Combinatorics of Garside monoids

Juan González-Meneses

Universidad de Sevilla

With Volker Gebhardt (U. Western Sydney)

Garside theory: State of the art and prospects

Cap Hornu, 30 may 2012.

M = Garside monoid

It has a **Lattice structure**

$a \preccurlyeq b \quad \Leftrightarrow \quad a$ is a prefix of b $\longrightarrow$ Unique gcd's ($\wedge$) and lcm's ($\vee$)

It has a **Garside element** Δ

The set of **simple elements** is **finite** $\longrightarrow$ Finite number of atoms

prefixes of Δ

…and maybe M is **homogeneous**

homogeneous relations w.r.t. the atoms

M = Homogeneous Garside monoid

All representatives have the same length

How many elements are there of length k?

$$\alpha_k = |M_k| = |\{x \in M;\ |x| = k\}|$$

$$\alpha_0 = 1$$

$\alpha_1 = n$ (number of atoms)

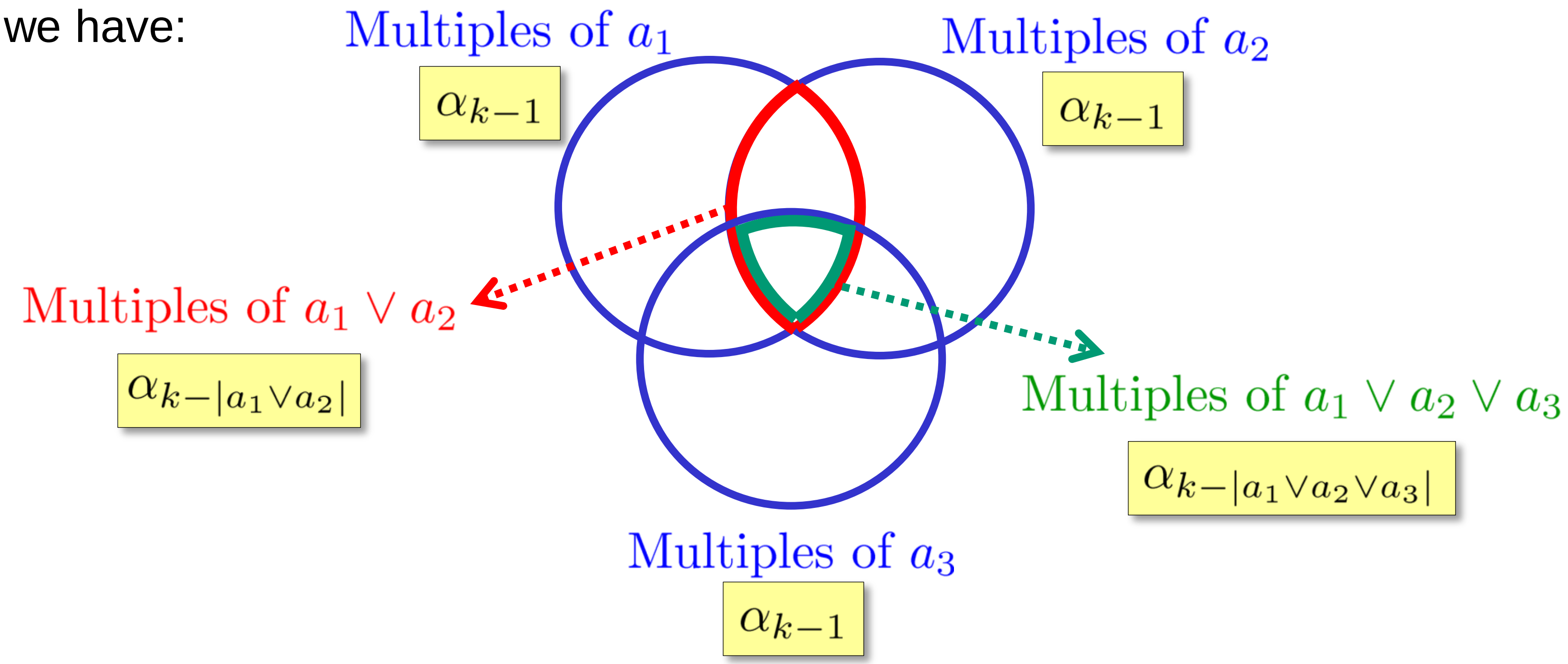
$$\vdots$$

and then?

M = Homogeneous Garside monoid

Atoms = $\{a_1, a_2, a_3\}$

In M_k we have:



M = Homogeneous Garside monoid

Atoms = $\{a_1, a_2, a_3\}$

Therefore, by the inclusion-exclusion principle:

$$\alpha_k = \alpha_{k-1} + \alpha_{k-1} + \alpha_{k-1} - \alpha_{k-|a_1 \vee a_2|} - \alpha_{k-|a_1 \vee a_3|} - \alpha_{k-|a_2 \vee a_3|} + \alpha_{k-|a_1 \vee a_2 \vee a_3|}$$

The spherical growth function is the inverse of a polynomial

Delinge (1972)

Example: In the monoid of 4-strands braids, $\alpha_k = 3\alpha_{k-1} - \alpha_{k-2} - 2\alpha_{k-3} + \alpha_{k-6}$

Spherical growth function = $\dfrac{1}{1 - 3t + t^2 + 2t^3 - t^6}$

New formula

Artin-Tits monoid of type $A_{n\text{-}1}$

(monoid of n-strands braids)

Theorem (GM, 2011) The spherical growth function of this monoid is:

$$g(t) = \begin{vmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ t & 1 & 1 & 0 & \ddots & \vdots \\ t^3 & t & 1 & 1 & \ddots & 0 \\ t^6 & t^3 & t & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & 1 \\ t^{\binom{n}{2}} & \cdots & t^6 & t^3 & t & 1 \end{vmatrix}^{-1}$$

New formula

Artin-Tits monoid of type $A_{n\text{-}1}$

(monoid of n-strands braids)

Example: For n = 8

$$g(t) = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ t & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ t^3 & t & 1 & 1 & 0 & 0 & 0 & 0 \\ t^6 & t^3 & t & 1 & 1 & 0 & 0 & 0 \\ t^{10} & t^6 & t^3 & t & 1 & 1 & 0 & 0 \\ t^{15} & t^{10} & t^6 & t^3 & t & 1 & 1 & 0 \\ t^{21} & t^{15} & t^{10} & t^6 & t^3 & t & 1 & 1 \\ t^{28} & t^{21} & t^{15} & t^{10} & t^6 & t^3 & t & 1 \end{vmatrix}^{-1}$$

Exponents are $\binom{i}{2}$

New formula

Artin-Tits monoid of type B_n

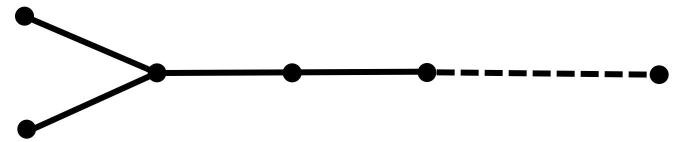
$$g(t) = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ t^2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ t^4 & t & 1 & 1 & 0 & 0 & 0 & 0 \\ t^9 & t^3 & t & 1 & 1 & 0 & 0 & 0 \\ t^{16} & t^6 & t^3 & t & 1 & 1 & 0 & 0 \\ t^{25} & t^{10} & t^6 & t^3 & t & 1 & 1 & 0 \\ t^{36} & t^{15} & t^{10} & t^6 & t^3 & t & 1 & 1 \\ t^{49} & t^{21} & t^{15} & t^{10} & t^6 & t^3 & t & 1 \end{vmatrix}^{-1}$$

Exponents are $\binom{i}{2}$

Exponents are i^2

New formula

Artin-Tits monoid of type D_n



$$g(t) = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2t - {\color{red}t^2} & 1 & 1 & 0 & 0 & 0 & 0 \\ 2t^3 - {\color{red}t^6} & t & 1 & 1 & 0 & 0 & 0 \\ 2t^6 - {\color{red}t^{12}} & t^3 & t & 1 & 1 & 0 & 0 \\ 2t^{10} - {\color{red}t^{20}} & t^6 & t^3 & t & 1 & 1 & 0 \\ 2t^{15} - {\color{red}t^{30}} & t^{10} & t^6 & t^3 & t & 1 & 1 \\ 2t^{21} - {\color{red}t^{42}} & t^{15} & t^{10} & t^6 & t^3 & t & 1 \end{vmatrix}^{-1}$$

Exponents are $\binom{i}{2}$

Exponents are $i(i-1)$

Growth rate

Spherical growth series: $g(t) = \dfrac{1}{p(t)}$ ← Polynomial

r = smallest root of $p(t)$ = radius of convergence

$\rho = \lim_{n\to\infty} \sqrt[n]{\alpha_n}$ = **Growth rate** of the monoid

$$\rho = \frac{1}{r}$$

If we know well the polynomial, we could say something about the growth rate.

Growth rate

Example: $M = A_2$ $\Longrightarrow$ $\rho =$ Golden ratio

(3-strands braid monoid)

$M = B_2$ $\Longrightarrow$ $\rho =$ Tribonacci constant

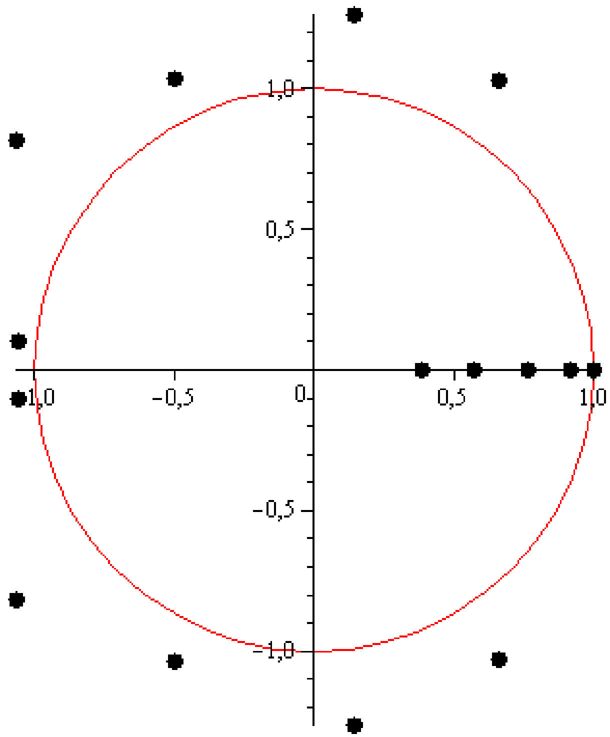
$M = I_2(p)$ $\Longrightarrow$ $\rho =$ Fibonacci constant of order p

Growth rate

For type A (braid monoids)

What is the smallest root of $\begin{vmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ t & 1 & 1 & 0 & \ddots & \vdots \\ t^3 & t & 1 & 1 & \ddots & 0 \\ t^6 & t^3 & t & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & 1 \\ t^{\binom{n}{2}} & \cdots & t^6 & t^3 & t & 1 \end{vmatrix}$?

n=5



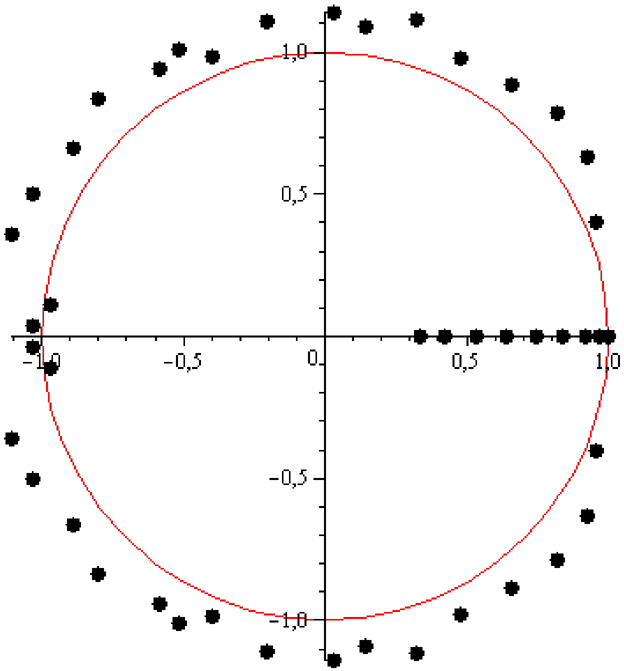
Juan González-Meneses

Growth rate

For type A (braid monoids)

What is the smallest root of
$$\begin{vmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ t & 1 & 1 & 0 & \ddots & \vdots \\ t^3 & t & 1 & 1 & \ddots & 0 \\ t^6 & t^3 & t & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & 1 \\ t^{\binom{n}{2}} & \cdots & t^6 & t^3 & t & 1 \end{vmatrix}$$
?

n=9

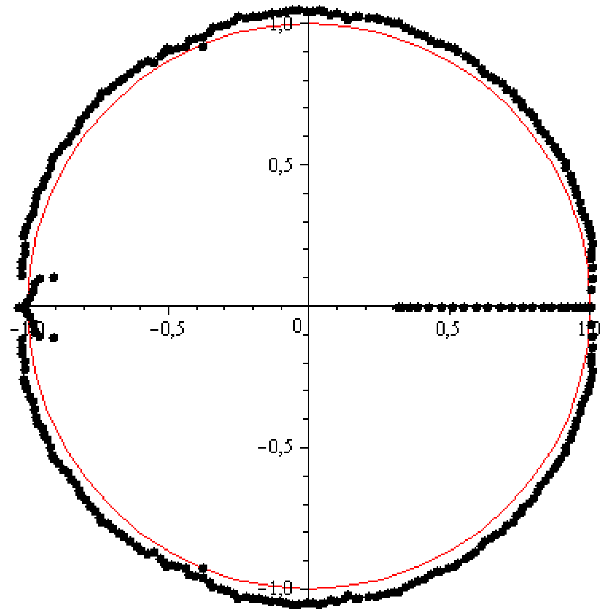


Growth rate

For type A (braid monoids)

What is the smallest root of
$$\begin{vmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ t & 1 & 1 & 0 & \ddots & \vdots \\ t^3 & t & 1 & 1 & \ddots & 0 \\ t^6 & t^3 & t & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & 1 \\ t^{\binom{n}{2}} & \cdots & t^6 & t^3 & t & 1 \end{vmatrix}$$
?

n=24



Conjecture:

When $n \to \infty$

$\rho \to 1 + \sqrt{5} \;= 2\varphi$

Second application

$$B_n^+ = A_{n-1} = \left\langle \sigma_1, \sigma_2, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{ll} \sigma_i\sigma_j = \sigma_j\sigma_i & |i-j|>1 \\ \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j & |i-j|=1 \end{array} \right\rangle$$

Problem: We want to generate a random positive braid of length k.

There can be many representatives of the same braid

Not all braids have the same number of representatives

Given a braid α, its Lex-representative, $w(\alpha)$, is the (lexicographically) smallest word representing α.

We generate Lex-representatives

Generating random braids

Teorem: (Gebhardt-GM, 2011) There is an algorithm which generates a random positive braid in ${B_n}^+$ of length k, having polynomial time and space complexity in n and k.

Complexity: $O(n^4 \ln n \; k^{\alpha+2})$ ← $\alpha = \log_2 3 \approx 1.585$

	n = 4	8	16	32	64	128	256	512
k = 4	1.09e0	1.33e0	1.76e0	2.07e0	2.45e0	2.95e0	3.58e0	8.03e0
8	4.62e0	6.39e0	5.79e0	6.10e0	1.26e1	2.19e1	3.13e1	4.84e1
16	2.20e1	5.10e1	4.56e1	7.52e1	9.73e1	1.35e2	2.31e2	2.42e2
32	1.07e2	3.81e2	4.63e2	6.70e2	8.81e2	1.01e3	1.15e3	1.79e3
64	4.84e2	2.07e3	4.55e3	6.72e3	5.72e3	7.94e3	7.61e3	7.72e3
128	2.01e3	9.55e3	3.30e4	5.24e4	5.70e4	1.03e5	1.13e5	1.04e5
256	8.42e3	4.27e4	1.90e5	4.17e5	6.06e5	1.20e6	1.81e6	2.06e6
512	4.36e4	1.78e5	9.70e5	3.27e6	5.85e6	1.23e7	2.56e7	3.29e7
1024	2.52e5	7.24e5	4.77e6	2.07e7	5.22e7	1.25e8	2.78e8	4.29e8

Time in μs to generate a braid

Ingredients of the algorithm

Suppose we want to write down a lex-representative w.

The first letter can be anyone. Say σ_4.

$$\boxed{w = \sigma_4 w_1}$$

$$\begin{array}{r} \sigma_4 \preccurlyeq w \\ \sigma_1 \not\preccurlyeq w \\ \sigma_2 \not\preccurlyeq w \\ \sigma_3 \not\preccurlyeq w \end{array} \Leftrightarrow \begin{array}{r} \sigma_4 \preccurlyeq w \\ \sigma_1 \vee \sigma_4 \not\preccurlyeq w \\ \sigma_2 \vee \sigma_4 \not\preccurlyeq w \\ \sigma_3 \vee \sigma_4 \not\preccurlyeq w \end{array} \Leftrightarrow \begin{array}{r} \sigma_4^{-1}(\sigma_1 \vee \sigma_4) \not\preccurlyeq w_1 \\ \sigma_4^{-1}(\sigma_2 \vee \sigma_4) \not\preccurlyeq w_1 \\ \sigma_4^{-1}(\sigma_3 \vee \sigma_4) \not\preccurlyeq w_1 \end{array} \Leftrightarrow \begin{array}{r} \sigma_1 \not\preccurlyeq w_1 \\ \sigma_2 \not\preccurlyeq w_1 \\ \sigma_3\sigma_4 \not\preccurlyeq w_1 \end{array}$$

Forbidden prefixes of w_1

Ingredients of the algorithm

Suppose we want to write down a lex-representative w.

$$\sigma_1 \not\preccurlyeq w_1 \qquad \sigma_2 \not\preccurlyeq w_1 \qquad \sigma_3\sigma_4 \not\preccurlyeq w_1$$

The first letter can be anyone. Say σ_4.

$$\boxed{w = \sigma_4 w_1}$$

$$\begin{array}{c} \sigma_4 \preccurlyeq w \\ \sigma_1 \not\preccurlyeq w \\ \sigma_2 \not\preccurlyeq w \\ \sigma_3 \not\preccurlyeq w \end{array} \Leftrightarrow \begin{array}{r} \sigma_4 \preccurlyeq w \\ \sigma_1 \vee \sigma_4 \not\preccurlyeq w \\ \sigma_2 \vee \sigma_4 \not\preccurlyeq w \\ \sigma_3 \vee \sigma_4 \not\preccurlyeq w \end{array} \Leftrightarrow \begin{array}{c} \sigma_4^{-1}(\sigma_1 \vee \sigma_4) \not\preccurlyeq w_1 \\ \sigma_4^{-1}(\sigma_2 \vee \sigma_4) \not\preccurlyeq w_1 \\ \sigma_4^{-1}(\sigma_3 \vee \sigma_4) \not\preccurlyeq w_1 \end{array} \Leftrightarrow$$

Forbidden prefixes of w_1

Ingredients of the algorithm

Suppose we want to write down a lex-representative w.

$$\begin{aligned} \sigma_1 &\not\preccurlyeq w_1 \\ \sigma_2 &\not\preccurlyeq w_1 \\ \sigma_3\sigma_4 &\not\preccurlyeq w_1 \end{aligned}$$

The first letter can be anyone. Say σ_4.

$$w = \sigma_4 w_1$$

The second letter cannot be σ_1, σ_2. Suppose it is σ_3.

$$w = \sigma_4\sigma_3 w_2$$

$$\begin{aligned} \sigma_3 &\preccurlyeq w_1 \\ \sigma_1 &\not\preccurlyeq w_1 \\ \sigma_2 &\not\preccurlyeq w_1 \\ \sigma_3\sigma_4 &\not\preccurlyeq w_1 \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} \sigma_3^{-1}(\sigma_1 \vee \sigma_3) &\not\preccurlyeq w_2 \\ \sigma_3^{-1}(\sigma_2 \vee \sigma_3) &\not\preccurlyeq w_2 \\ \sigma_3^{-1}(\sigma_3\sigma_4 \vee \sigma_3) &\not\preccurlyeq w_2 \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} \sigma_1 &\not\preccurlyeq w_2 \\ \sigma_2\sigma_3 &\not\preccurlyeq w_2 \\ \sigma_4 &\not\preccurlyeq w_2 \end{aligned}$$

Forbidden prefixes of w_2

The third letter cannot be σ_1, σ_4.

Ingredients of the algorithm

In general: w = Lex-representative

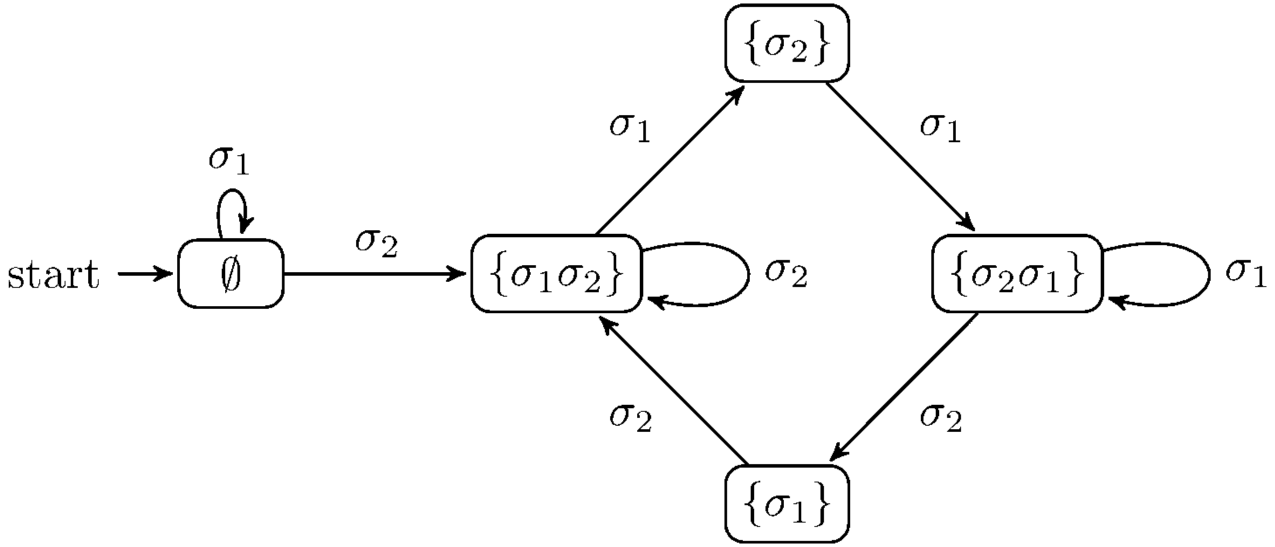
$F(w)$ = { Forbidden prefixes of w' such that ww' is a lex-representative}

$$\boxed{F(w\sigma_j) = F(\sigma_j) \cup \sigma_j^{-1}(F(w) \vee \sigma_j)}$$

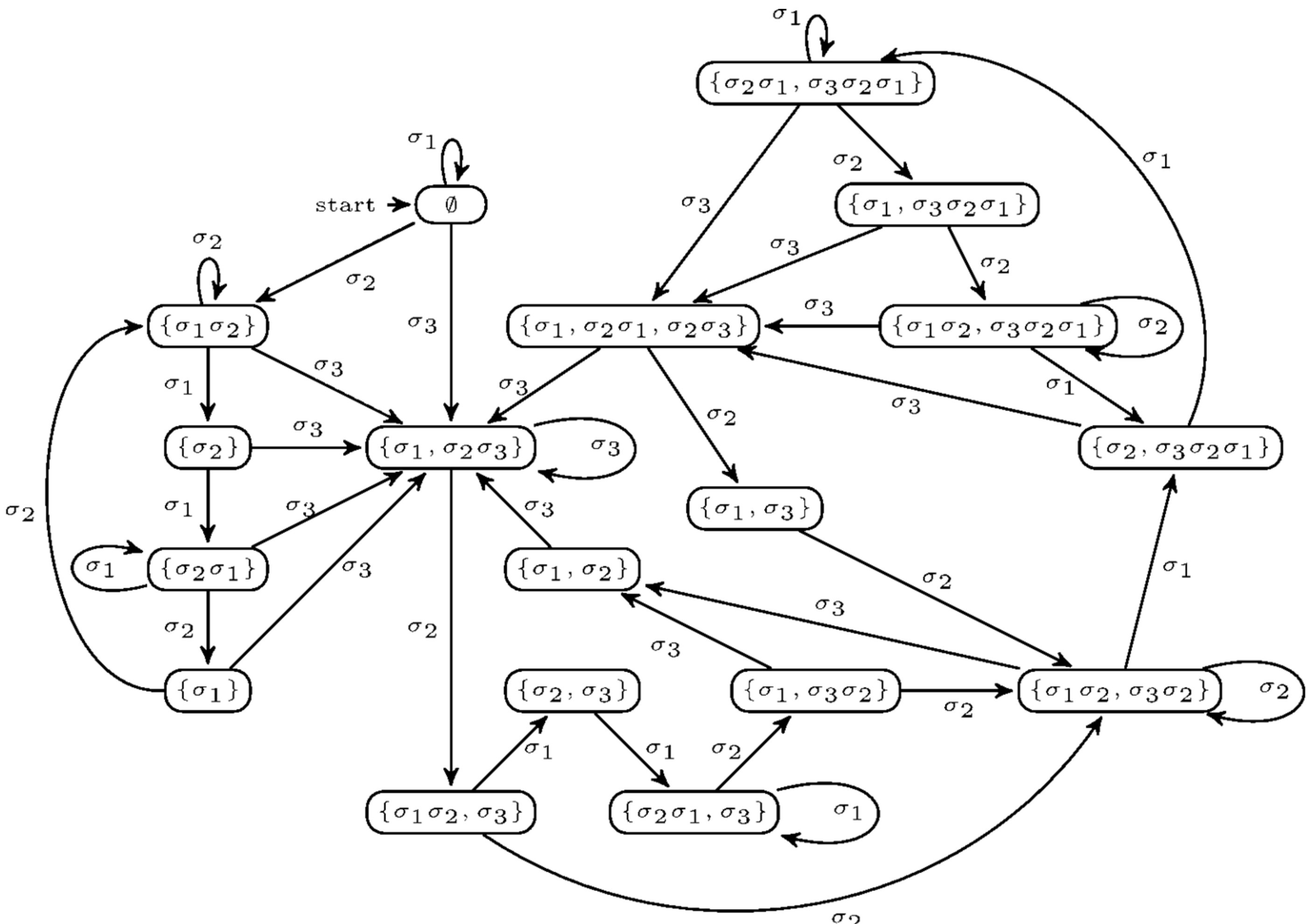
- $F(w\sigma_j)$ only depends on $F(w)$ and j — Can construct an automaton
- Forbidden prefixes are always **simple elements** — Finite number of states!

This works for every Garside monoid

Automaton accepting Lex-representantives of B_3^+



Automaton accepting Lex-representatives of $B_4^{\,+}$



How big is the automaton?

In the braid monoid B_n^+, sets of forbidden prefixes have at most n elements

Example:

Forbidden prefixes for $w = \sigma_8\sigma_7\sigma_6\sigma_6\sigma_5\sigma_4\sigma_4\sigma_3\sigma_3\sigma_2\sigma_2\sigma_1\sigma_4\sigma_3\sigma_3$:

$$F(w) = \{\sigma_1, \quad \sigma_2\sigma_3, \quad \sigma_4\sigma_3, \quad \sigma_5\sigma_4\sigma_3\sigma_2\sigma_1, \quad \sigma_6, \quad \sigma_7\sigma_6\sigma_5\sigma_4\sigma_3\sigma_2\sigma_1, \quad \sigma_8\}.$$

If the automaton is small, it can be efficiently used to generate random braids!

Theorem (Gebhardt-GM, 2011) The described automata are the smallest possible ones, and their size is exponential in n.

Using them would not be efficient

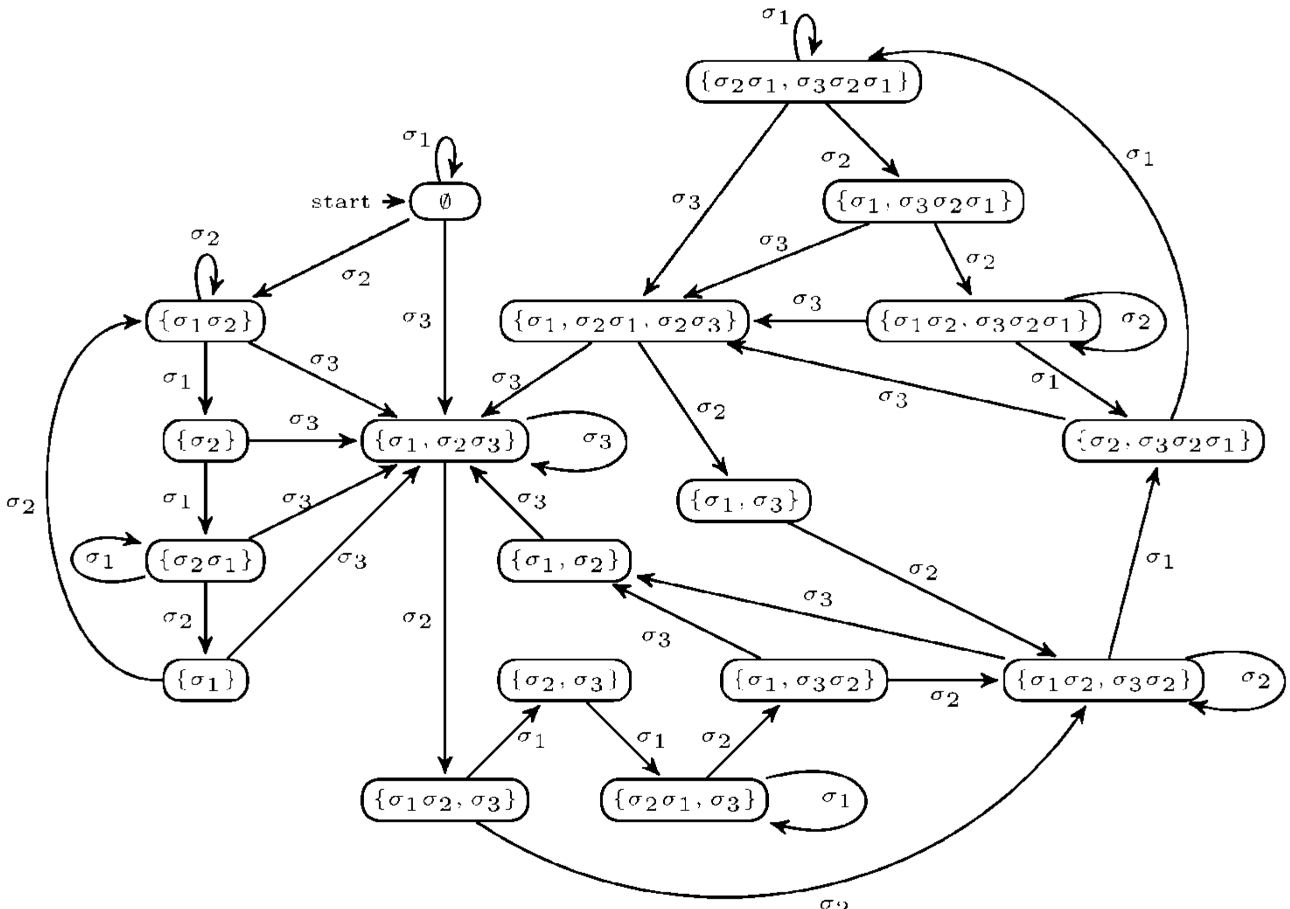
Smallest possible automaton

Why is this automaton the smallest possible one?

Because each state encodes all future accepted paths.

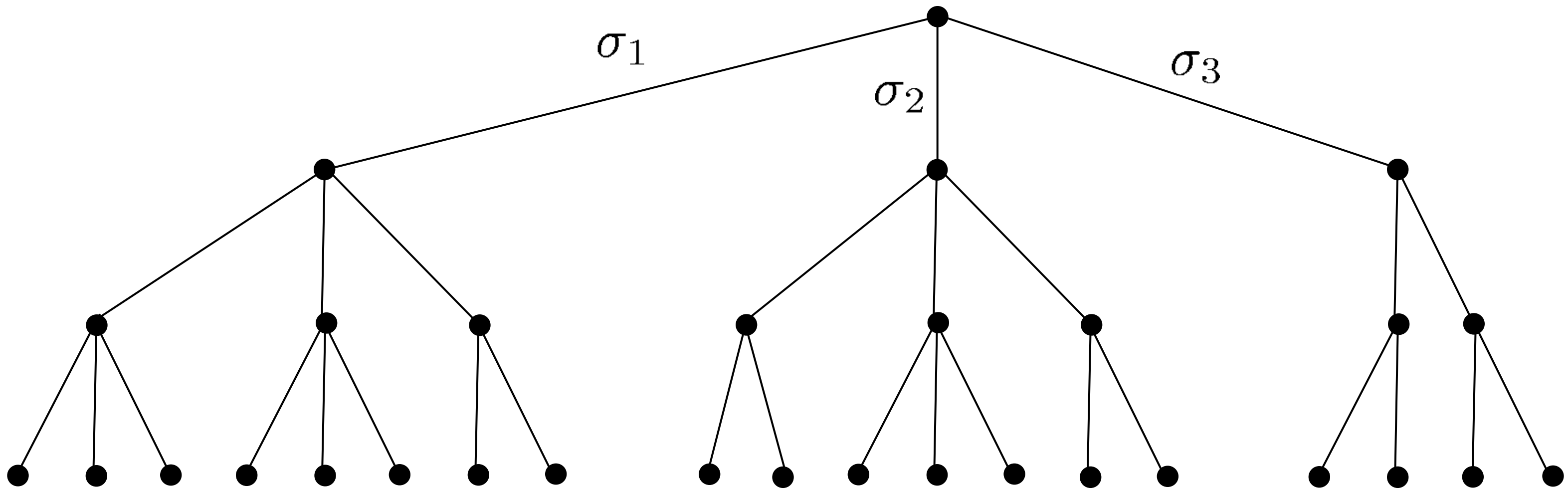
Two words with same state in some automaton…

also have the same state here.



Lex-representatives form a tree.

Example, in B_4^+ :

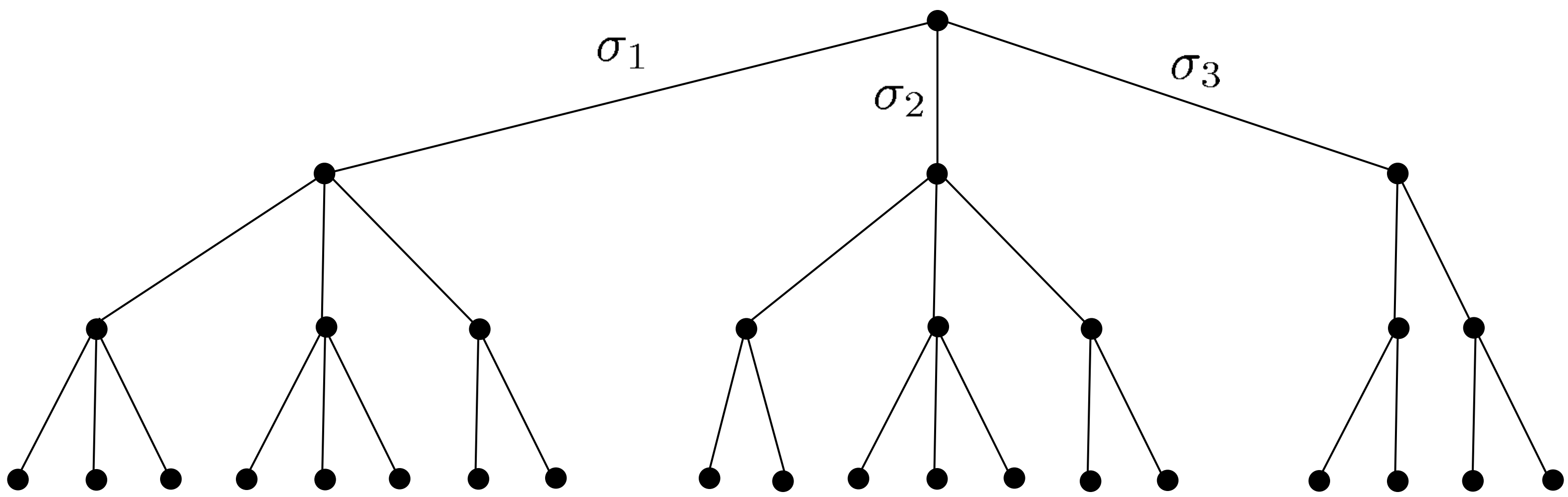


19 braids of length 3 in B_4^+.

Generating random braids

Lex-representatives form a tree.

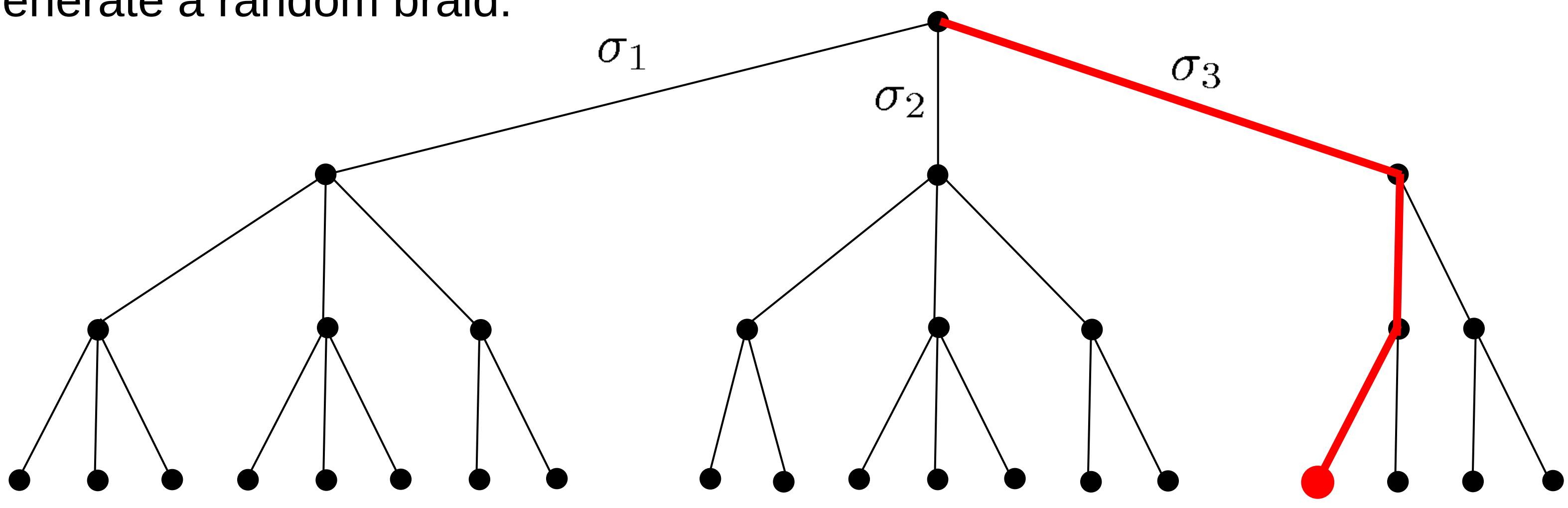
Example, in B_4^+ :



19 braids of length 3 in B_4^+.

Generating random braids

To generate a random braid:



1) Compute the number of leaves of the tree: 19

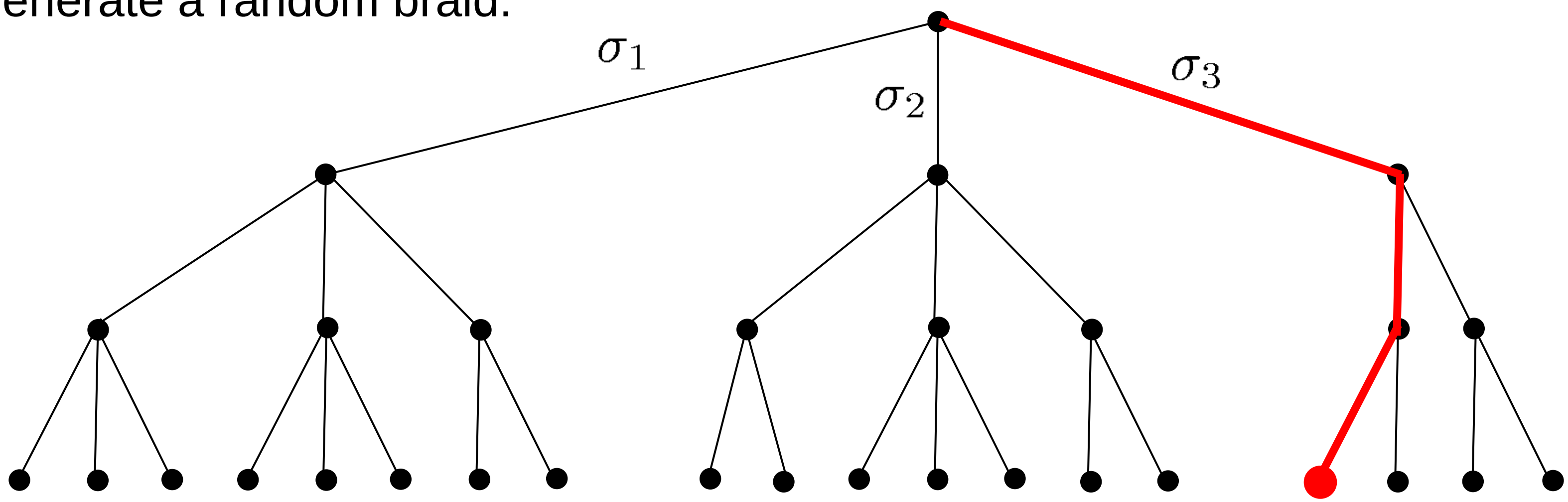
2) Choose a random number between 1 and 19: 16

3) Find the braid corresponding to the 16th leaf: $\sigma_3\sigma_2\sigma_1$

In polynomial time!

Warning: the tree is exponentially big!

To generate a random braid:



1) Compute the number of leaves of the tree: 19

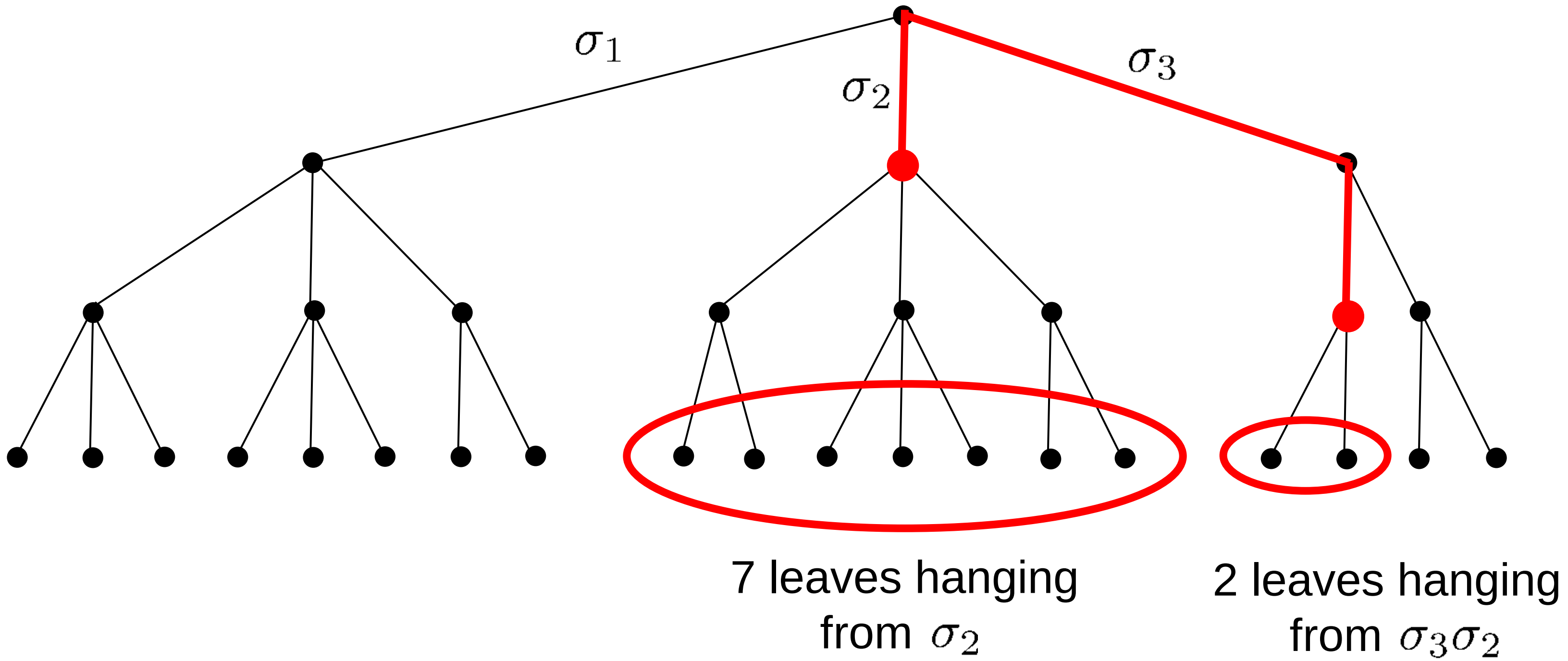
2) Choose a random number between 1 and 19: 16

3) Find the braid corresponding to the 16th leaf: $\sigma_3\sigma_2\sigma_1$

Next

Finding the rth braid of length k

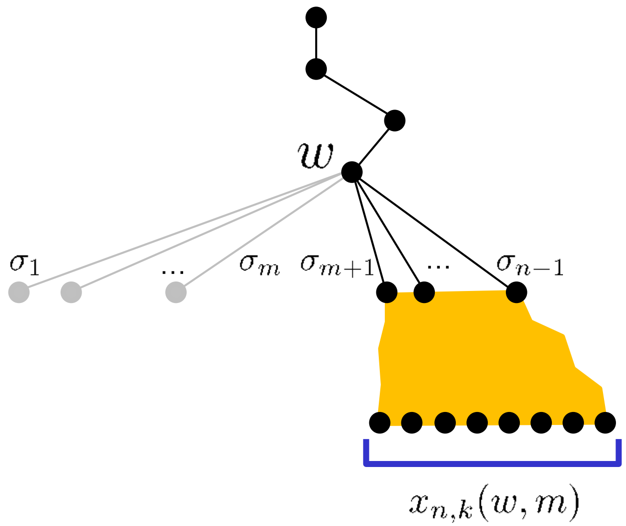
Question: How many lex-representatives start with a given prefix?



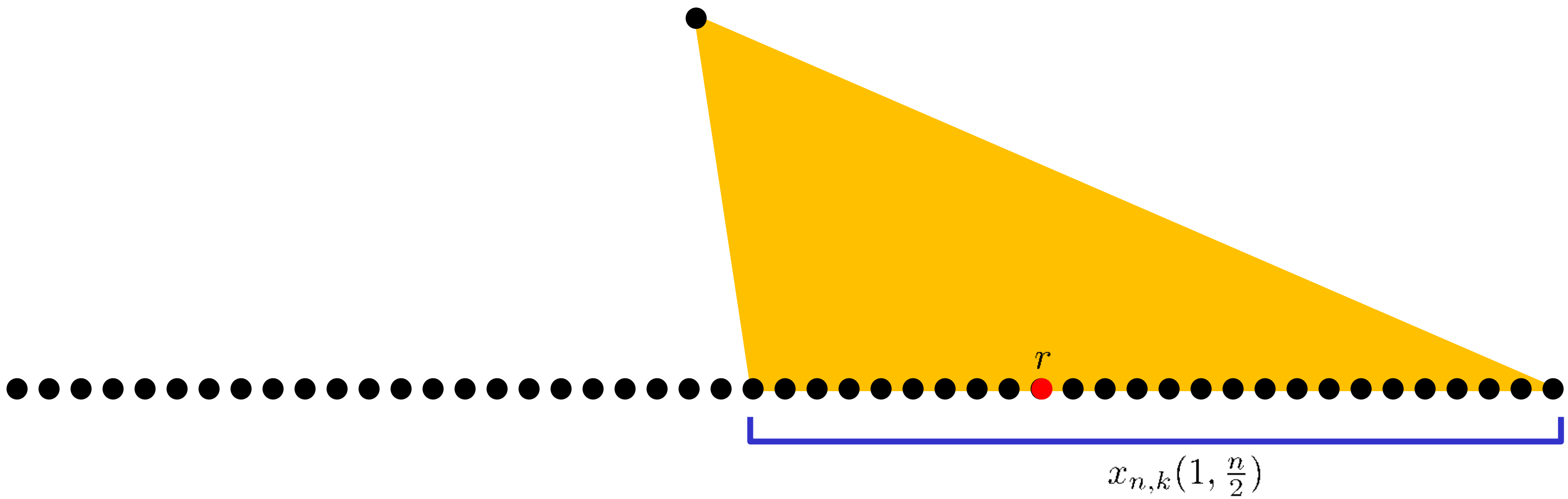
Finding the rth braid of length k

w = lex-representative = vertex of the tree $\qquad m \in \{0, \ldots, n-1\}$

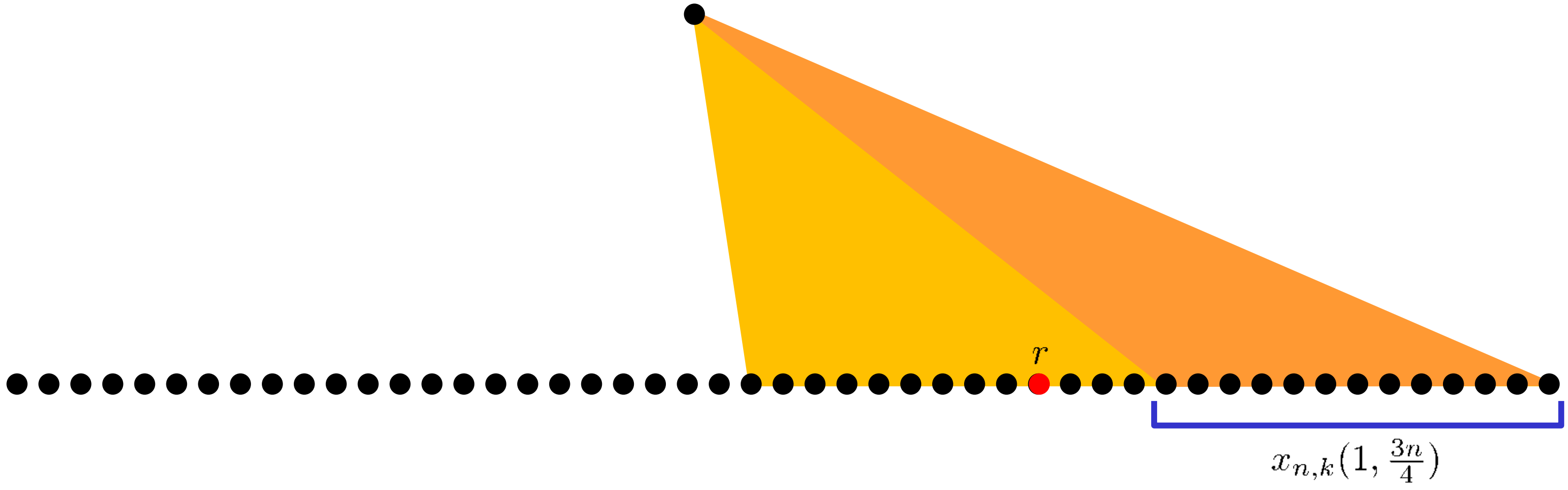
$x_{n,k}(w,m)$ = leaves hanging from w but not from $w\sigma_1, \ldots, w\sigma_m$



To find the rth braid: Binary search



To find the rth braid: Binary search



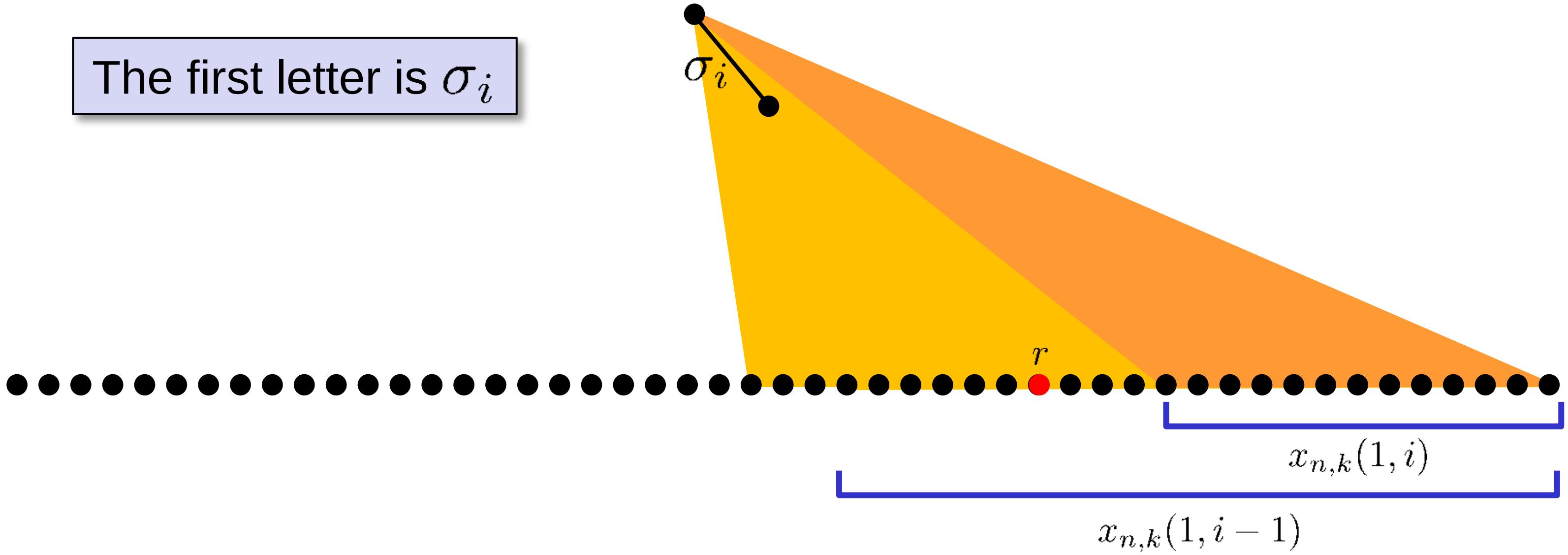
Finding the rth braid of length k

To find the rth braid:

Binary search

Now do binary search with $x_{n,k}(\sigma_i, m)$

The first letter is σ_i



Finding the rth braid of length k

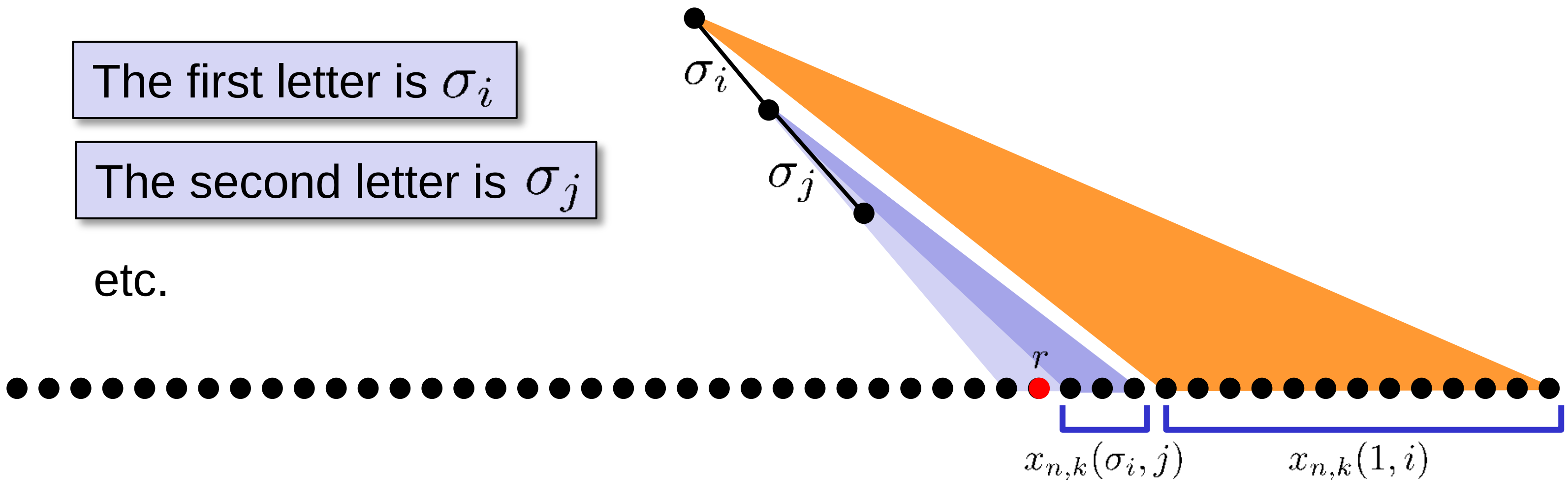
To find the rth braid: Binary search

Now do binary search with $x_{n,k}(\sigma_i, m)$

The first letter is σ_i

The second letter is σ_j

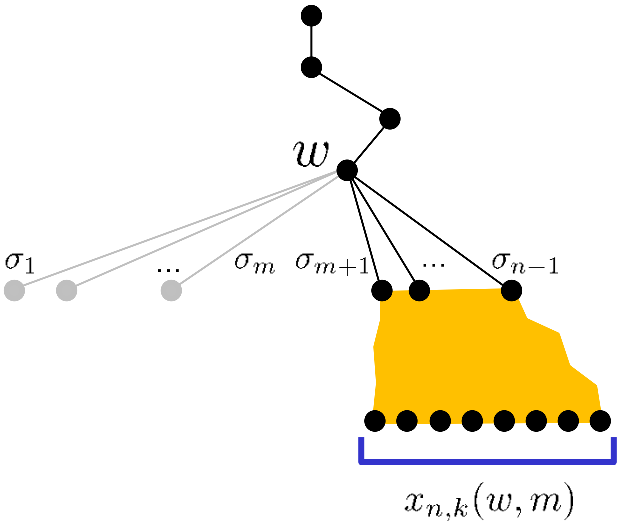
etc.



We find each letter in $\log n$ steps

Computing hanging leaves

How to compute the number $x_{n,k}(w,m)$?



It is the number of braids of length $k-|w|$ ← We know it

minus the number of braids having some prefix from:

$$\{\sigma_1,\ldots,\sigma_m\} \cup F(w)$$

← Computed with the inclusion-exculsion principle

Conclusion

Thanks to the lattice structure of Garside monoids we have obtained:

- An efficient algorithm to generate random elemens. (type A)
- A new formula for the growth function. (type A,B,D)
- The smallest possible finite state automata recognizing lex-representatives. (all Garside monoids)