0\ 1}_{l_1}\ \underbrace{0\ 0\ 0\ 1\ 0\ 0\ 1}_{l_2}$$

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),





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.

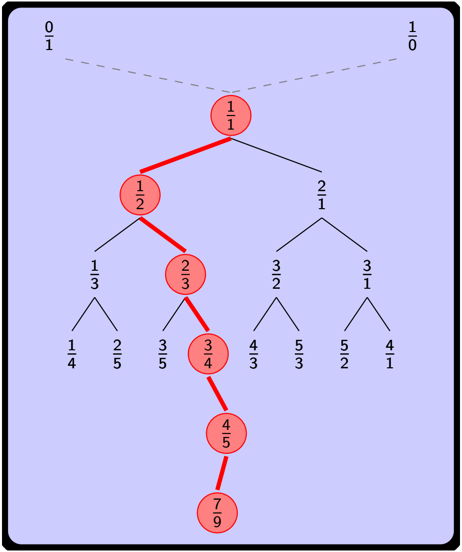
   When comparing two different letters, let $l_0 = (u, v)$:
   - if $|pb| = |v|$ then
     $a = 0$ and $b = 1$ and l_0' is a Christoffel word.
   - if $|pb| \neq |v|$ and $a = 1$ and $b = 0$ then
     l_0 is the first edge of the convex hull.
   - if $|pb| \neq |v|$ and $a = 0$ and $b = 1$ then
     Shape is not convex.

Part II

Construction guided by Euclid

Euclid Algorithm

Stern-Brocot tree



Euclid algorithm	Approximation
$(7, 9)$	$(1, 1)$
$\downarrow$	$\downarrow$
$(7, 2)$	$(1, 2)$
$\downarrow$	$\downarrow$
$(5, 2)$	$(2, 3)$
$\downarrow$	$\downarrow$
$(3, 2)$	$(3, 4)$
$\downarrow$	$\downarrow$
$(1, 2)$	$(4, 5)$
$\downarrow$	$\downarrow$
$(1, 1)$	$(7, 9)$

Matricial view

	Euclid algorithm	Approx.
n	v_n	a_n
0	$(\underline{7}, 9)$	$(1, 1)$
	$\downarrow$	$\downarrow$
1	$(7, \underline{2})$	$(1, 2)$
	$\downarrow$	$\downarrow$
2	$(5, \underline{2})$	$(2, 3)$
	$\downarrow$	$\downarrow$
3	$(3, \underline{2})$	$(3, 4)$
	$\downarrow$	$\downarrow$
4	$(\underline{1}, 2)$	$(4, 5)$
	$\downarrow$	$\downarrow$
5	$(1, 1)$	$(7, 9)$

Euclid algorithm
Given a vector (x, y), return

- $\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$ if $x < y$,
- $\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$ if $x > y$,
- **stop** if $x = y$.

Given a vector $v \in (\mathbb{N} \setminus \{0\})^2$, let :

- $v_0 = v$,
- For all $n \geq 1$: $\begin{cases} M_n = \textbf{Euclid}(v_{n-1}) \\ v_n = M_n v_{n-1}. \end{cases}$

Matricial view

	Euclid algorithm	Approx.
n	v_n	a_n
0	$(\underline{7}, 9)$	$(1, 1)$
	$\downarrow$	$\downarrow$
1	$(7, \underline{2})$	$(1, 2)$
	$\downarrow$	$\downarrow$
2	$(5, \underline{2})$	$(2, 3)$
	$\downarrow$	$\downarrow$
3	$(3, \underline{2})$	$(3, 4)$
	$\downarrow$	$\downarrow$
4	$(\underline{1}, 2)$	$(4, 5)$
	$\downarrow$	$\downarrow$
5	$(1, 1)$	$(7, 9)$

Euclid algorithm
Given a vector (x, y), return

- $\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$ if $x < y$,
- $\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$ if $x > y$,
- **stop** if $x = y$.

Given a vector $v \in (\mathbb{N} \setminus \{0\})^2$, let :

- $v_0 = v$,
- For all $n \geq 1$: $\begin{cases} M_n = \textbf{Euclid}(v_{n-1}) \\ v_n = M_n v_{n-1}. \end{cases}$

Property

- $v_n = M_n M_{n-1} \cdots M_1 v$
- $a_n = M_1^{-1} M_2^{-1} \cdots M_n^{-1} \binom{1}{1}$

Matricial view

Lemma

Let A, B, C be Christoffel words such that $C = (A, B)$ and $\overrightarrow{C} = a_n$. Let $\overrightarrow{A} = (A_x, A_y)$, $\overrightarrow{B} = (B_x, B_y)$, then:

$$M_1^\top M_2^\top \cdots M_n^\top = \begin{bmatrix} A_x & -B_x \\ -A_y & B_y \end{bmatrix}$$

Proof. By recurrence. True for $n = 0$, $\mathrm{Id} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Suppose true for n,

$$\begin{array}{ccc} & (A, B) & \\ \swarrow & & \searrow \\ (A, AB) & & (AB, B) \end{array}$$

$$M_1^\top \cdots M_{n+1}^\top = \begin{bmatrix} A_x & -B_x \\ -A_y & B_y \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} A_x & -A_x - B_x \\ -A_y & A_y + B_y \end{bmatrix}.$$

Matricial view

Lemma

Let A, B, C be Christoffel words such that $C = (A, B)$ and $\overrightarrow{C} = a_n$. Let $\overrightarrow{A} = (A_x, A_y)$, $\overrightarrow{B} = (B_x, B_y)$, then:

$$M_1^\top M_2^\top \cdots M_n^\top = \begin{bmatrix} A_x & -B_x \\ -A_y & B_y \end{bmatrix}$$

Proof. By recurrence. True for $n = 0$, $\mathrm{Id} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Suppose true for n,

$$\begin{tikzcd} & (A,B) \arrow[dl] \arrow[dr, dash] & \\ (A,AB) & & (AB,B) \end{tikzcd}$$

$$M_1^\top \cdots M_{n+1}^\top = \begin{bmatrix} A_x & -B_x \\ -A_y & B_y \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} A_x & -A_x - B_x \\ -A_y & A_y + B_y \end{bmatrix}.$$

$$M_1^\top \cdots M_n^\top e_1 = (A_x, -A_y)$$
$$M_1^\top \cdots M_n^\top e_2 = (-B_x, B_y)$$

The Translation-Union Construction

Construction
[Domenjoud, Vuillon 12],
[Berthé, Jamet, Jolivet, P. 2013]

Let $v_0 = v$, $B_0 = \{\mathbf{0}\}$ and for all $n \geq 1$ let :

M_n : the matrix selected from v_{n-1},

$v_n = M_n v_{n-1}$

δ_n : the index of the coordinate of v_{n-1} that is subtracted,

$T_n = M_1^\top \cdots M_n^\top e_{\delta_n}$, *(translation)*

$B_n = B_{n-1} \cup (T_n + B_{n-1})$, *(body)*

$H_n = \sum_{i \in \{1,\ldots,n\}} T_i$, *(highest point)*

$L_n = H_n + \{M_1^\top \cdots M_n^\top e_i\}$. *(legs)*

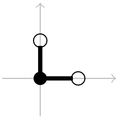
Note that:

$H_n \in B_n$,

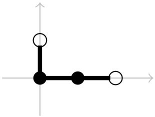
$L_n \cap B_n = \emptyset$.

$\bullet \in B_n$, $\circ \in L_n$

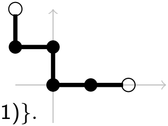
$v_0 = (2,3)$,
$a_0 = (1,1)$
$H_0 = (0,0)$,
$L_0 = \{(1,0),(0,1)\}$.



$v_1 = (2,1), \delta_1 = 1$
$a_1 = (1,2)$
$T_1 = (1,0)$
$H_1 = (1,0)$,
$L_1 = \{(2,0),(0,1)\}$.



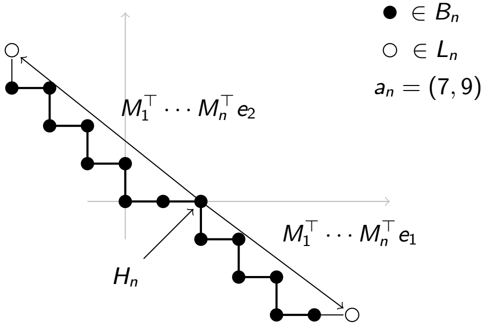
$v_2 = (1,1), \delta_2 = 2$
$a_2 = (2,3)$
$T_2 = (-1,1)$
$H_2 = (0,1)$,
$L_1 = \{(2,-1),(-1,1)\}$.



The Translation-Union Construction

Property

The points of $B_n \cup L_n$ for the Christoffel word of vector a_n.
Moreover, let $\{x, y\} = L_n$ then $\langle x, a_n \rangle = \langle y, a_n \rangle$.



Part III

Generalization to higher dimensions

3D continued fraction algorithms

Euclid algorithm : given two number subtract the smaller to the larger.

$(7,9) \to (7,2) \to (5,2) \to (3,2) \to (1,2) \to (1,1) \to (1,0)$

3D continued fraction algorithms

Euclid algorithm : given two number subtract the smaller to the larger.
$(7,9) \to (7,2) \to (5,2) \to (3,2) \to (1,2) \to (1,1) \to (1,0)$

Given three numbers :

- **Selmer** : subtract the smallest to the largest.
  $(3,7,5) \to (3,4,5) \to (3,4,2) \to (3,2,2) \to (1,2,2) \to (1,2,0) \to (1,1,0) \to (1,0,0)$.
- **Brun** : subtract the second largest to the largest.
  $(3,7,5) \to (3,2,5) \to (3,2,2) \to (1,2,2) \to (1,2,0) \to (1,1,0) \to (1,0,0)$.
- **Fully subtractive** : subtract the smallest to the two others.
  $(3,7,5) \to (3,4,2) \to (1,2,2) \to (1,1,1) \to (1,0,0)$.
- **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).
  $(3,7,5) \to \mathrm{impossible}$.
- ...

The Translation-Union Construction

Construction

Let $v_0 = v$, $B_0 = \{\mathbf{0}\}$ and for all $n \geq 1$ let :

M_n : the matrix selected from v_{n-1},

$v_n = M_n v_{n-1}$

δ_n : the index of the coordinate of v_{n-1} that is subtracted,

$$T_n = M_1^\top \cdots M_n^\top e_{\delta_n}, \qquad (\textit{translation})$$

$$B_n = B_{n-1} \cup (T_n + B_{n-1}), \qquad (\textit{body})$$

$$H_n = \sum_{i \in \{1,\ldots,n\}} T_i, \qquad (\textit{highest point})$$

$$L_n = H_n + \{M_1^\top \cdots M_n^\top e_i\}. \qquad (\textit{legs})$$

The Translation-Union Construction

Construction

Let $v_0 = v$, $B_0 = \{\mathbf{0}\}$ and for all $n \geq 1$ let :

M_n : the matrix selected from v_{n-1},

$v_n = M_n v_{n-1}$

δ_n : the index of the coordinate of v_{n-1} that is subtracted,

$$T_n = M_1^\top \cdots M_n^\top e_{\delta_n}, \qquad (\textit{translation})$$

$$B_n = B_{n-1} \cup (T_n + B_{n-1}), \qquad (\textit{body})$$

$$H_n = \sum_{i \in \{1,\dots,n\}} T_i, \qquad (\textit{highest point})$$

$$L_n = H_n + \{M_1^\top \cdots M_n^\top e_i\}. \qquad (\textit{legs})$$

Property

If the action of M_n is to subtract a coordinate to at least one other coordinate while keeping it positive, then $B_n \in \mathcal{P}(v, 0)$.

Proof : $\langle T_n, v\rangle = \langle M_1^\top \dots M_n^\top e_{\delta_n}, v\rangle = \langle e_{\delta_n}, M_n \cdots M_1 v\rangle = \langle e_{\delta_n}, v_n\rangle$ is equal to the value of the coordinate that is subtracted.

Let $x \in B_n$, then $x = \sum_{i \in I} T_i$ for some $I \subset \{1, \cdots, n\}$ and

$$0 \leq \langle x, v\rangle < \|v\|_1$$

Construction using fully Subtractive

The **fully subtractive** algorithm :

Subtract the smallest coordinate to the two others.

The matrices are :

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

Construction using fully Subtractive

The **fully subtractive** algorithm :

Subtract the smallest coordinate to the two others.

The matrices are :

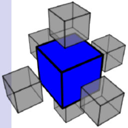
$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

Definition
Let $\mathcal{K}$ be the set of vectors v such $\mathbf{FS}^N(v) = (1,1,1)$ for some $N \geq 1$.

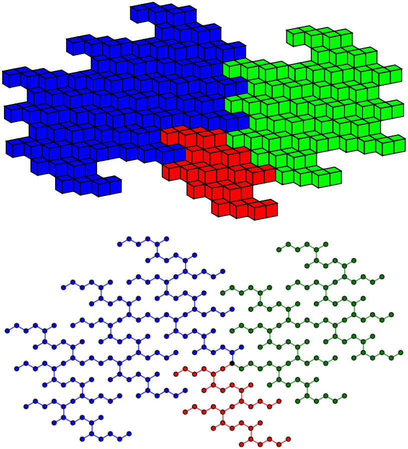
- $\mathcal{K} \ni (1,2,2) \xrightarrow{\mathbf{FS}} (1,1,1)$
- $\mathcal{K} \not\ni (2,2,5) \xrightarrow{\mathbf{FS}} (0,2,3)$

Tree structure

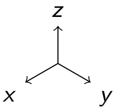
Theorem ([Domenjoud, Vuillon 12])
When using the fully subtractive algorithm, the graph of B_n is a tree.

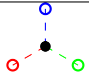
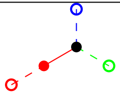
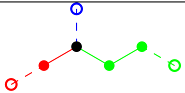
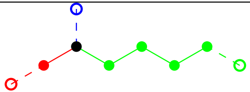
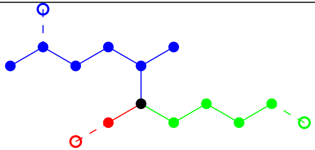


Example : $v = (136, 184, 249) \in \mathcal{K}$



Recursive construction with Fully Subtractive



$B_n \cup L_n$	Approx.	Fully subtractive algorithm
	$(1,1,1)$	$(6,8,11)$
	$(1,2,2)$	$(6,2,5)$
	$(2,3,4)$	$(4,2,3)$
	$(3,4,6)$	$(2,2,1)$
	$(6,8,11)$	$(1,1,1)$

Recursive construction with Fully Subtractive

Property

Using Fully Subtractive on $v \in \mathcal{K}$, let N be such that $v_N = (1, 1, 1)$ and so $a_N = v$:

1. $B_N \cup L_N$ *is connected.*
2. B_N *has exactly one point at each height from* 0 *to* $\left\lfloor \frac{\|v\|_1}{2} \right\rfloor - 1$
3. *All points of* L_N *have height* $\left\lfloor \frac{\|v\|_1}{2} \right\rfloor$

Recursive construction with Fully Subtractive

Property

Using Fully Subtractive on $v \in \mathcal{K}$, let N be such that $v_N = (1,1,1)$ and so $a_N = v$:

1. $B_N \cup L_N$ *is connected.*
2. B_N *has exactly one point at each height from* 0 *to* $\left\lfloor \frac{\|v\|_1}{2} \right\rfloor - 1$
3. *All points of* L_N *have height* $\left\lfloor \frac{\|v\|_1}{2} \right\rfloor$

1. B_n is a tree.

Recursive construction with Fully Subtractive

Property

Using Fully Subtractive on $v \in \mathcal{K}$, let N be such that $v_N = (1,1,1)$ and so $a_N = v$:

1. $B_N \cup L_N$ *is connected.*
2. B_N *has exactly one point at each height from* 0 *to* $\left\lfloor \frac{\|v\|_1}{2} \right\rfloor - 1$
3. *All points of* L_N *have height* $\left\lfloor \frac{\|v\|_1}{2} \right\rfloor$

1. B_n is a tree.
2. $v = v_0 \xrightarrow{\textbf{FS}} v_1 \xrightarrow{\textbf{FS}} \cdots \xrightarrow{\textbf{FS}} v_N = (1,1,1)$

   The heigth of each T_i is equal to the coordinate that has been subtracted to the two other coordinates.

   $\|v_n\|_1 = \|v_{n-1}\|_1 - 2\langle T_n, v\rangle$.

Recursive construction with Fully Subtractive

Property

Using Fully Subtractive on $v \in \mathcal{K}$, let N be such that $v_N = (1,1,1)$ and so $a_N = v$:

1. $B_N \cup L_N$ *is connected.*
2. B_N *has exactly one point at each height from* 0 *to* $\left\lfloor \frac{\|v\|_1}{2} \right\rfloor - 1$
3. *All points of* L_N *have height* $\left\lfloor \frac{\|v\|_1}{2} \right\rfloor$

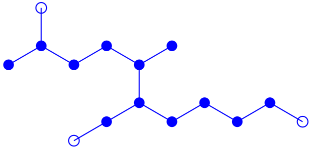
1. B_n is a tree.
2. $v = v_0 \xrightarrow{\textbf{FS}} v_1 \xrightarrow{\textbf{FS}} \cdots \xrightarrow{\textbf{FS}} v_N = (1,1,1)$

   The heigth of each T_i is equal to the coordinate that has been subtracted to the two other coordinates.

   $\|v_n\|_1 = \|v_{n-1}\|_1 - 2\langle T_n, v\rangle$.
3. $L_n = H_n + \{M_1^T \cdots M_n^T e_i\}$ and
   $\langle M_1^T \cdots M_N^T e_i, v\rangle = \langle e_i, M_N \cdots M_1 v\rangle = \langle e_i, v_N\rangle = \langle e_i, (1,1,1)\rangle = 1.$

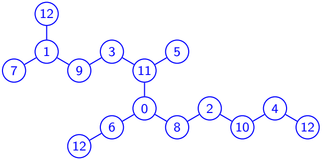
From pattern to digital plane

$kev = (6, 8, 11), \left\lfloor \frac{\|v\|_1}{2} \right\rfloor = 12$



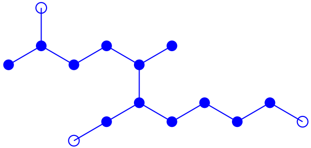
From pattern to digital plane

$kev = (6, 8, 11), \left\lfloor \frac{\|v\|_1}{2} \right\rfloor = 12$



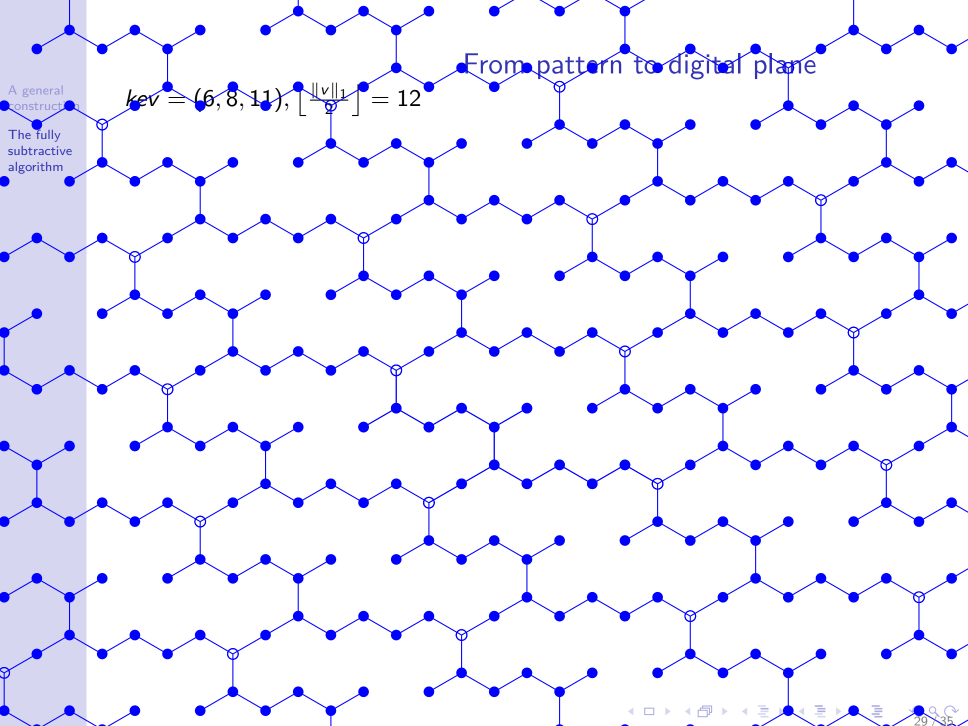
From pattern to digital plane

$kev = (6, 8, 11), \left\lfloor \frac{\|v\|_1}{2} \right\rfloor = 12$



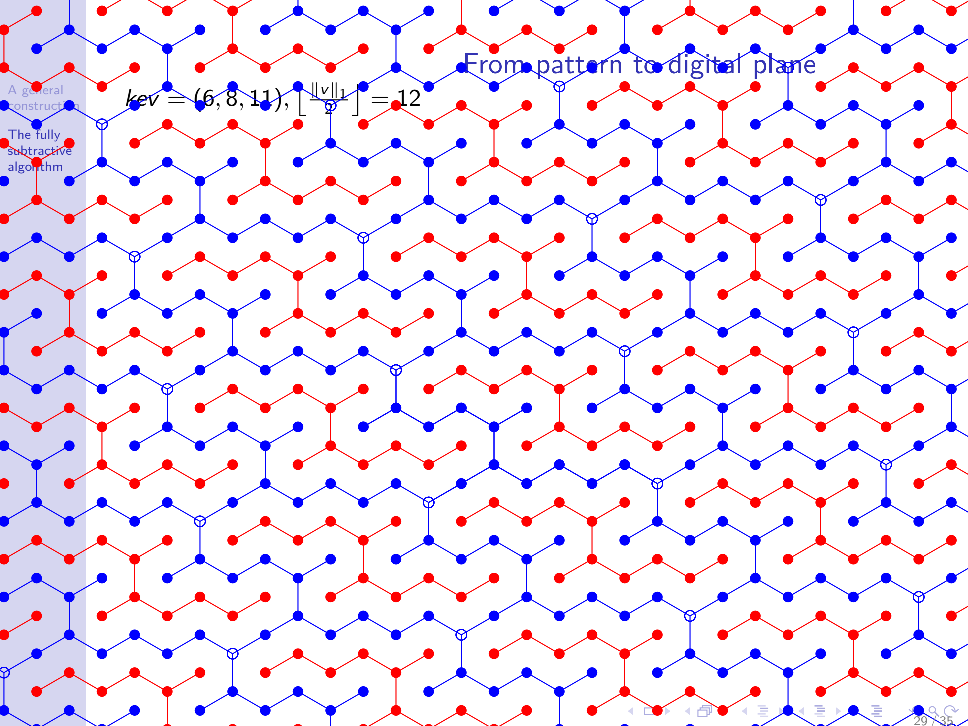
From pattern to digital plane

$kev = (6, 8, 11), \left\lceil \frac{\|v\|_1}{2} \right\rceil = 12$



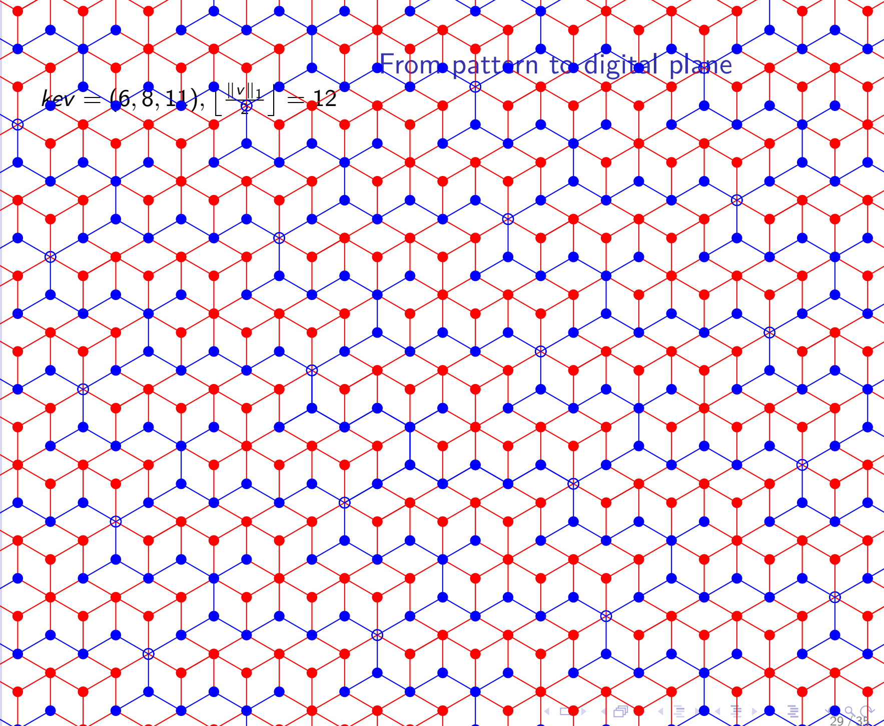
From pattern to digital plane

$kev = (6, 8, 11), \left\lfloor \frac{\|v\|_1}{2} \right\rfloor = 12$



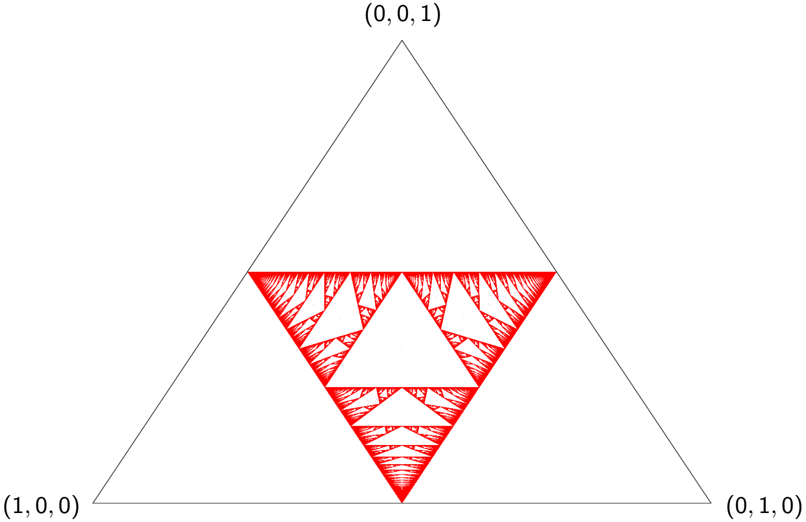
From pattern to digital plane

$kev = (6, 8, 11), \left\lfloor \frac{\|v\|_1}{2} \right\rfloor = 12$



The set $\mathcal{K}$

$$v \xrightarrow{\mathsf{FS}} \cdots \xrightarrow{\mathsf{FS}} (1,1,1)$$



Vectors not in $\mathcal{K}$

Let $v \in (\mathbb{N} \setminus \{0\})^3$ such that $v \notin \mathcal{K}$, then either :

1. $\mathbf{FS}^n(v) = (g, g, g)$ with $g \geq 2$.
2. $\mathbf{FS}^n(v) = (a, a, b)$ with $a < b$ so that $\mathbf{FS}((a, a, b)) = (0, a, b - a)$.
3. $\mathbf{FS}^n(v) = (a, b, c)$ with $a + b \leq c$.

Vectors not in $\mathcal{K}$

Let $v \in (\mathbb{N} \setminus \{0\})^3$ such that $v \notin \mathcal{K}$, then either :

1. $\mathbf{FS}^n(v) = (g, g, g)$ with $g \geq 2$.
2. $\mathbf{FS}^n(v) = (a, a, b)$ with $a < b$ so that $\mathbf{FS}((a, a, b)) = (0, a, b - a)$.
3. $\mathbf{FS}^n(v) = (a, b, c)$ with $a + b \leq c$.

Solution:

1. Then $g = \gcd(v)$, use $v/g \in \mathcal{K}$.

Vectors not in $\mathcal{K}$

Let $v \in (\mathbb{N} \setminus \{0\})^3$ such that $v \notin \mathcal{K}$, then either :

1. $\mathbf{FS}^n(v) = (g, g, g)$ with $g \geq 2$.
2. $\mathbf{FS}^n(v) = (a, a, b)$ with $a < b$ so that $\mathbf{FS}((a, a, b)) = (0, a, b - a)$.
3. $\mathbf{FS}^n(v) = (a, b, c)$ with $a + b \leq c$.

Solution:

1. Then $g = \gcd(v)$, use $v/g \in \mathcal{K}$.
2. Do not use **FS**. . .
3. Do not use **FS**. . .

Vectors not in $\mathcal{K}$

Let $v \in (\mathbb{N} \setminus \{0\})^3$ such that $v \notin \mathcal{K}$, then either :

1. $\mathbf{FS}^n(v) = (g, g, g)$ with $g \geq 2$.
2. $\mathbf{FS}^n(v) = (a, a, b)$ with $a < b$ so that $\mathbf{FS}((a, a, b)) = (0, a, b - a)$.
3. $\mathbf{FS}^n(v) = (a, b, c)$ with $a + b \leq c$.

Solution:

1. Then $g = \gcd(v)$, use $v/g \in \mathcal{K}$.
2. Do not use **FS**. . .
3. Do not use **FS**. . .

. . . ok but *what else* ?

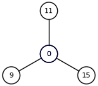
Vectors not in $\mathcal{K}$

Idea : Use hybrid algorithm, suppose $a \leq b \leq c$,

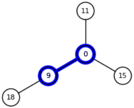
$$(a, b, c) = \begin{cases} \textbf{FS}((a,b,c)) \text{ if } a \neq b \text{ and } a + b \leq c, \\ \textbf{Brun}((a,b,c)) \text{ otherwise.} \end{cases}$$

Brun: subtract the second biggest coordinate to the biggest one.

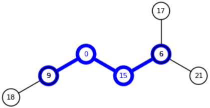
$(9, 15, 11)$



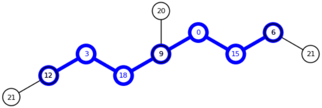
$\xrightarrow{\textbf{FS}} (9, 6, 2)$



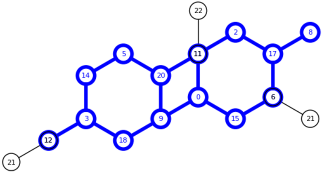
$\xrightarrow{\textbf{Brun}} (3, 6, 2)$



$\xrightarrow{\textbf{Brun}} (3, 3, 2)$



$\xrightarrow{\textbf{FS}} (1, 1, 2)$



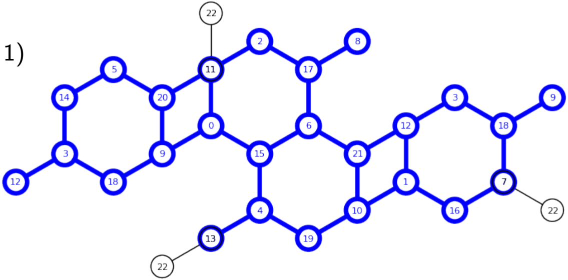
The hybrid **FS+Brun** algorithm

Property ([Lafrenière, Jamet, P.])

*Using the hybrid **FS+Brun** algorithm, for all vector* $v \in (\mathbb{N} \setminus \{0\})^3$

1. $\exists N$ *such that* $v_N = (1,1,1)$ *(or* gcd... *).*
2. *Vectors of* L_n *have same height, (providing period vectors).*
3. $B_n \cup L_n$ *is connected but in general not a tree.*
4. $\left\lfloor \frac{\|v\|_1}{2} \right\rfloor - 1 \leq \langle H_N \rangle < \|v\|_1.$
5. *There is a least one point at each height from* 0 *to* $\langle H_N \rangle$ *but in general no unicity.*

$\xrightarrow{\textbf{Brun}} (1,1,1)$



Conclusion

Good:

- Generalization of Christoffel words to higher dimensions.
- Construction is recursive and based on continued fraction algorithms.
- Construction of the periodic pattern of the digital plane for $\mathcal{K}$.

~~Problems:~~ Open questions :

- Provide a gcd algorithm that builds minimal patterns for $\mathcal{K}^{\mathsf{C}}$.
- Give a geometrical interpretation of the patterns produced by the hybrid algorihtm.
- Control the anisotropy of the patterns (avoid stretched forms in favor of *potato-likeness*).
- Apply recursive structure to image analysis algorithms.

Merci