Permutree Sorting, Pattern Avoidance, and Automata

Daniel Tamayo Jiménez

Université Paris-Saclay (LRI)

Joint work with Vincent Pilaud and Viviane Pons

Doctorants en Géométrie Combinatoire
January 21, 2021

Outline

- Some motivation
- The path from congruences to automata
- Automata and algorithms
- c-sorting

Why automata?

The congruences that we consider come from a more general case studied by Reading [Rea07] through several different perspectives.

- Lattice congruences
- Root systems
- Reduced words
- Pattern avoidance (Types A, B, and D).

Why automata?

The congruences that we consider come from a more general case studied by Reading [Rea07] through several different perspectives.

- Lattice congruences
- Root systems
- Reduced words (Automata!)
- Pattern avoidance (Types A, B, and D).

Objective

Characterize minimal elements of certain weak order lattice congruences (permutrees) via reduced expressions.

There are already other characterizations [PP18], [CPP19]:

- Pattern avoidance.
- Minimality of linear extensions of permutrees.
- Inversion sets.

What is sorting?

It is an algorithm that rearranges permutations. If the algorithm outputs the identity permutation, we say that the input permutation is *sortable for this algorithm*.

Example: Stack sorting (Knuth 60's)

The map $S : \mathfrak{S}_n \to \mathfrak{S}_n$ defined as $S(\tau n \rho) = S(\tau)S(\rho)n$

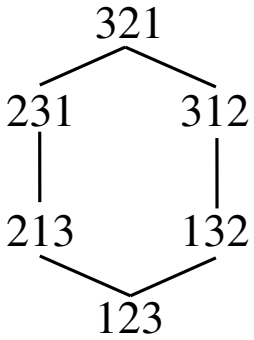
What is sorting?

It is an algorithm that rearranges permutations. If the algorithm outputs the identity permutation, we say that the input permutation is *sortable for this algorithm*.

Example: Stack sorting (Knuth 60's)

The map $S : \mathfrak{S}_n \to \mathfrak{S}_n$ defined as $S(\tau n \rho) = S(\tau)S(\rho)n$

$$
\begin{aligned}
S(321) &= S(21)3 &&= 123\\
S(231) &= S(2)S(1)3 &&= 213\\
S(312) &= S(12)3 &&= 123\\
S(213) &= S(21)3 &&= 123\\
S(132) &= S(1)S(2)3 &&= 123\\
S(123) &= S(12)3 &&= 123
\end{aligned}
$$



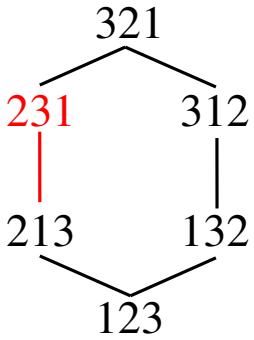
What is sorting?

It is an algorithm that rearranges permutations. If the algorithm outputs the identity permutation, we say that the input permutation is *sortable for this algorithm*.

Example: Stack sorting (Knuth 60's)

The map $S : \mathfrak{S}_n \to \mathfrak{S}_n$ defined as $S(\tau n \rho) = S(\tau)S(\rho)n$

$$
\begin{aligned}
S(321) &= S(21)3 &&= 123\\
S(231) &= S(2)S(1)3 &&= 213\\
S(312) &= S(12)3 &&= 123\\
S(213) &= S(21)3 &&= 123\\
S(132) &= S(1)S(2)3 &&= 123\\
S(123) &= S(12)3 &&= 123
\end{aligned}
$$



Permutations

We are interested in working with the following presentation of the symmetric group

$$\mathfrak{S}_n = \langle \ \{s_1, \ldots, s_{n-1}\} \ : \ (s_i s_{i+1})^3 = (s_i s_j)^2 = e \ \rangle$$

where $s_i = (i \ i+1)$ are the simple transpositions.

Weak order

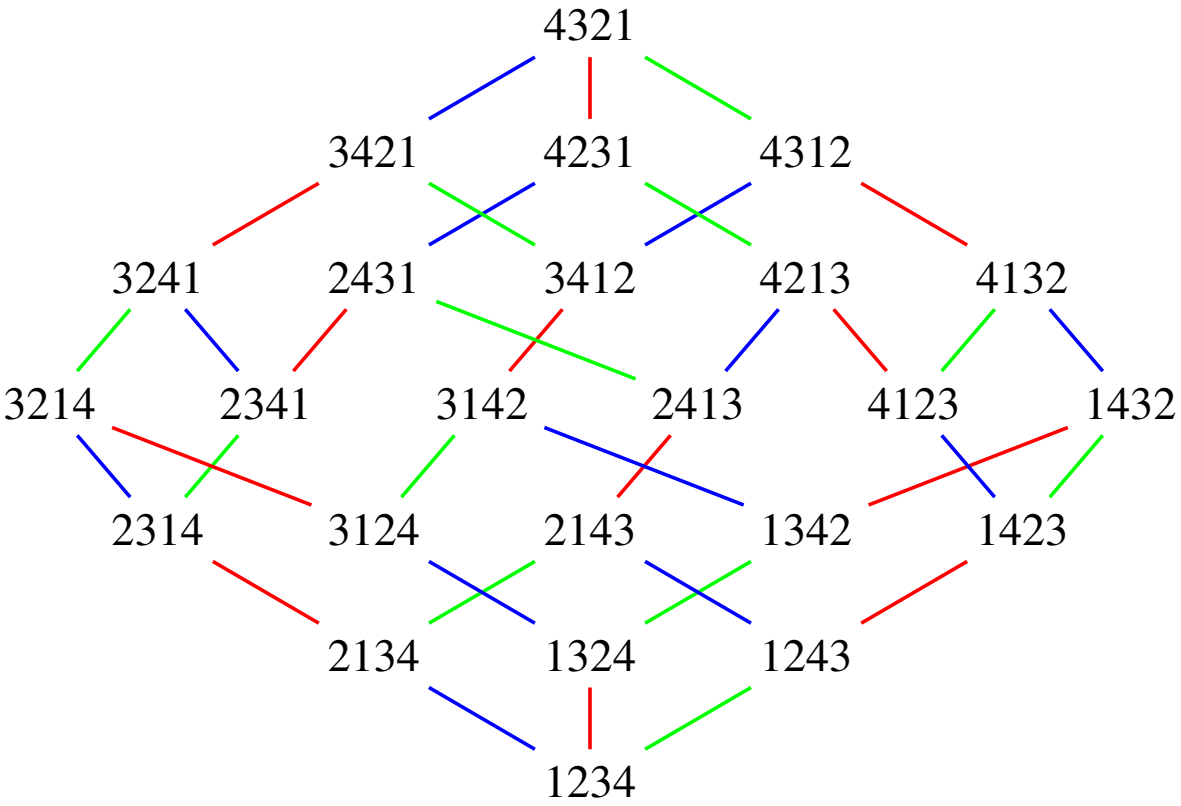


Figure 1: The (right) weak order of $\mathfrak{S}_4$ generated by s_1, s_2, s_3.

Weak order congruences

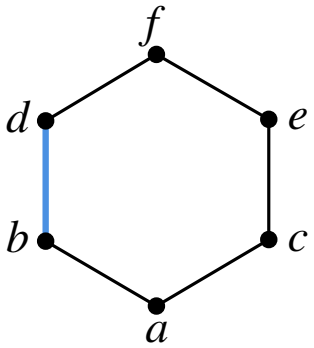
We want to take congruences that preserve meets and joins. That is,

$$x \equiv x' \text{ and } y \equiv y' \implies x \vee y \equiv x' \vee y' \text{ and } x \wedge y \equiv x' \wedge y'$$

Weak order congruences

We want to take congruences that preserve meets and joins. That is,

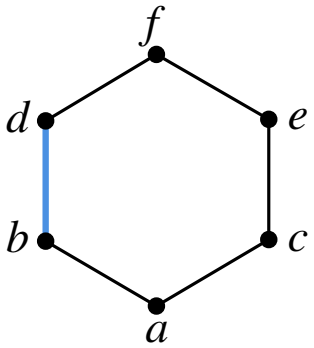
$$x \equiv x' \text{ and } y \equiv y' \implies x \vee y \equiv x' \vee y' \text{ and } x \wedge y \equiv x' \wedge y'$$



Weak order congruences

We want to take congruences that preserve meets and joins. That is,

$$x \equiv x' \text{ and } y \equiv y' \implies x \vee y \equiv x' \vee y' \text{ and } x \wedge y \equiv x' \wedge y'$$

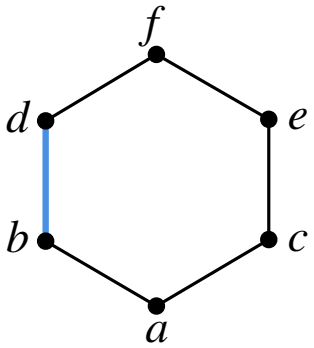
- 

Nothing else is affected.

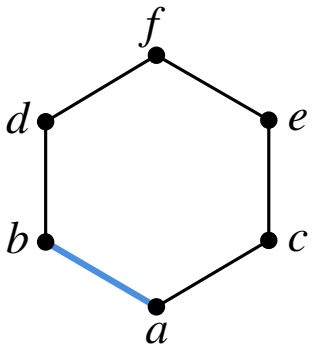
Weak order congruences

We want to take congruences that preserve meets and joins. That is,

$$x \equiv x' \text{ and } y \equiv y' \implies x \vee y \equiv x' \vee y' \text{ and } x \wedge y \equiv x' \wedge y'$$

- 

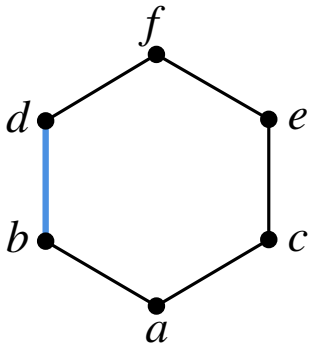
Nothing else is affected.

- 

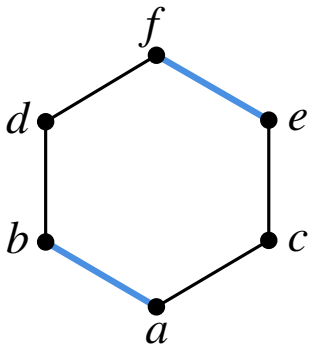
Weak order congruences

We want to take congruences that preserve meets and joins. That is,

$$x \equiv x' \text{ and } y \equiv y' \implies x \vee y \equiv x' \vee y' \text{ and } x \wedge y \equiv x' \wedge y'$$

- 

  Nothing else is affected.

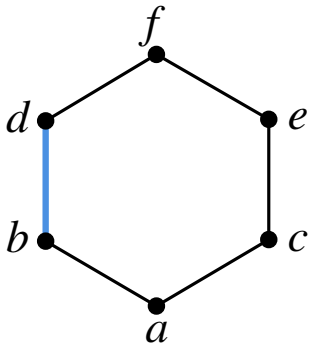
- 

  $e \vee b \equiv e \vee a \quad \Rightarrow \quad f \equiv e.$

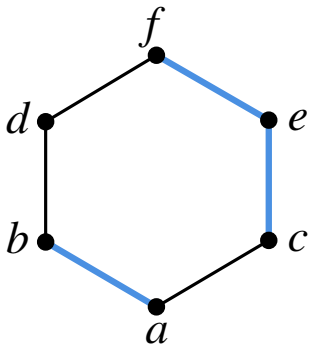
Weak order congruences

We want to take congruences that preserve meets and joins. That is,

$$x \equiv x' \text{ and } y \equiv y' \implies x \vee y \equiv x' \vee y' \text{ and } x \wedge y \equiv x' \wedge y'$$

- 

  Nothing else is affected.

- 

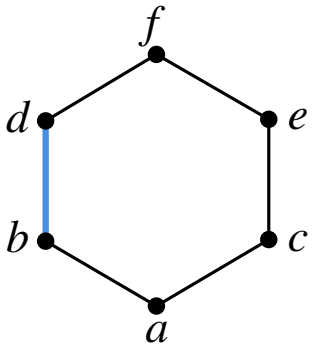
  $e \vee b \equiv e \vee a \quad \Rightarrow \quad f \equiv e.$

  $c \vee b \equiv c \vee a \quad \Rightarrow \quad f \equiv c.$

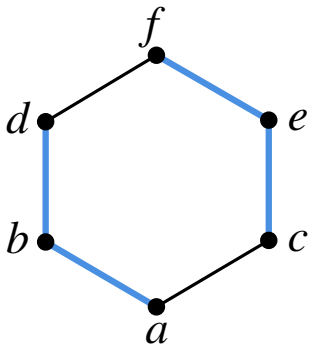
Weak order congruences

We want to take congruences that preserve meets and joins. That is,

$$x \equiv x' \text{ and } y \equiv y' \implies x \vee y \equiv x' \vee y' \text{ and } x \wedge y \equiv x' \wedge y'$$

- 

  Nothing else is affected.

- 

  $e \vee b \equiv e \vee a \quad \Rightarrow \quad f \equiv e.$

  $c \vee b \equiv c \vee a \quad \Rightarrow \quad f \equiv c.$

  $d \wedge f \equiv d \wedge e \quad \Rightarrow \quad d \equiv a.$

Weak order congruences

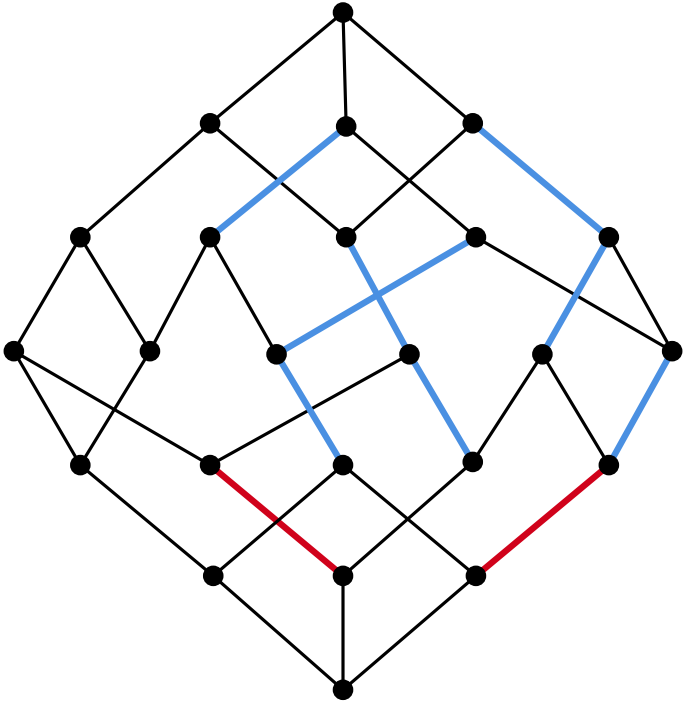


Figure 2: The finest lattice congruence that contracts 2 edges.

Why these congruences?

The equivalence classes are an equivalent definition of permutrees.

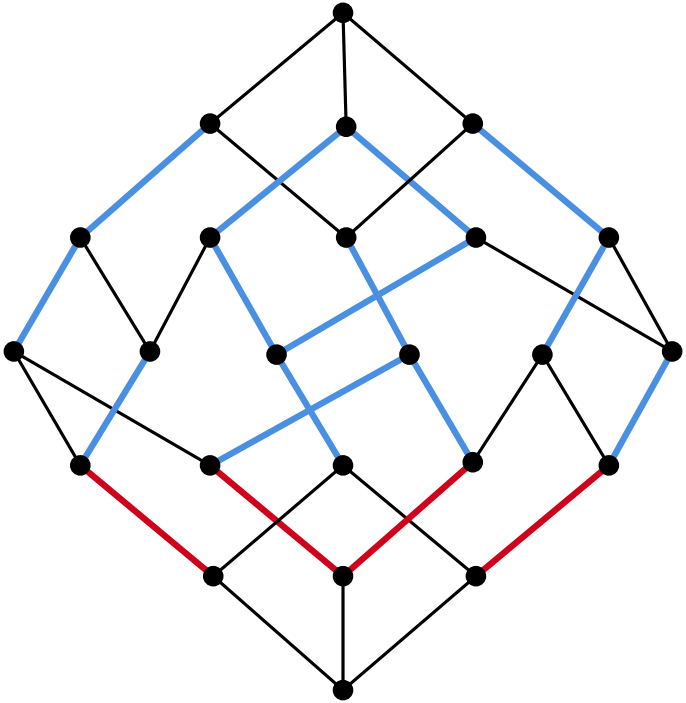


Figure 3: The finest lattice congruence that contracts all 4 edges.

Coxeter Graphs

We can identify our congruences through orientations of the Coxeter graph of $\mathfrak{S}_n$.

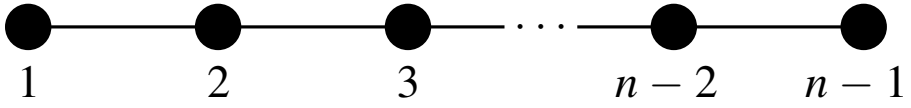


Figure 4: Coxeter graph of $\mathfrak{S}_n$.

Coxeter Graphs

We can identify our congruences through orientations of the Coxeter graph of $\mathfrak{S}_n$.

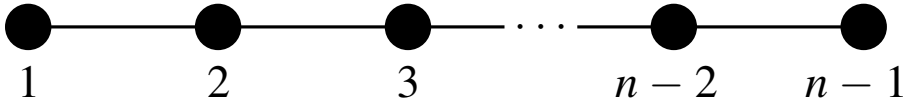
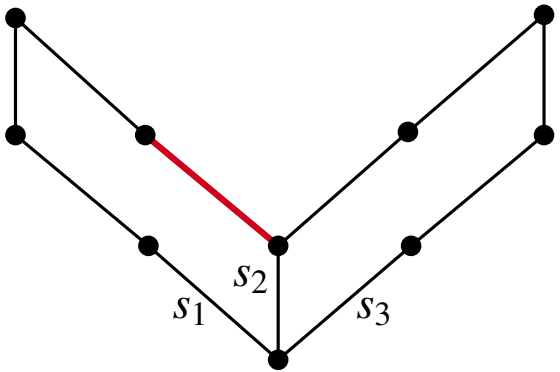


Figure 4: Coxeter graph of $\mathfrak{S}_n$.



Coxeter Graphs

We can identify our congruences through orientations of the Coxeter graph of $\mathfrak{S}_n$.

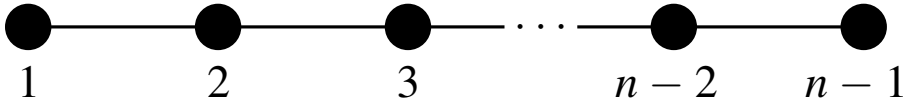
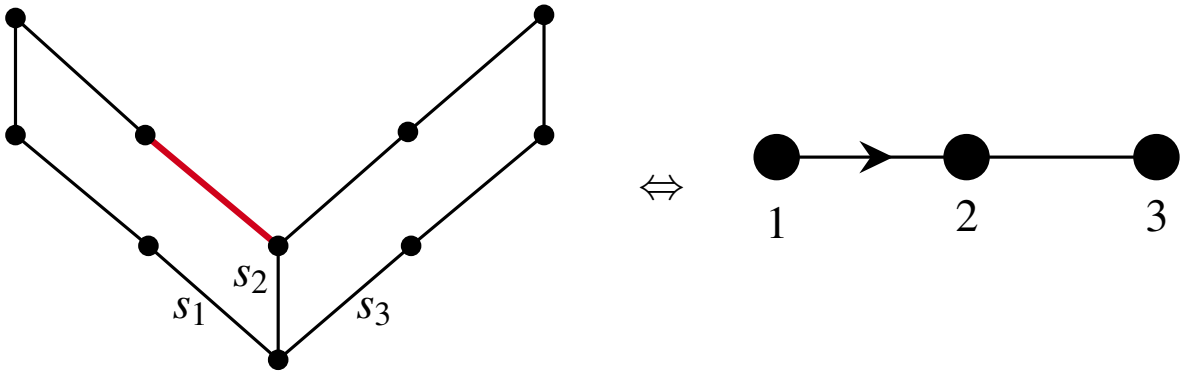


Figure 4: Coxeter graph of $\mathfrak{S}_n$.



Coxeter Graphs

We can identify our congruences through orientations of the Coxeter graph of $\mathfrak{S}_n$.

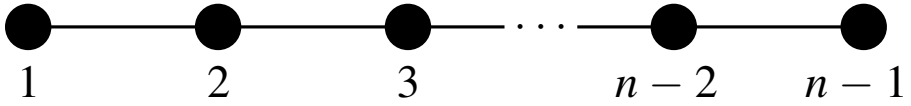
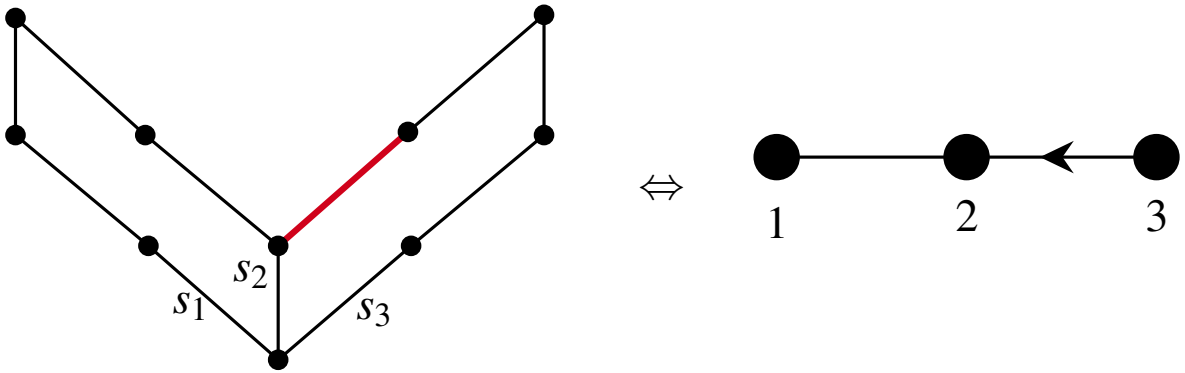


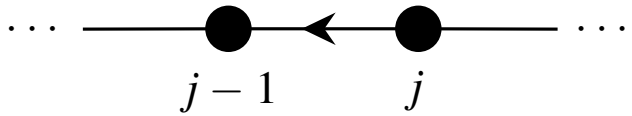
Figure 5: Coxeter graph of $\mathfrak{S}_n$.



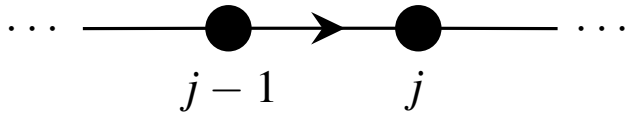
U and D

Furthermore, we can encode the orientations through sets U and D.

- $j \in U$ if



- $j \in D$ if



Note that depending of the orientation U and D may be disjoint or overlapping and may form a partition of $\{2, \ldots, n-1\}$.

Single orientation automata

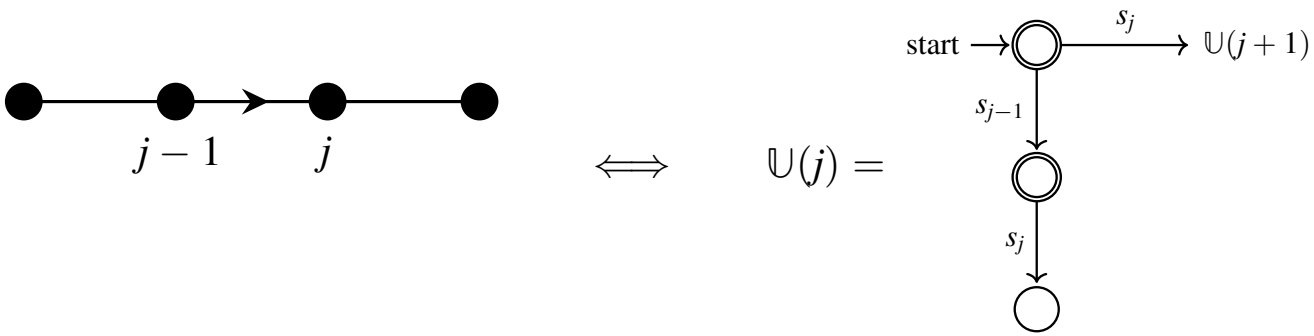


Figure 6: A single orientation and its automaton.

Single orientation automata

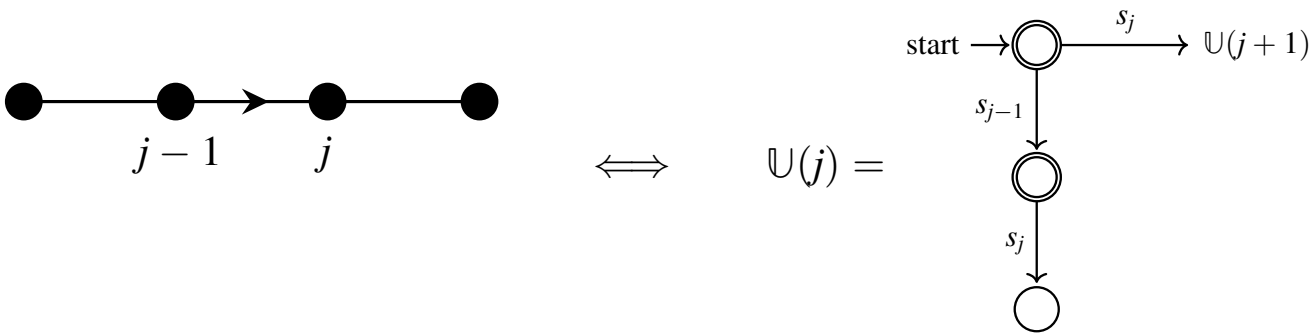


Figure 6: A single orientation and its automaton.

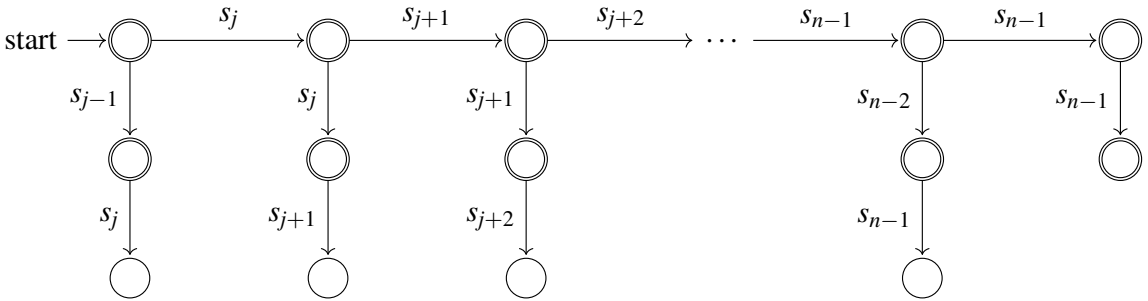


Figure 7: The complete automaton $\mathbb{U}(j)$.

Properties of the automata

Theorem (Pilaud, Pons, T. 2020)

Fix $j \in \{2, \dots, n-1\}$. The following conditions are equivalent for $\pi \in \mathfrak{S}_n$:

- π has a reduced expression accepted by the automaton $\mathbb{U}(j)$,
- π avoids jki with $i < j < k$. (j is fixed!)

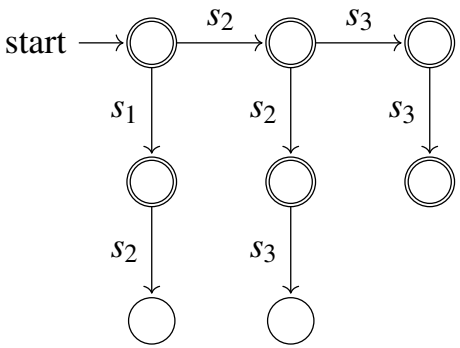
Properties of the automata

Theorem (Pilaud, Pons, T. 2020)

Fix $j \in \{2, \dots, n-1\}$. The following conditions are equivalent for $\pi \in \mathfrak{S}_n$:

- π has a reduced expression accepted by the automaton $\mathbb{U}(j)$,
- π avoids jki with $i < j < k$. (j is fixed!)

Example:



4213 avoids $2ki$ and has the reduced expression $s_3 \cdot s_2 \cdot s_1 \cdot s_2$ which is accepted by $\mathbb{U}(2)$.

Some other properties

The accepted reduced words have a nice structure!

- They are closed by prefix.

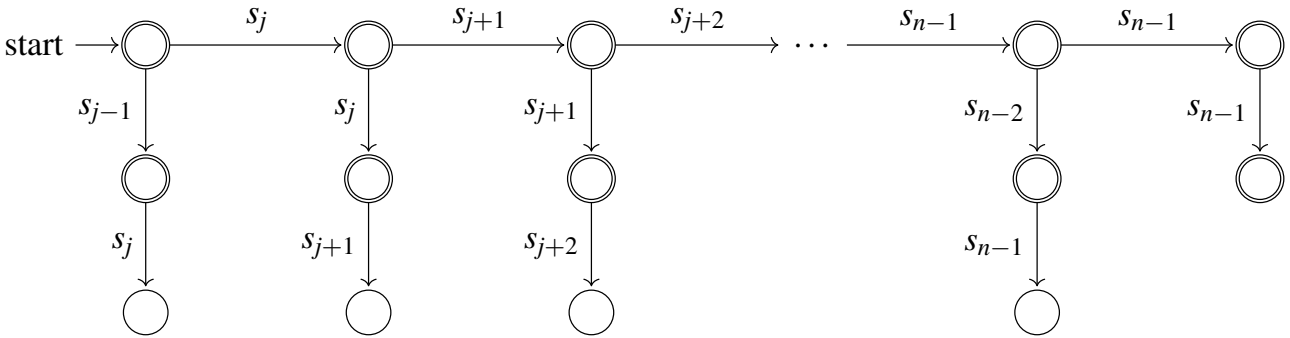


Figure 8: The complete automaton $\mathbb{U}(j)$.

Some other properties

The accepted reduced words have a nice structure!

- They are closed by prefix.
- All are accepted in the same state of our automata.

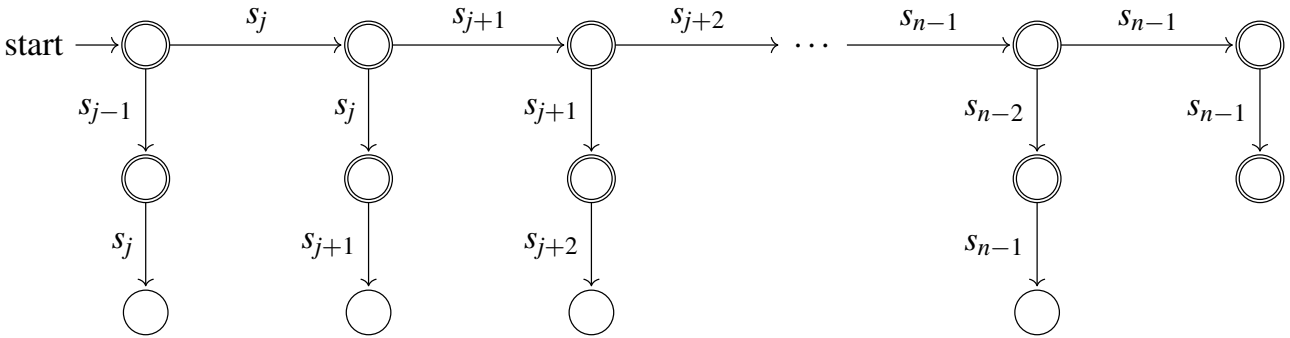


Figure 8: The complete automaton $\mathbb{U}(j)$.

Some other properties

The accepted reduced words have a nice structure!

- They are closed by prefix.
- All are accepted in the same state of our automata.
- If a permutation is accepted, then there is a reduced expression starting with its descents that prioritizes healthy states.

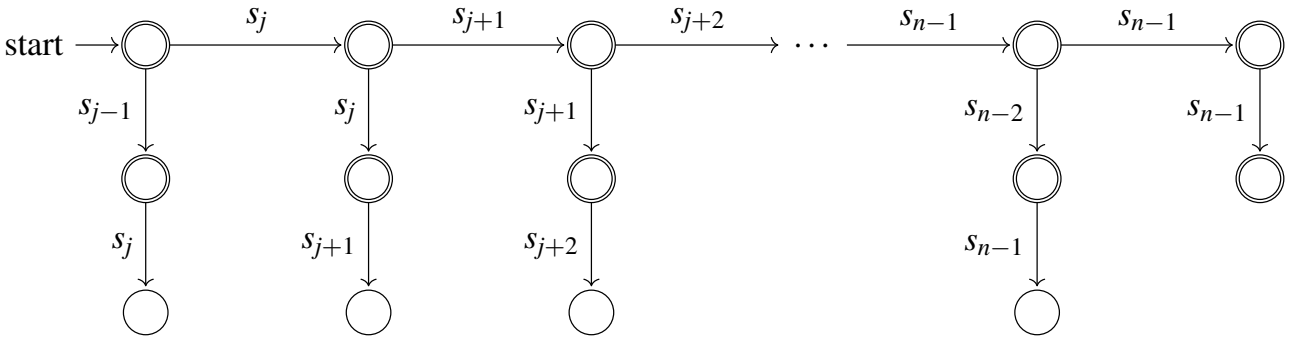


Figure 8: The complete automaton $\mathbb{U}(j)$.

Some other properties

The accepted reduced words have a nice structure!

- They are closed by prefix.
- All are accepted in the same state of our automata.
- If a permutation is accepted, then there is a reduced expression starting with its descents that prioritizes healthy states.

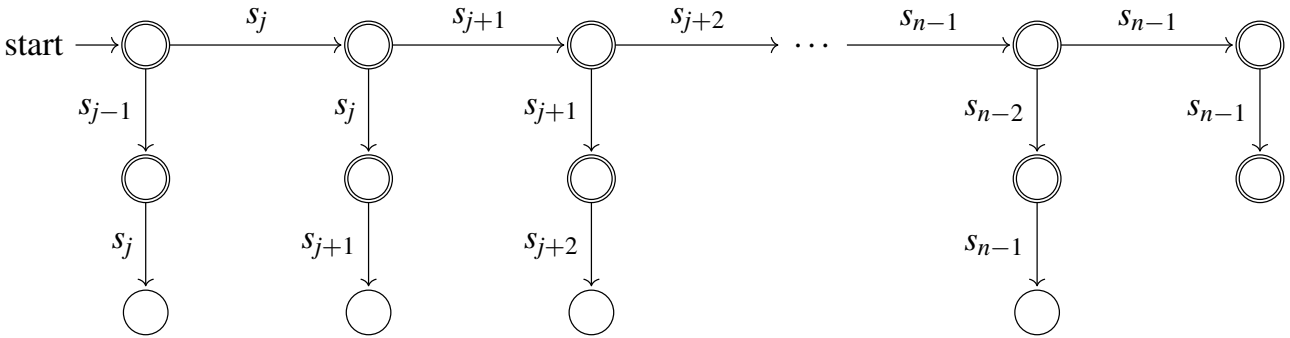


Figure 9: The complete automaton $\mathbb{U}(j)$.

Proposition (Pilaud, Pons, T. 2020)

For $\pi \in \mathfrak{S}_n$, let $\mathrm{inv}^j(\pi) = \big\{(j,i) \;\big|\; i < j \text{ and } \pi^{-1}(i) > \pi^{-1}(j)\big\}$
and $\mathrm{inv}_j(\pi) = \big\{(k,j) \;\big|\; j < k \text{ and } \pi^{-1}(j) > \pi^{-1}(k)\big\}$.

1. if $\mathrm{inv}^j(\pi) = \emptyset$, all red. exp. for π end at the same healthy state of $\mathbb{U}(j)$,
2. if $\mathrm{inv}_j(\pi) = \emptyset$, all red. exp. for π end at the same state of $\mathbb{U}(j)$,
3. if $\mathrm{inv}^j(\pi) \neq \emptyset \neq \mathrm{inv}_j(\pi)$, all accepted reduced expressions for π end at the same ill state of $\mathbb{U}(j)$.

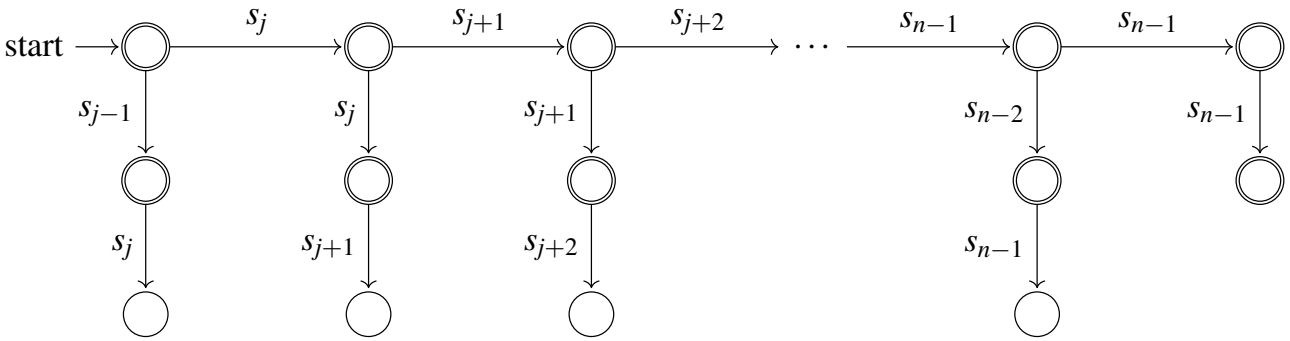
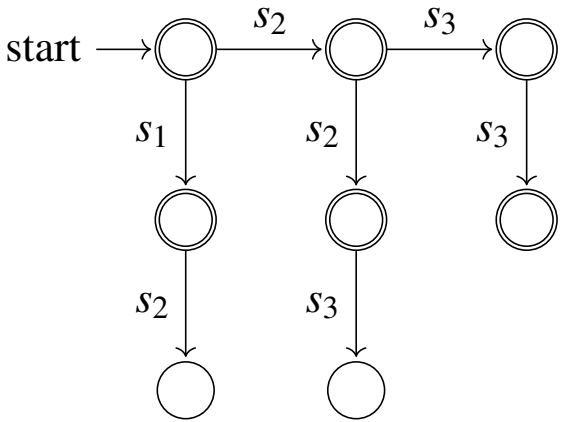


Figure 10: The complete automaton $\mathbb{U}(j)$.

Accepted reduced words



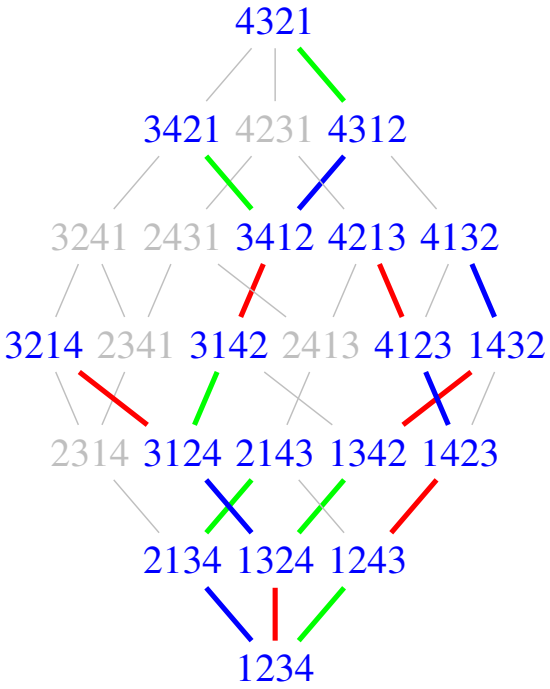


Figure 11: $\mathbb{U}(2)$ and its accepted reduced words.

Some other properties

The accepted reduced words have a nice structure!

- They are closed by prefix.
- All are accepted in the same state of our automata.
- If a permutation is accepted, then there is a reduced expression starting with its descents that prioritizes healthy states.

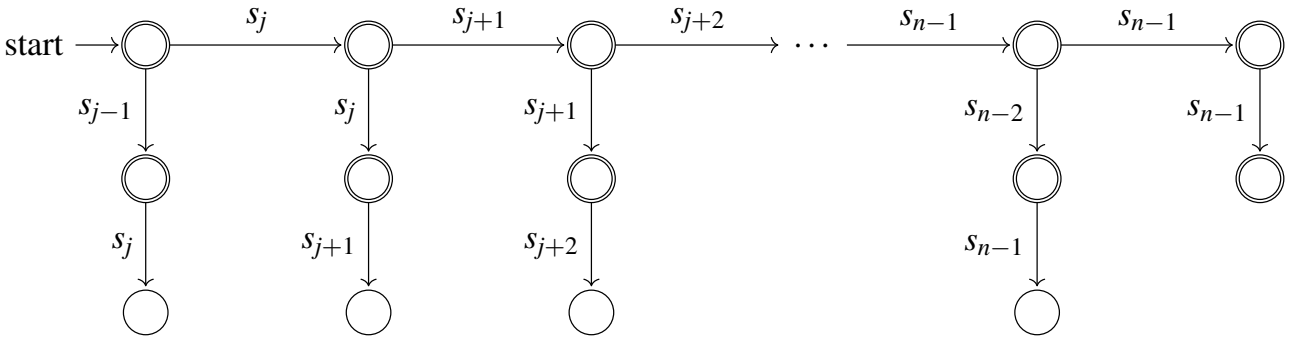


Figure 12: The complete automaton $\mathbb{U}(j)$.

Algorithm 1

Data: a permutation $\pi \in \mathfrak{S}_n$ and an integer $j \in \{2, \ldots, n-1\}$
Result: a reduced word accepted by $\mathbb{U}(j)$, candidate reduced expression for π

$w := \varepsilon$
repeat
  if $\exists\ \ell \neq j-1$ *such that* ℓ *and* $\ell+1$ *are reversed in* π **then**
    $\pi := s_\ell \cdot \pi, \quad w := w \cdot s_\ell$
    if $\ell = j$ **then** $j := j+1$

Algorithm 1

Data: a permutation $\pi \in \mathfrak{S}_n$ and an integer $j \in \{2, \dots, n-1\}$
Result: a reduced word accepted by $\mathbb{U}(j)$, candidate reduced expression for π

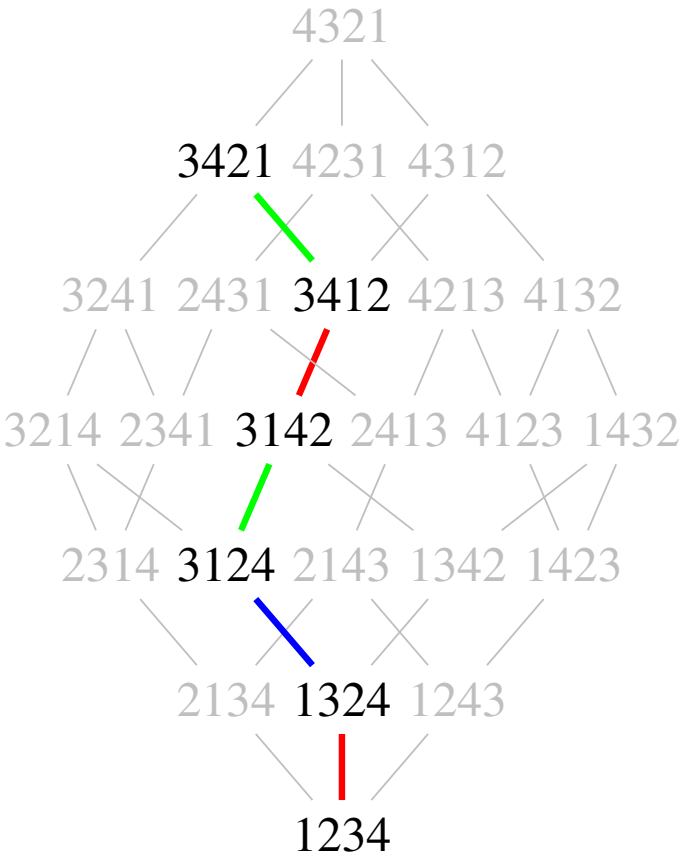
$w := \varepsilon$
repeat
  if $\exists\ \ell \neq j-1$ *such that* ℓ *and* $\ell+1$ *are reversed in* π **then**
    $\pi := s_\ell \cdot \pi, \quad w := w \cdot s_\ell$
    if $\ell = j$ **then** $j := j+1$
if $j-1$ *and* j *are reversed in* π **then**
  $\pi := s_{j-1} \cdot \pi, \quad w := w \cdot s_{j-1}$
  $w := w \cdot w' \cdot w''$ where w' sorts $\pi_{[j]}$ and w'' sorts $\pi_{[n] \setminus [j]}$
return w

Algorithm 2: $(\{j\}, \varnothing)$-permutree sorting.

Example

π	w	j	ℓ
3421	ε	2	2
2431	s_2	3	1
1432	$s_2 \cdot s_1$	3	3
1342	$s_2 \cdot s_1 \cdot s_3$	4	2
1243	$s_2 \cdot s_1 \cdot s_3 \cdot s_2$	4	3
1234	$s_2 \cdot s_1 \cdot s_3 \cdot s_2 \cdot s_3$	4	

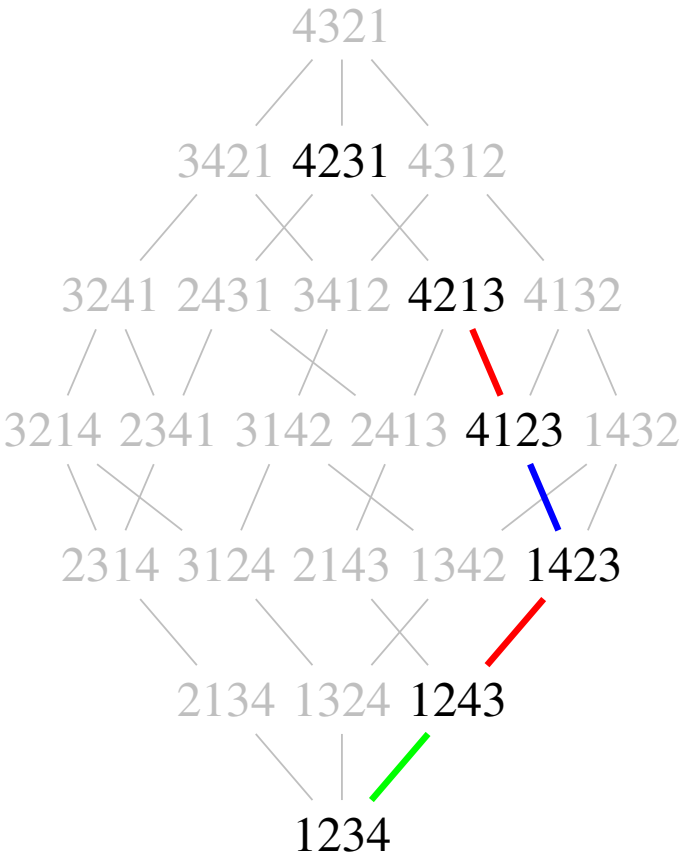
Table 1: The $(\{2\}, \varnothing)$-permutree sorting of $\pi_2 := 3421$.



Non-example

π	w	j	ℓ
4231	ε	2	3
3241	s_3	2	2
2341	$s_3 \cdot s_2$	3	1
1342	$s_3 \cdot s_2 \cdot s_1$	3	2
1243	$s_3 \cdot s_2 \cdot s_1 \cdot s_2$	3	

Table 2: The $(\{2\}, \varnothing)$-permutree sorting of $\pi_2 := 4231$.



Multiple orientation automata

Since several orientations mean multiple congruences, we need to take multiple automata. That is, their intersection.

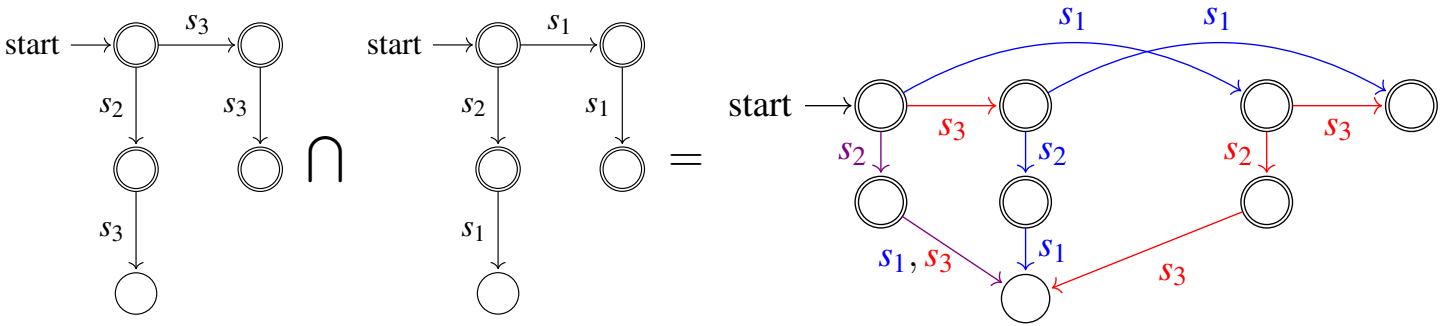


Figure 13: The automaton $\mathbb{P}(\{3\},\{2\})$.

Red (resp. blue) transitions indicate we are staying in the same type of state in $U(3)$ (resp. $D(2)$). Purple ones indicate we are changing state in both automata.

Intersection of automata

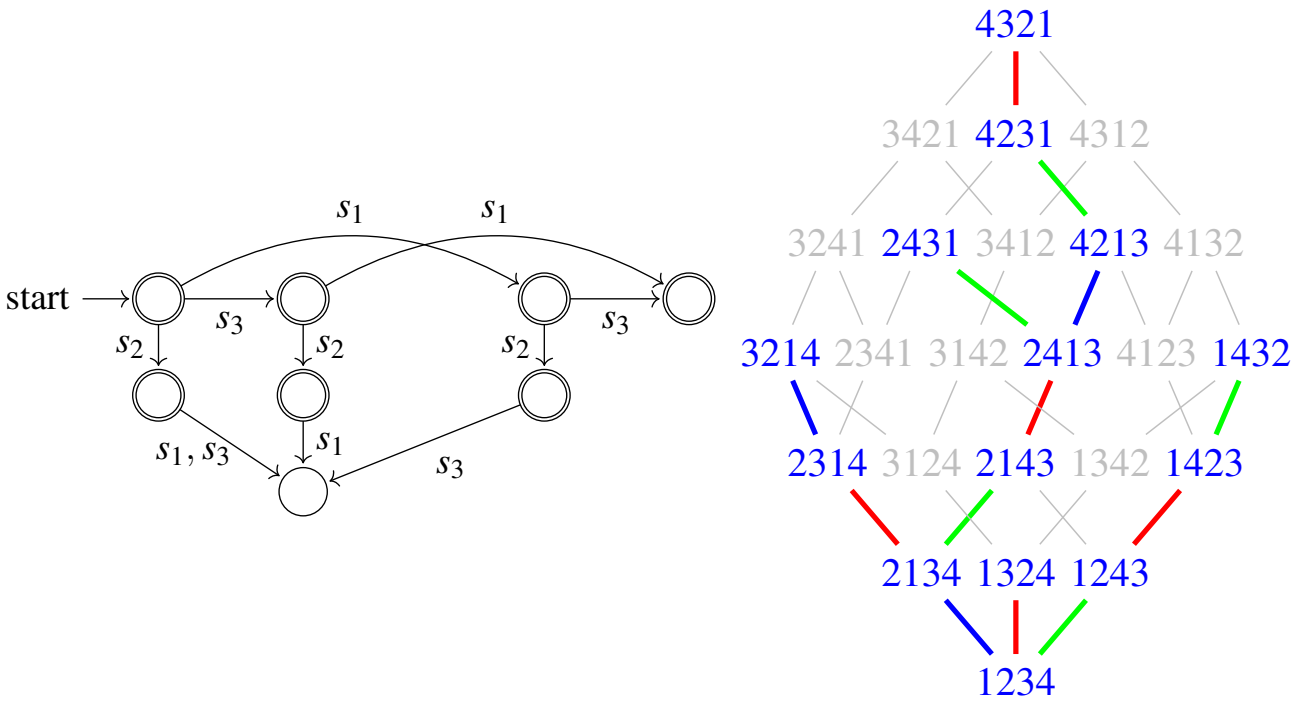


Figure 14: $\mathbb{P}(\{3\}, \{2\})$ and its accepted reduced words.

Algorithm 2

Data: a permutation $\pi \in \mathfrak{S}_n$ and two disjoint subsets U and D of $\{2, \ldots, n-1\}$

Result: a reduced word accepted by $\mathbb{P}(U, D)$, candidate reduced expression for π

if $\exists\ \ell \in [n-1]$ *such that* ℓ *and* $\ell+1$ *are reversed in* π, *and* $\ell+1 \notin U$ *and* $\ell \notin D$ **then**

  return $s_\ell \cdot \text{permutreeSort}(s_\ell \cdot \pi,\ \text{moveU}(U, \ell),\ \text{moveD}(D, \ell))$

Algorithm 2

Data: a permutation $\pi \in \mathfrak{S}_n$ and two disjoint subsets U and D of $\{2, \ldots, n-1\}$

Result: a reduced word accepted by $\mathbb{P}(U, D)$, candidate reduced expression for π

if $\exists\ \ell \in [n-1]$ *such that* ℓ *and* $\ell+1$ *are reversed in* π, *and* $\ell+1 \notin U$ *and* $\ell \notin D$ **then**

  return $s_\ell \cdot \mathrm{permutreeSort}(s_\ell \cdot \pi,\ \mathrm{moveU}(U, \ell),\ \mathrm{moveD}(D, \ell))$

if $\exists\ \ell \in [n-1]$ *such that* ℓ *and* $\ell+1$ *are reversed in* π, *and* ($\ell+1 \notin U$ *or* $\pi([\ell+1]) = [\ell+1]$) *and* ($\ell \notin D$ *or* $\pi([\ell-1]) = [\ell-1]$) **then**

  return $s_\ell \cdot \mathrm{permutreeSort}(s_\ell \cdot \pi,\ \mathrm{moveU}(U \setminus \{\ell+1\}, \ell),$ $\mathrm{moveD}(D \setminus \{\ell\}, \ell))$

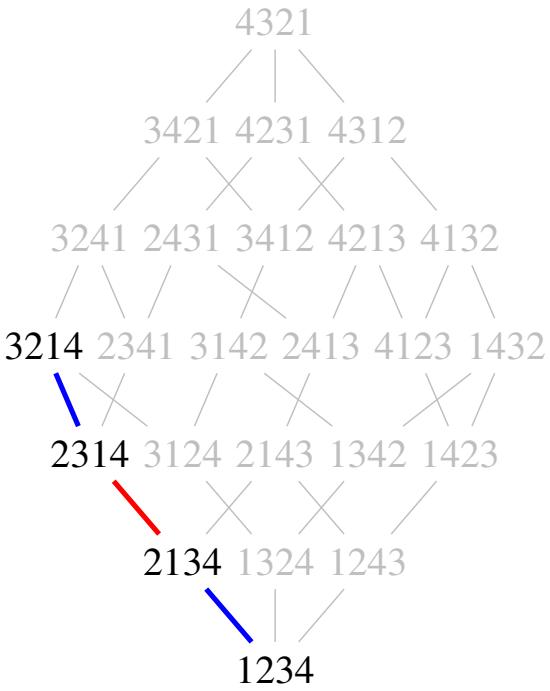
return ε

Algorithm 4: (U, D)-permutree sorting.

Example

π	w	U	D	ℓ	k
3214	ε	$\{3\}$	$\{2\}$	1	.
3124	s_1	$\{3\}$	$\{1\}$	2	3
2134	$s_1 \cdot s_2$	$\varnothing$	$\{1\}$	1	.
1234	$s_1 \cdot s_2 \cdot s_1$				

Table 3: The $(\{3\}, \{2\})$-permutree sorting of $\pi_2 := 3214$.



c-sorting [Rea07]

Let $\pi := 3421$ and take the *Coxeter element* $c := s_2 \cdot s_1 \cdot s_3$.

c-sorting [Rea07]

Let $\pi := 3421$ and take the *Coxeter element* $c := s_2 \cdot s_1 \cdot s_3$.

Consider the infinite word

$$c^\infty = c \cdot c \cdot c \cdots = s_2 \cdot s_1 \cdot s_3 \cdot s_2 \cdot s_1 \cdot s_3 \cdot s_2 \cdot s_1 \cdot s_3 \cdots .$$

c-sorting [Rea07]

Let $\pi := 3421$ and take the *Coxeter element* $c := s_2 \cdot s_1 \cdot s_3$.

Consider the infinite word

$$c^\infty = c \cdot c \cdot c \cdots = s_2 \cdot s_1 \cdot s_3 \cdot s_2 \cdot s_1 \cdot s_3 \cdot s_2 \cdot s_1 \cdot s_3 \cdots .$$

Out of all the reduced expressions of π, denote the lexicographically first in c^∞ as $\pi(c) = s_2 \cdot s_1 \cdot s_3 \cdot s_2 \cdot s_3$ and call it the *c-sorting word*.

c-sorting [Rea07]

Let $\pi := 3421$ and take the *Coxeter element* $c := s_2 \cdot s_1 \cdot s_3$.

Consider the infinite word

$$c^\infty = c \cdot c \cdot c \cdots = s_2 \cdot s_1 \cdot s_3 \cdot s_2 \cdot s_1 \cdot s_3 \cdot s_2 \cdot s_1 \cdot s_3 \cdots .$$

Out of all the reduced expressions of π, denote the lexicographically first in c^∞ as $\pi(c) = s_2 \cdot s_1 \cdot s_3 \cdot s_2 \cdot s_3$ and call it the *c-sorting word*.

If $Supp(\pi(c)) \supseteq Supp(\pi(c)) \supseteq Supp(\pi(c)) \supseteq \cdots$, we say that π is *c-sortable*.

Theorem (Pilaud, Pons, T. 2020)

For any Coxeter element c and permutation π, TFAE:

1. π is c-sortable,
2. the c-sorting word $\pi(c)$ is accepted by the automaton $\mathbb{P}(U_c, D_c)$,
3. there exists a reduced expression of π accepted by $\mathbb{P}(U_c, D_c)$,
4. for each j, there exists a reduced expression for π that is accepted by $\mathbb{U}(j)$ if $j \in U_c$ and $\mathbb{D}(j)$ if $j \in D_c$,
5. π avoids jki for $j \in U_c$ and kij for $j \in D_c$.

Recent developments and advantages

Recent developments and advantages

- Generalizable to type B.

Recent developments and advantages

- Generalizable to type B.
- Things work but new problems arise in other Coxeter types like type D and H.

Recent developments and advantages

- Generalizable to type B.
- Things work but new problems arise in other Coxeter types like type D and H.
- Computationally faster than doing lattice congruences in SageMath.

References


Donald E. Knuth.
The art of computer programming. Volume 3.
Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont., 1973.
Sorting and searching, Addison-Wesley Series in Computer Science and Information Processing.


Vincent Pilaud and Viviane Pons.
Permutrees.
Algebraic Combinatorics, 1(2):173–224, 2018.


Nathan Reading.
Clusters, Coxeter-sortable elements and noncrossing partitions.
Trans. Amer. Math. Soc., 359(12):5931–5958, 2007.