Permutrees: Permutation sorting, lattice quotients and automata

Daniel Tamayo Jiménez

Université Paris-Saclay (LISN)

Joint work with Vincent Pilaud and Viviane Pons

Journée de Combinatoire
Institut de Recherche Mathématique Avancée, Strasbourg
Novembre 4, 2021

Outline

- What is sorting?
- Lattice congruences
- Automatas
- Coxeter sorting

What is sorting?

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

Example: Stack sorting (Knuth 60's)

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

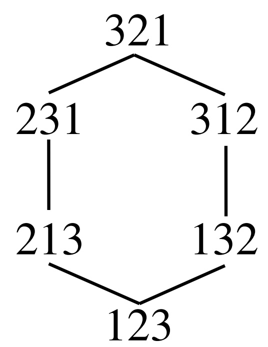
What is sorting?

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

Example: Stack sorting (Knuth 60's)

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

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



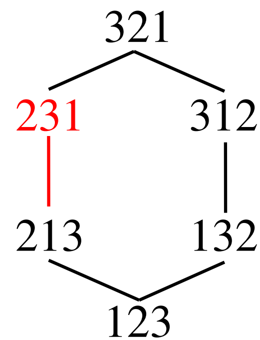
What is sorting?

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

Example: Stack sorting (Knuth 60's)

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

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



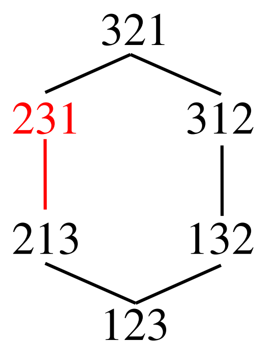
What is sorting?

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

Example: Stack sorting (Knuth 60's)

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

$321 = s_2 \cdot s_1 \cdot s_2$
$231 = s_1 \cdot s_2$
$312 = s_2 \cdot s_1$
$213 = s_1$
$132 = s_2$
$123 = e.$



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.

Examples

- $1243 = s_3$,
- $1423 = s_2 \cdot s_3$,
- $3421 = s_2 \cdot s_1 \cdot s_3 \cdot s_2 \cdot s_3$.

Weak order

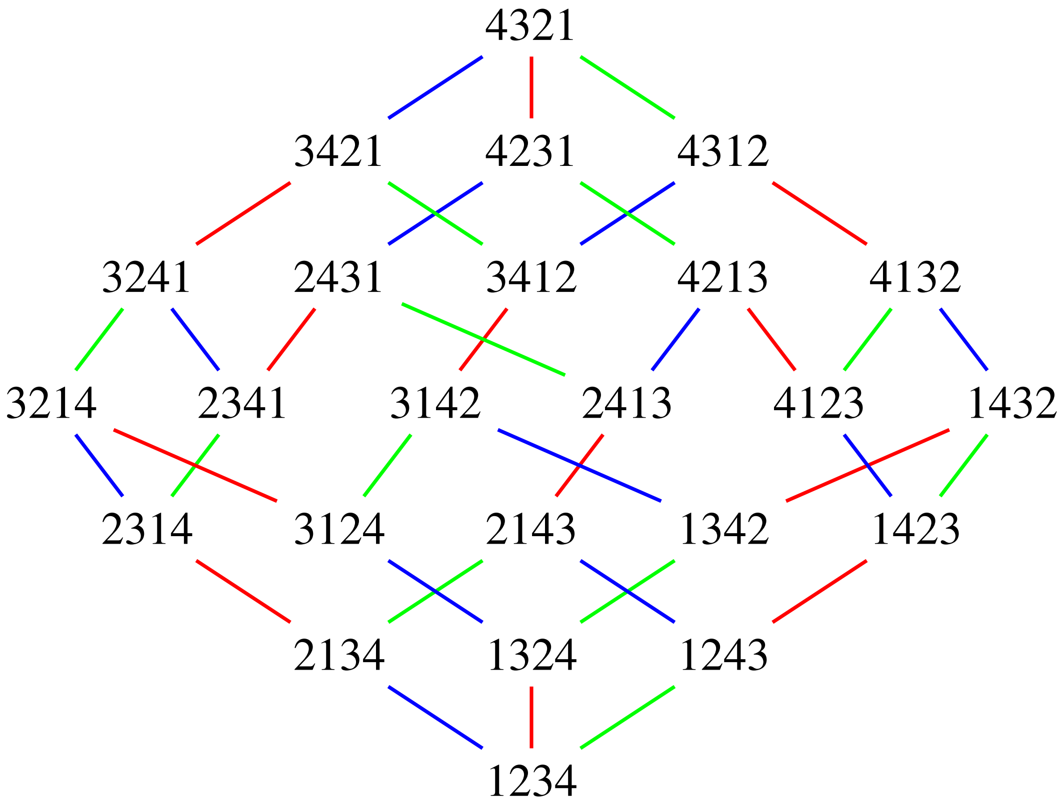


Figure 1: The (right) weak order of $\mathfrak{S}_4$ generated by s_1, s_2, s_3.

Weak order (inversions)

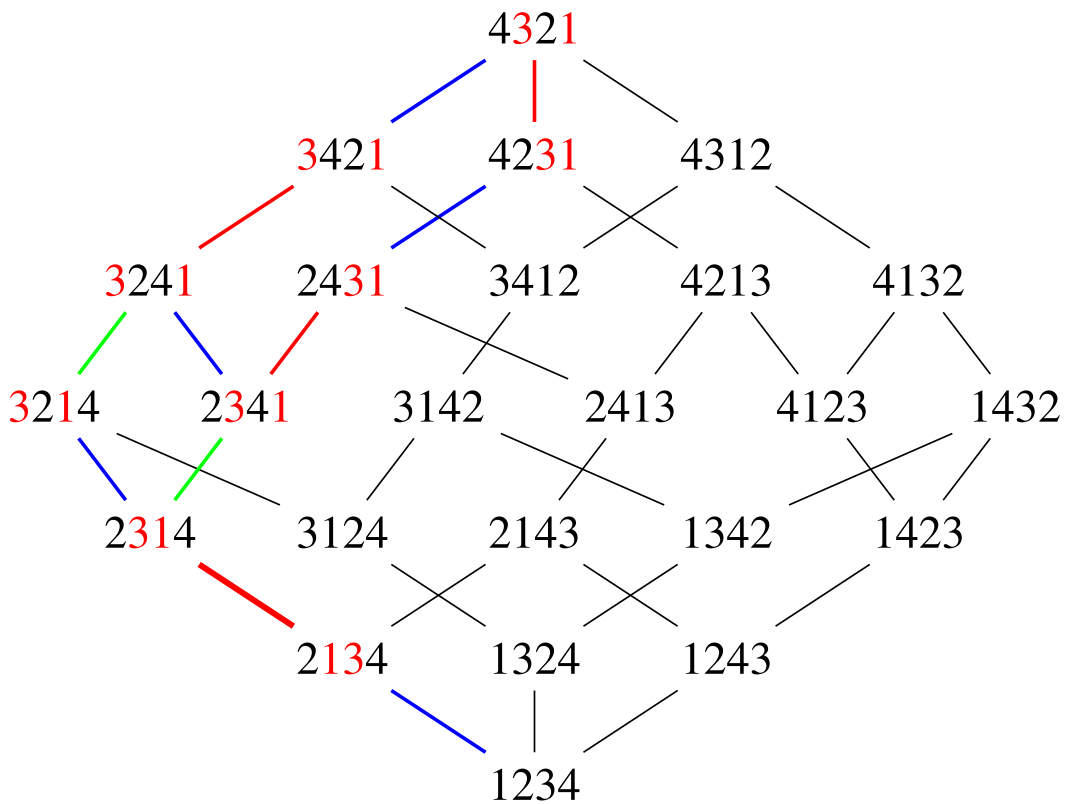


Figure 2: The (right) weak order of $\mathfrak{S}_4$ generated by s_1, s_2, s_3.

Weak order congruences

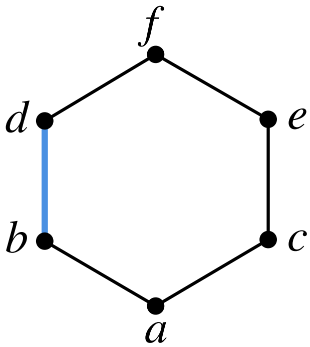
We want to take congruences that preserve meets and joins. That is,

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

Weak order congruences

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

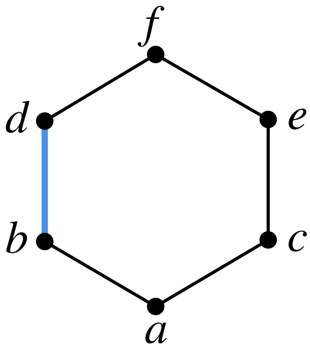
$$x \equiv x' \text{ and } y \equiv y' \implies x \vee y \equiv x' \vee y' \text{ and } x \wedge y \equiv x' \wedge y'$$

- 

Weak order congruences

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

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

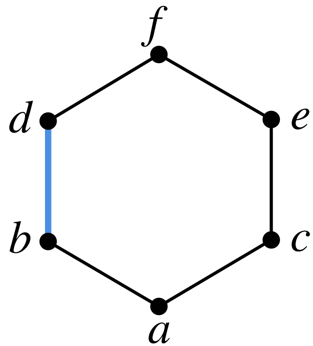
- 

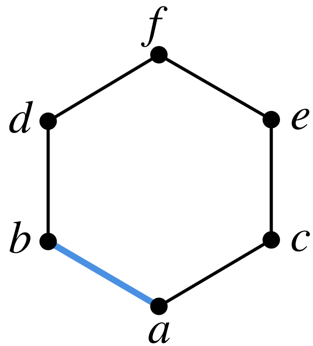
Nothing else is affected.

Weak order congruences

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

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

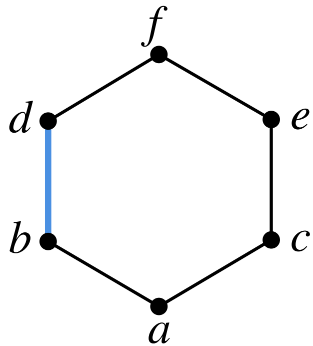
- 

Nothing else is affected.
- 

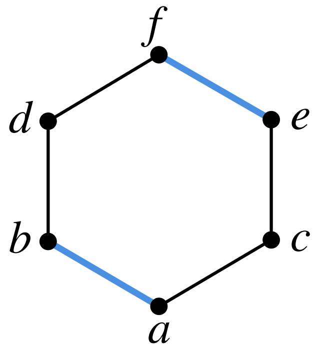
Weak order congruences

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.

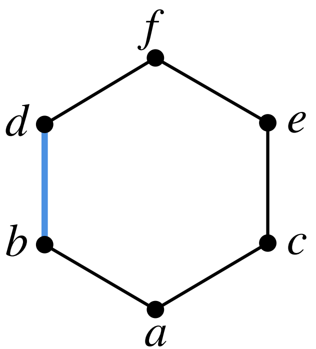
- 

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

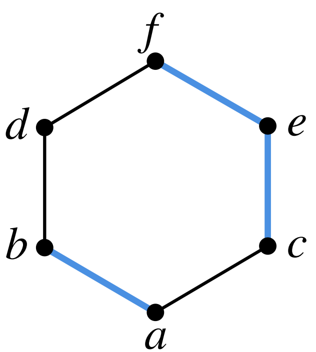
Weak order congruences

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.

- 

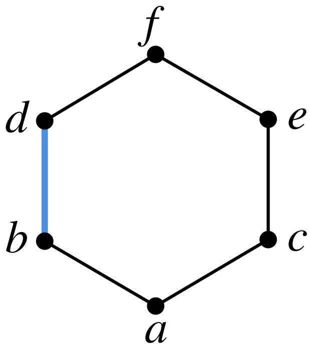
  $$e \vee b \equiv e \vee a \quad \Rightarrow \quad f \equiv e.$$

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

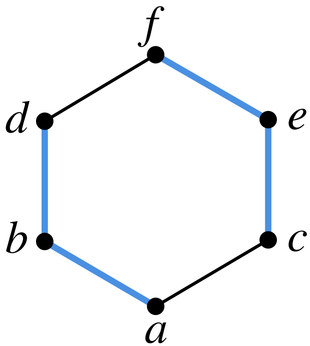
Weak order congruences

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

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

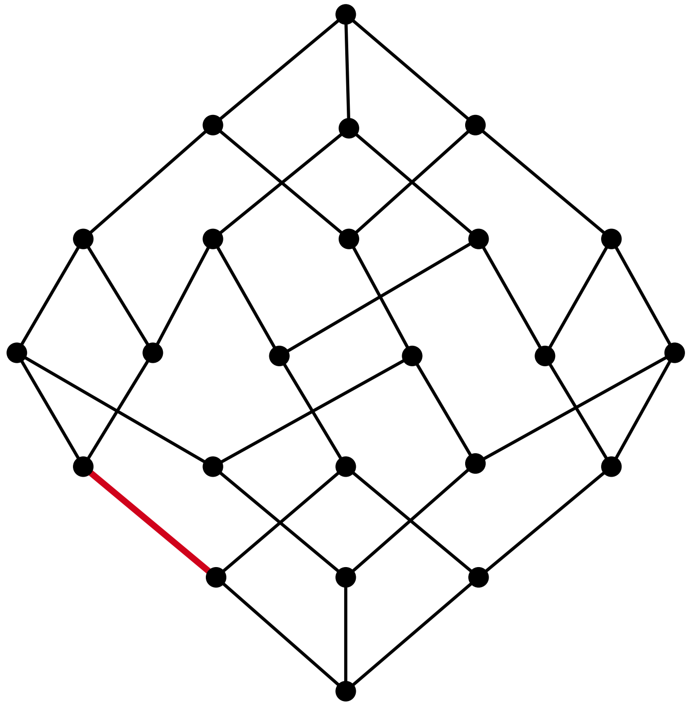
- 

Nothing else is affected.

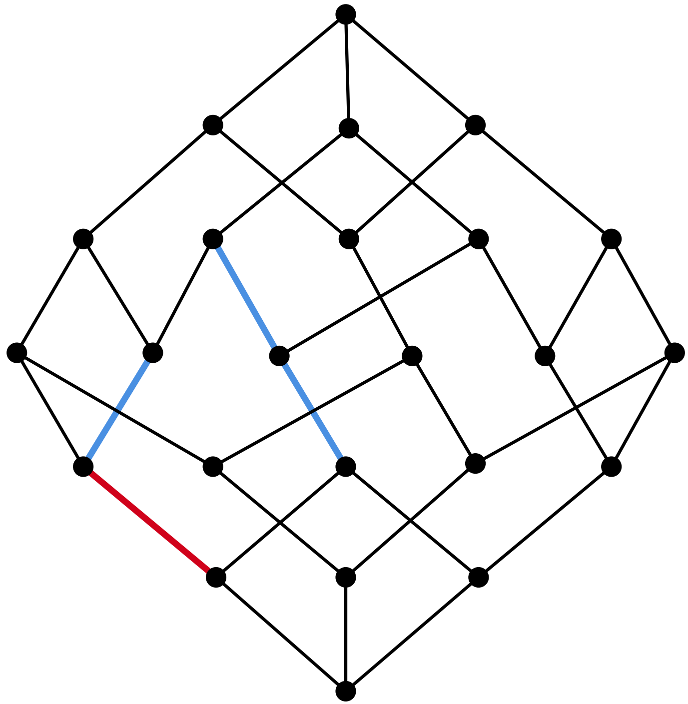
- 

$$e \vee b \equiv e \vee a \quad \Rightarrow \quad f \equiv e.$$
$$c \vee b \equiv c \vee a \quad \Rightarrow \quad f \equiv c.$$
$$d \wedge f \equiv d \wedge e \quad \Rightarrow \quad d \equiv a.$$

Lattice quotient example



Lattice quotient example



Lattice quotient example

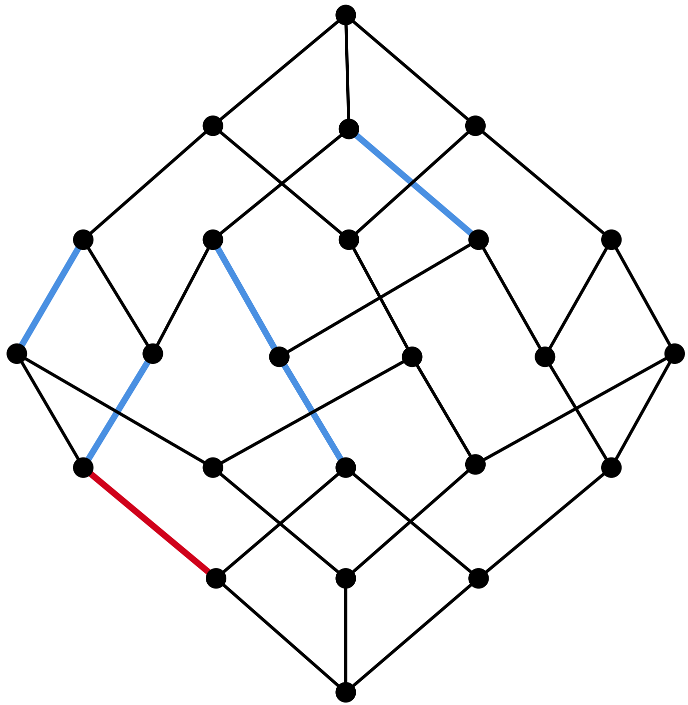


Figure 3: The resulting lattice quotient from contracting the red edge.

Weak order congruences

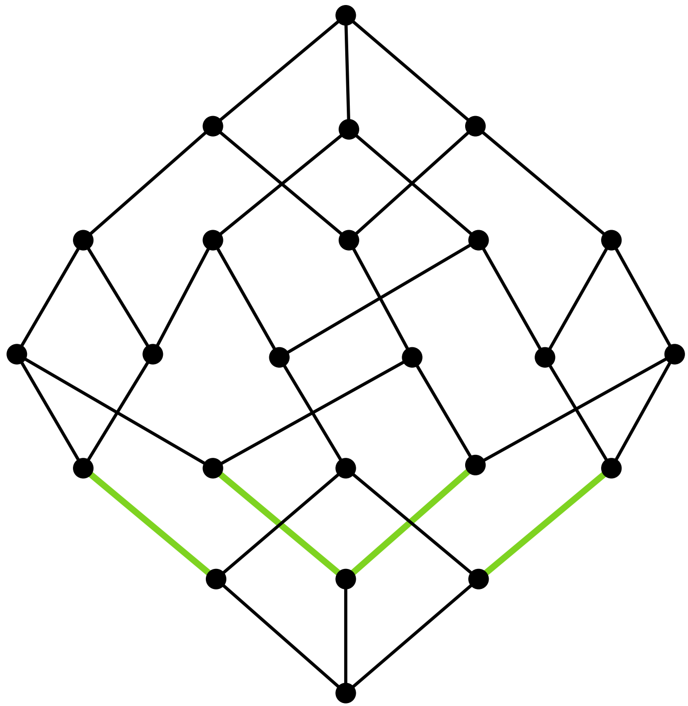


Figure 4: We study the contraction of any combination of green edges on $\mathfrak{S}_4$. We call these **permutree congruences**.

Why these particular ones?

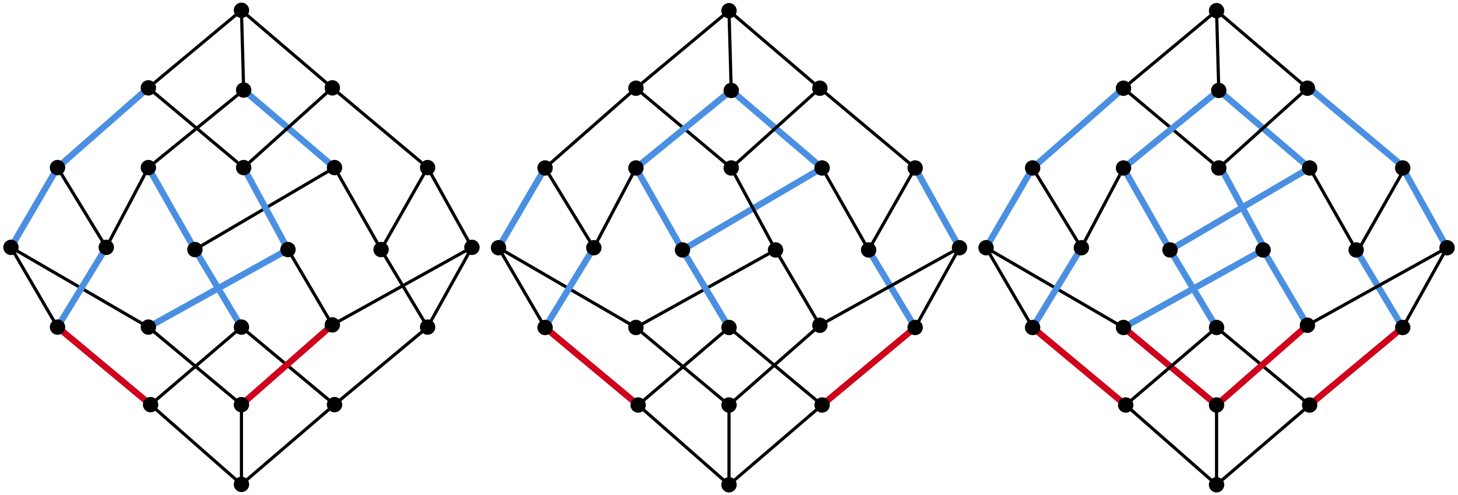


Figure 5: The Tamari lattice, a Cambrian lattice, and the boolean lattice as lattice quotients from contractions in red.

Coxeter Groups

It is a group generated by elements s_i that satisfy relations $(s_i s_j)^{m_{ij}} = e$ encoded as vertices and edges of the following graphs:

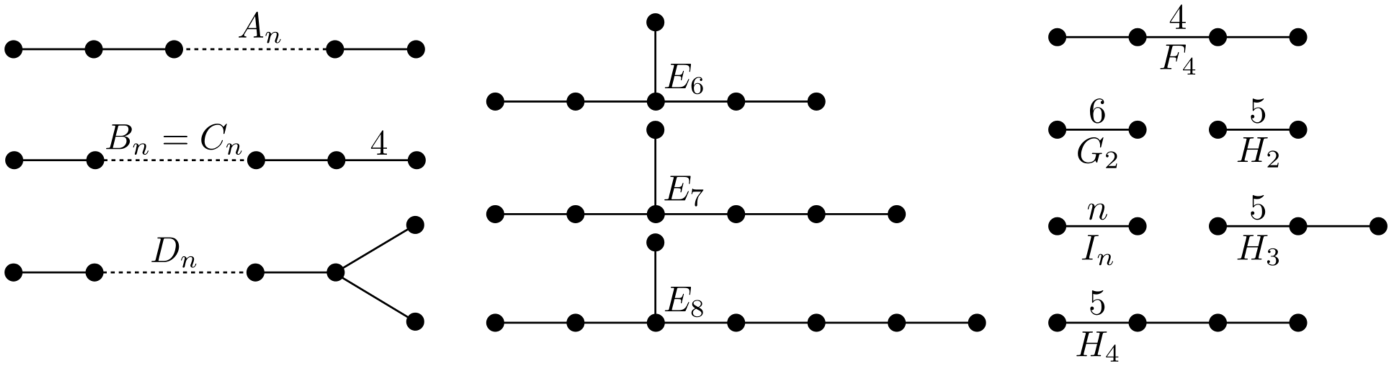


Figure 6: Finite Coxeter Groups (source: Wikipedia)

Coxeter Groups

It is a group generated by elements s_i that satisfy relations $(s_i s_j)^{m_{ij}} = e$ encoded as vertices and edges of the following graphs:

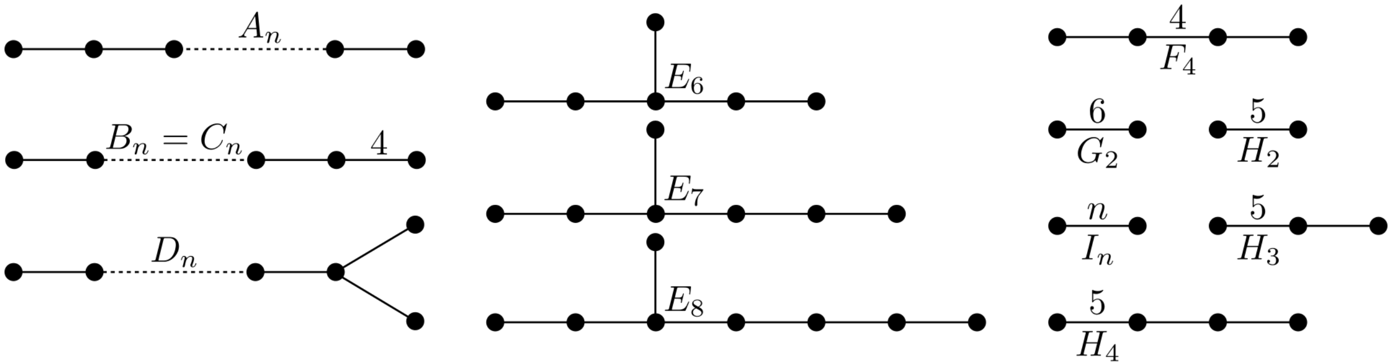


Figure 6: Finite Coxeter Groups (source: Wikipedia)

Each Coxeter Group has a weak order lattice associated to it where we can define permutree congruences.

Coxeter Groups

It is a group generated by elements s_i that satisfy relations $(s_i s_j)^{m_{ij}} = e$ encoded as vertices and edges of the following graphs:

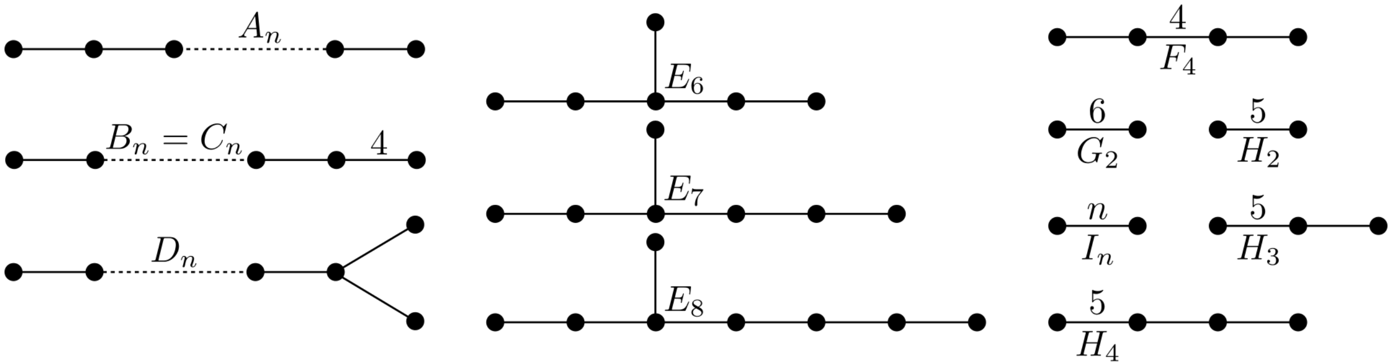


Figure 6: Finite Coxeter Groups (source: Wikipedia)

Each Coxeter Group has a weak order lattice associated to it where we can define permutree congruences.

Not all of them have combinatorial objects like permutations associated to them.

Why automata?

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

Why automata?

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

Different points of view have been found to study them:

- Lattice congruences
- Root systems
- Pattern avoidance.
- Reduced words

Why automata?

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

Different points of view have been found to study them:

- Lattice congruences
- Root systems
- Pattern avoidance.
- Reduced words (automata!)

Big objective

Characterize minimal elements of permutree congruences in all Coxeter groups.

Little objective

Get an algorithm based on reduced words that characterizes minimal elements of permutrees congruences in type A (permutations).

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

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

Translations

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

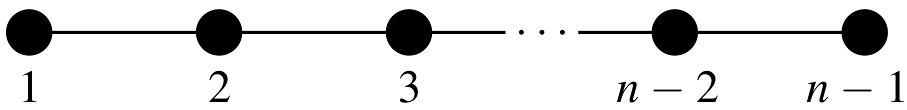


Figure 7: Coxeter graph of $\mathfrak{S}_n$.

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

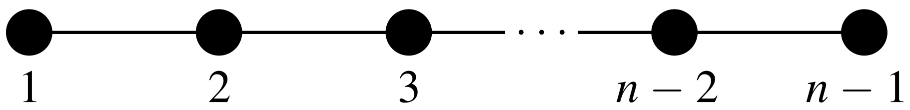
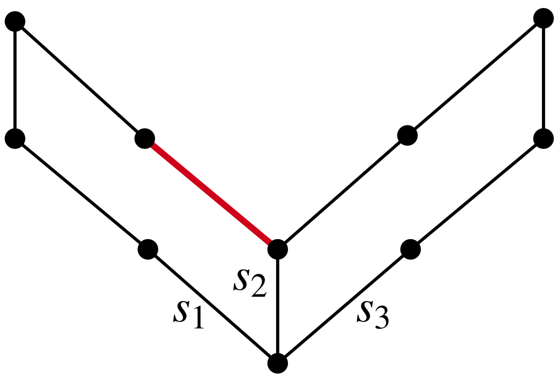


Figure 7: Coxeter graph of $\mathfrak{S}_n$.



Translations

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

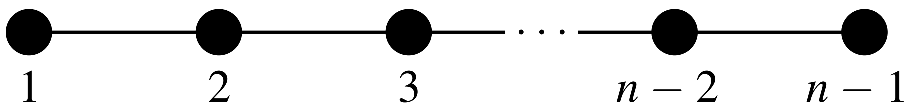
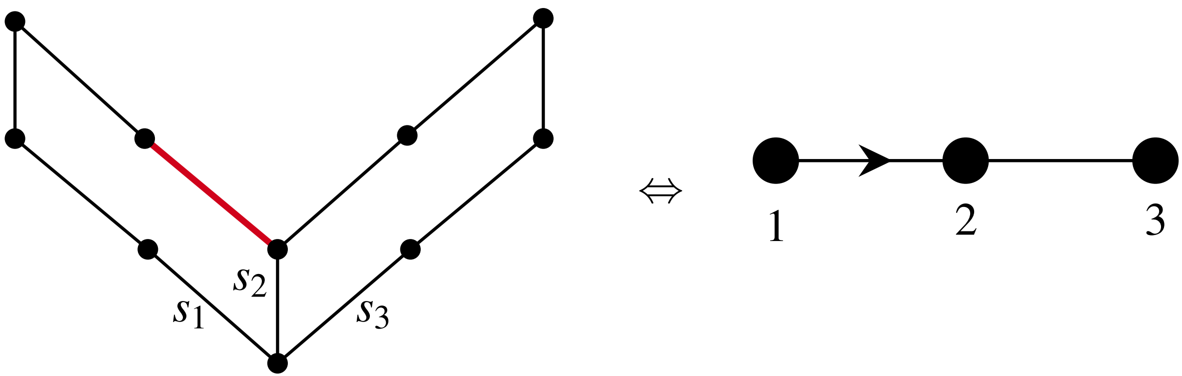


Figure 7: Coxeter graph of $\mathfrak{S}_n$.



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

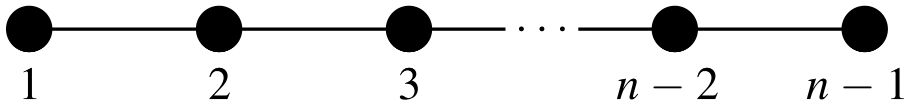
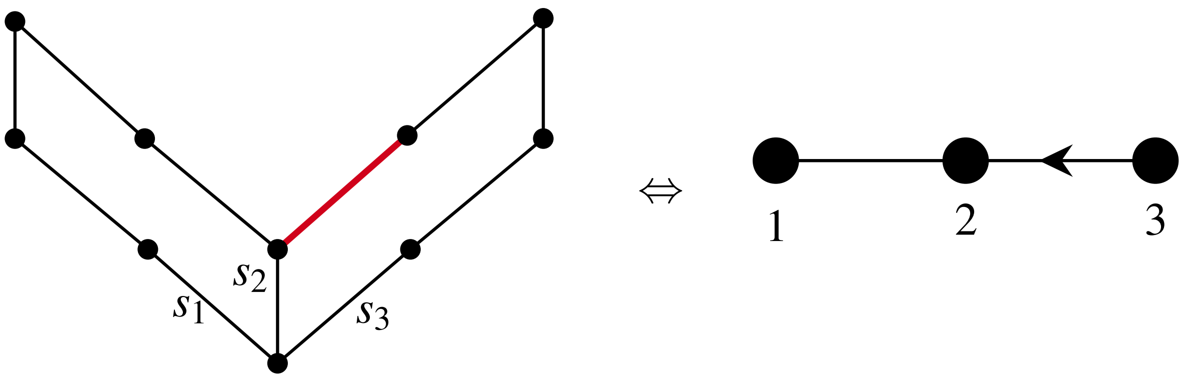


Figure 8: Coxeter graph of $\mathfrak{S}_n$.



Single orientation automata

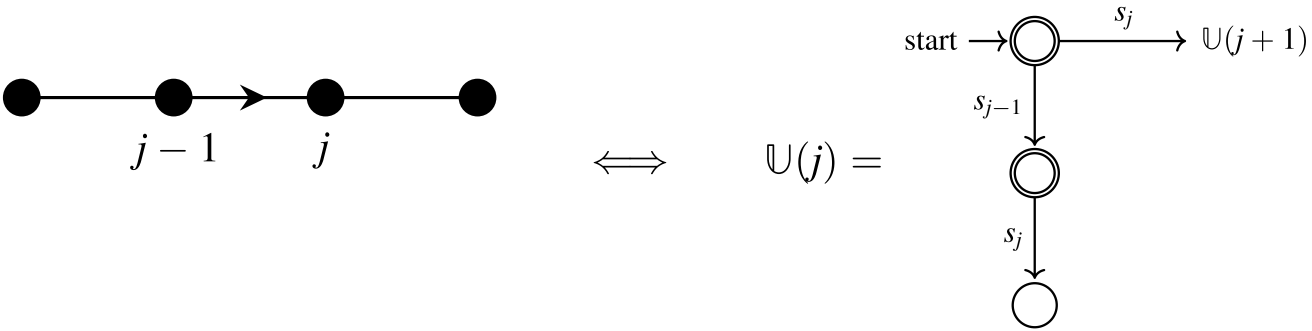


Figure 9: A single orientation and its automaton.

Single orientation automata

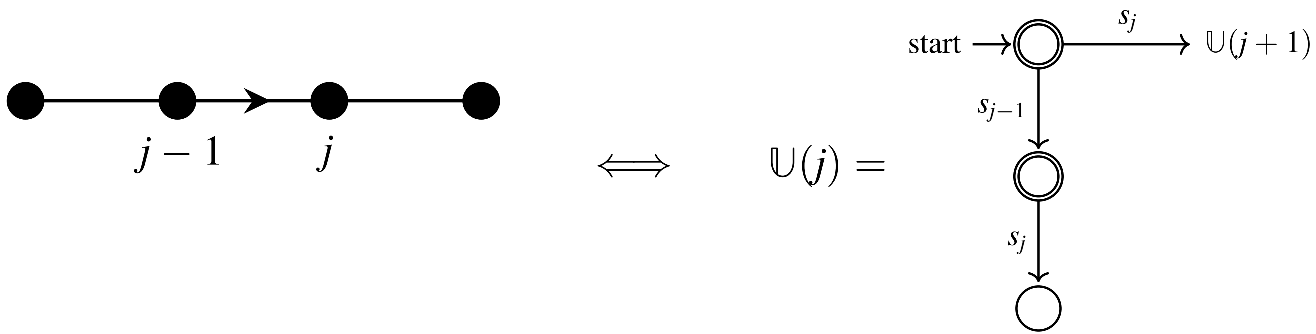


Figure 9: A single orientation and its automaton.

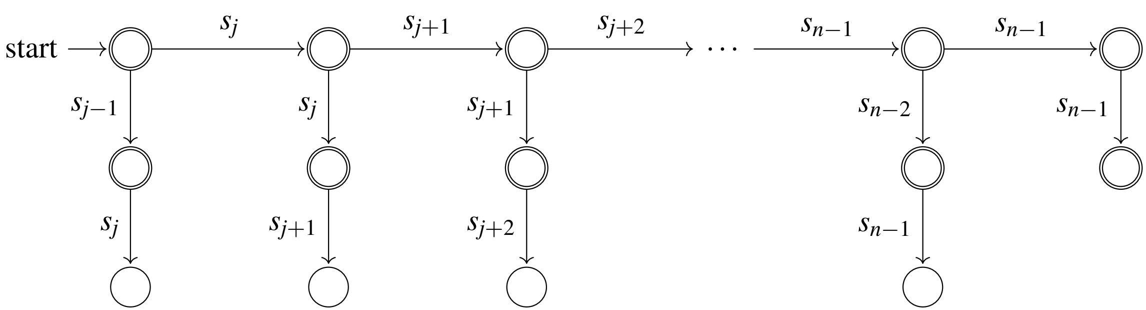


Figure 10: The complete automaton $\mathbb{U}(j)$.

Example

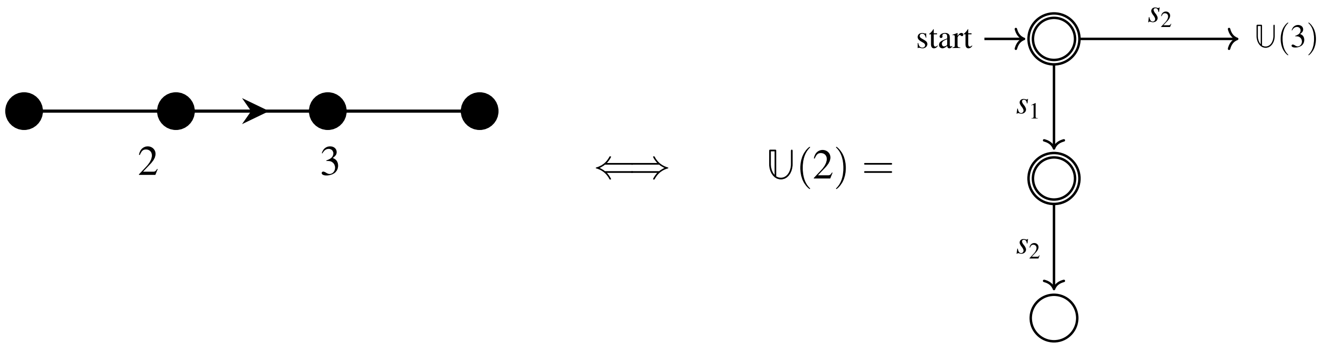


Figure 11: A single orientation and its automaton.

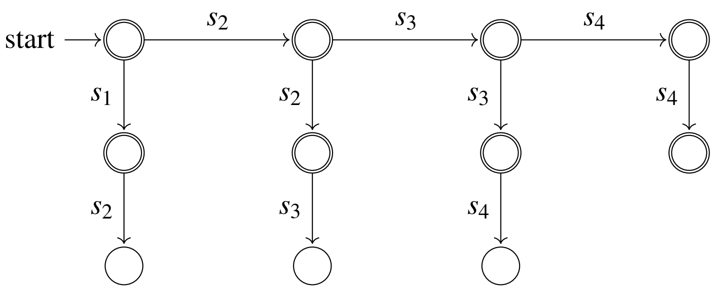


Figure 12: The complete automaton $\mathbb{U}(2)$ for $\mathfrak{S}_5$.

Example

- $s_3 \cdot s_2 \cdot s_1 \cdot s_2$ is accepted by $\mathbb{U}(2)$.
- $s_3 \cdot s_1 \cdot s_2 \cdot s_1$ is rejected by $\mathbb{U}(2)$.
- $s_1 \cdot s_3 \cdot s_2 \cdot s_1$ is rejected by $\mathbb{U}(2)$.

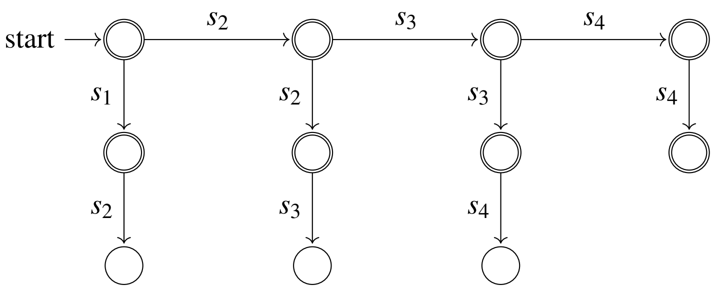


Figure 13: The complete automaton $\mathbb{U}(2)$ for $\mathfrak{S}_5$.

Properties of the automata

Theorem (Pilaud, Pons, T. 2020)

Fix $j \in \{2, \ldots, n-1\}$. 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!)

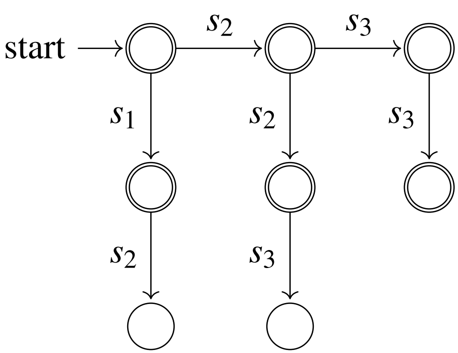
Properties of the automata

Theorem (Pilaud, Pons, T. 2020)

Fix $j \in \{2, \ldots, n-1\}$. 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.

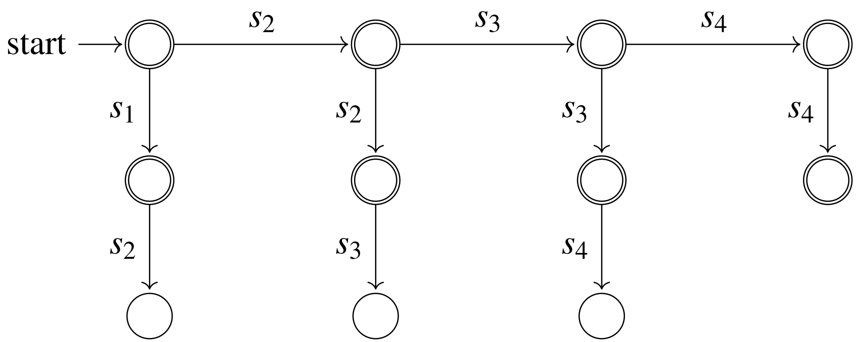


Figure 14: The complete automaton $\mathbb{U}(2)$ for $\mathfrak{S}_5$.

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.

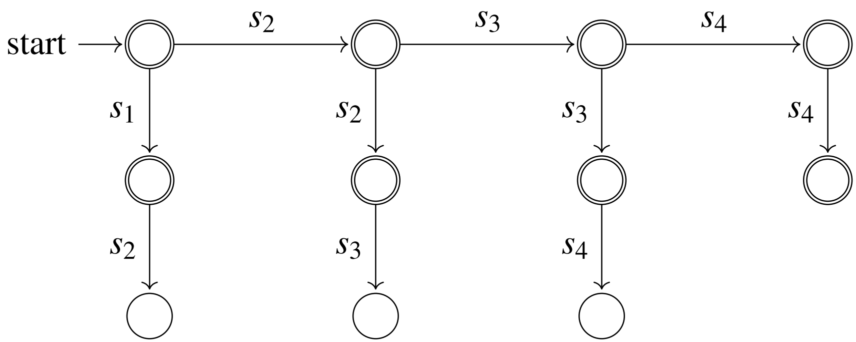


Figure 14: The complete automaton $\mathbb{U}(2)$ for $\mathfrak{S}_5$.

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.

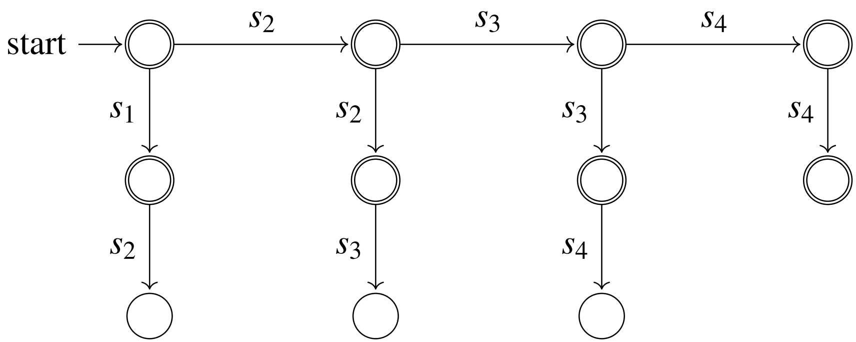


Figure 14: The complete automaton $\mathbb{U}(2)$ for $\mathfrak{S}_5$.

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.

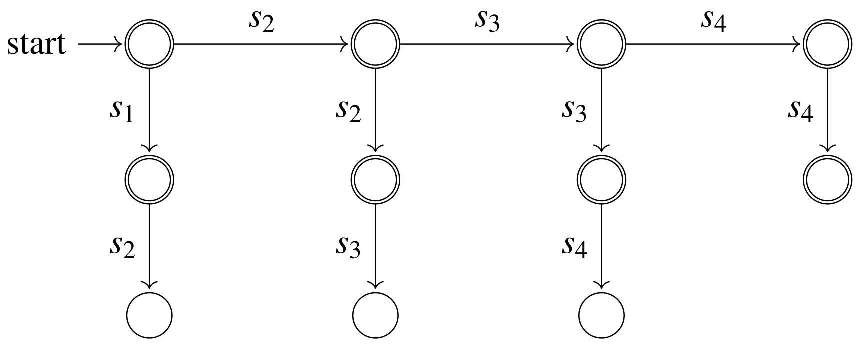


Figure 15: The complete automaton $\mathbb{U}(2)$ for $\mathfrak{S}_5$.

Proposition (Pilaud, Pons, T. 2020)

All reduced expressions of a permutation $\pi \in \mathfrak{S}_n$ end at

1. the same healthy state of $\mathbb{U}(j)$ if π keeps the values $[j]$ in the same relative order.
2. the same state of $\mathbb{U}(j)$ if π keeps the values $[n] \setminus [j-1]$ in the same relative order.
3. the same ill state of $\mathbb{U}(j)$ otherwise.

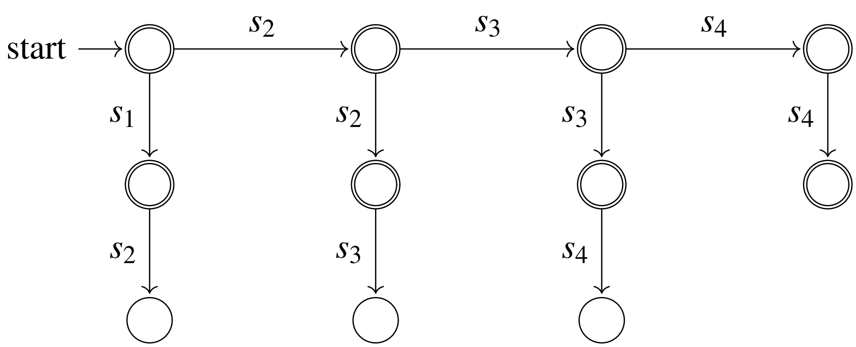


Figure 16: The complete automaton $\mathbb{U}(2)$ for $\mathfrak{S}_5$.

Accepted reduced words

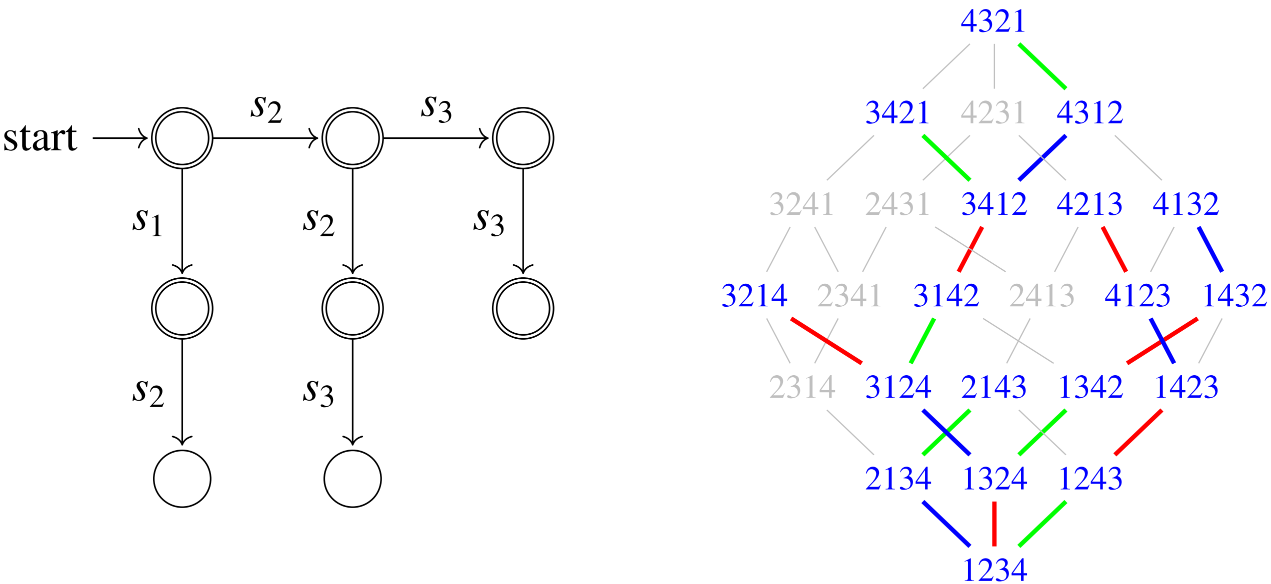


Figure 17: $\mathbb{U}(2)$ and its accepted reduced words.

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.

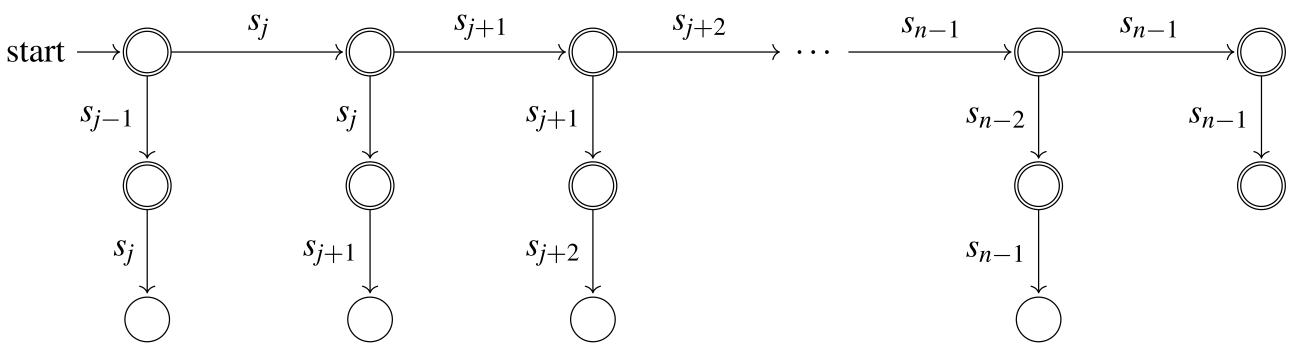


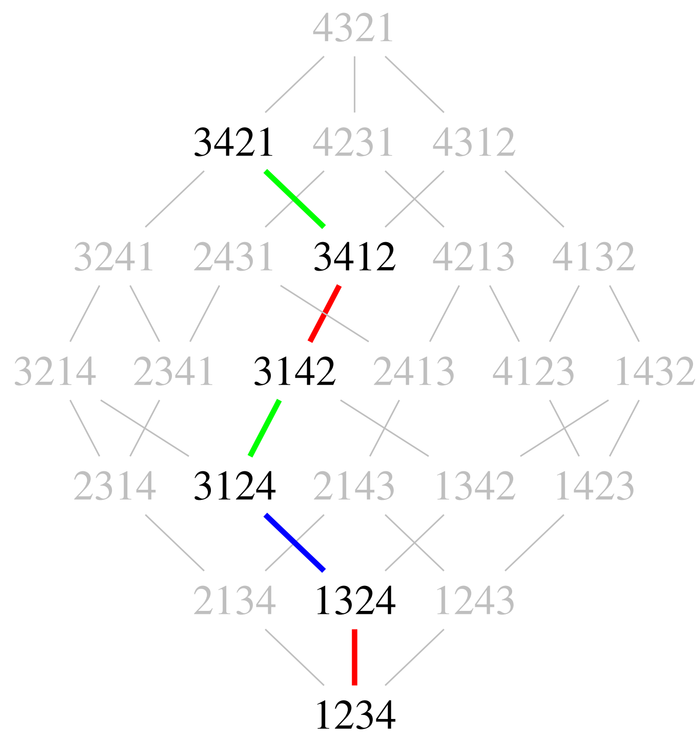
Figure 18: The complete automaton $\mathbb{U}(j)$.

Algorithm example

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

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 $\pi_2 := 3421$.

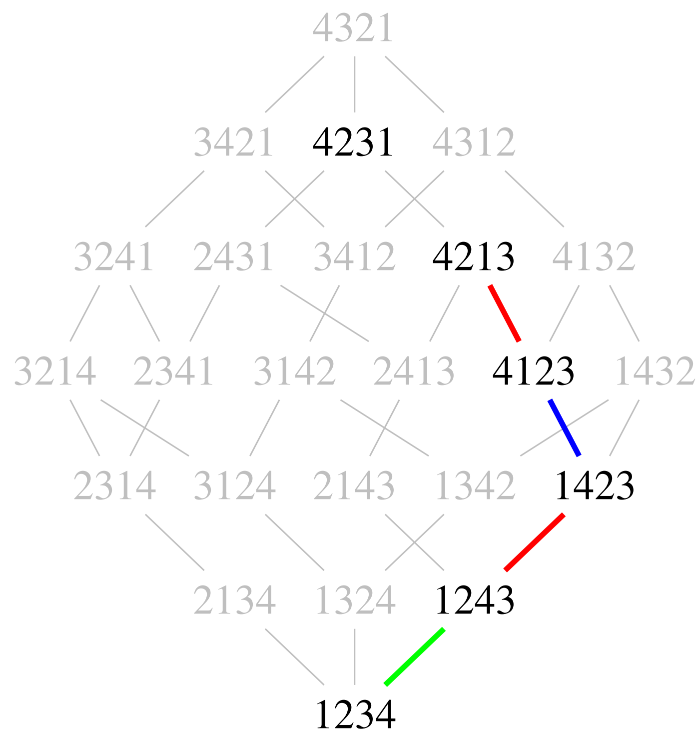


Algorithm non-example

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

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

Several orientations lead to multiple congruences, which correspond to multiple automata.

Multiple orientation automata

Several orientations lead to multiple congruences, which correspond to multiple automata.

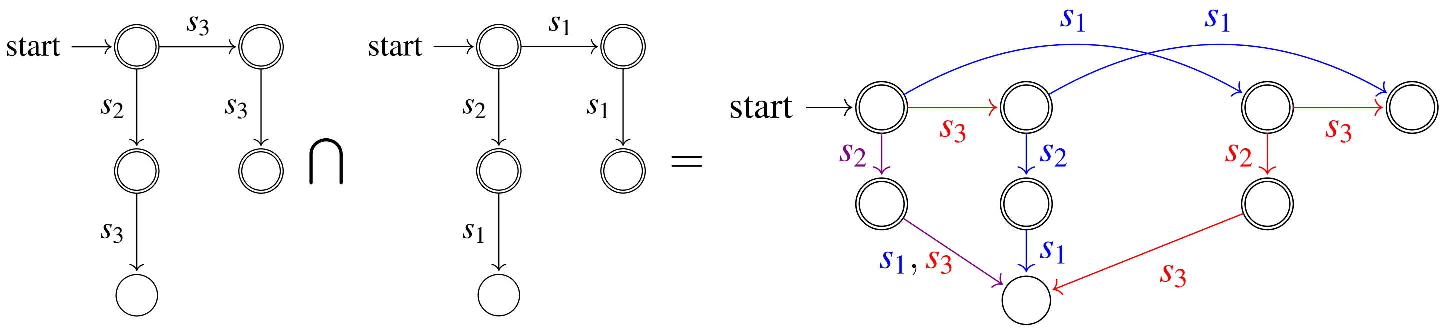
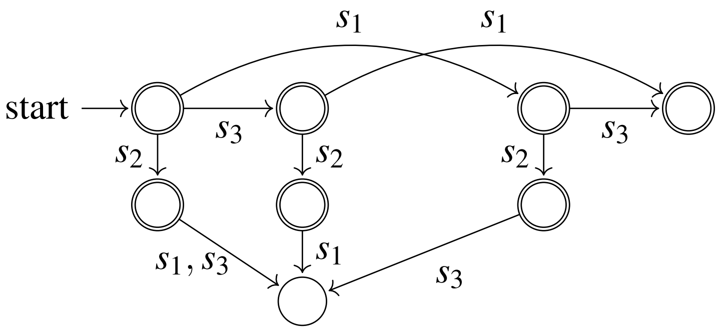


Figure 19: The automaton $\mathbb{P}(\{3\}, \{2\})$.

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

Intersection of automata



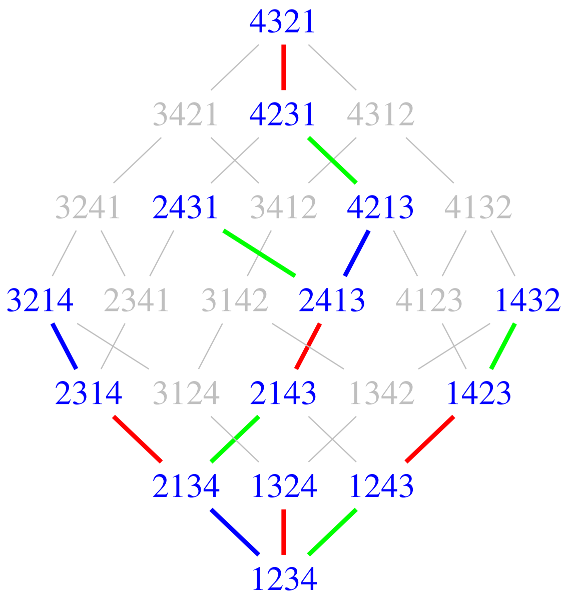
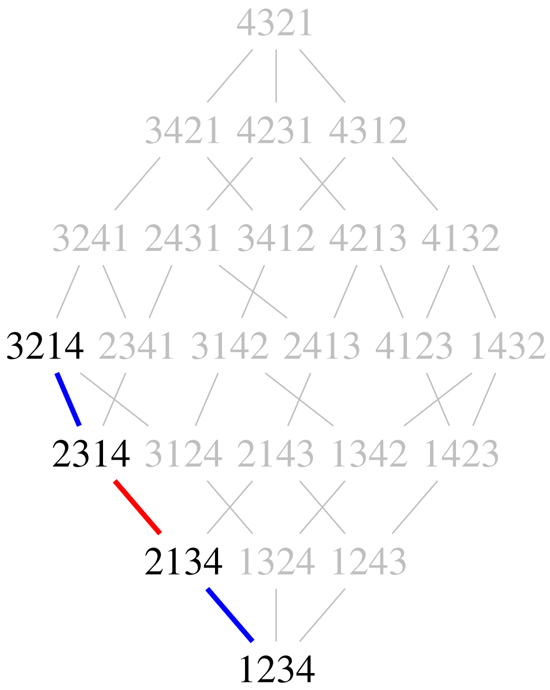


Figure 20: $\mathbb{P}(\{3\}, \{2\})$ and its accepted reduced words.

Algorithm 2 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 (Coxeter sorting) [Rea07]

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

c-sorting (Coxeter 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 (Coxeter 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 (Coxeter 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 $\mathit{Supp}(\pi(c)) \supseteq \mathit{Supp}(\pi(c)) \supseteq \mathit{Supp}(\pi(c)) \supseteq \cdots$, we say that π is *c-sortable*.

Theorem (Pilaud, Pons, T. 2020)

For any Coxeter element c and permutation π, TFAE:

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

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.

Recent developments and advantages

Recent developments and advantages

- Generalizable to type B.

Recent developments and advantages

- Generalizable to type B.
- Problems arise in other Coxeter types like type D and H.

Recent developments and advantages

- Generalizable to type B.
- Problems arise in other Coxeter types like type D and H.
- Computationally faster than doing lattice congruences in SageMath (albeit some details).

Other types

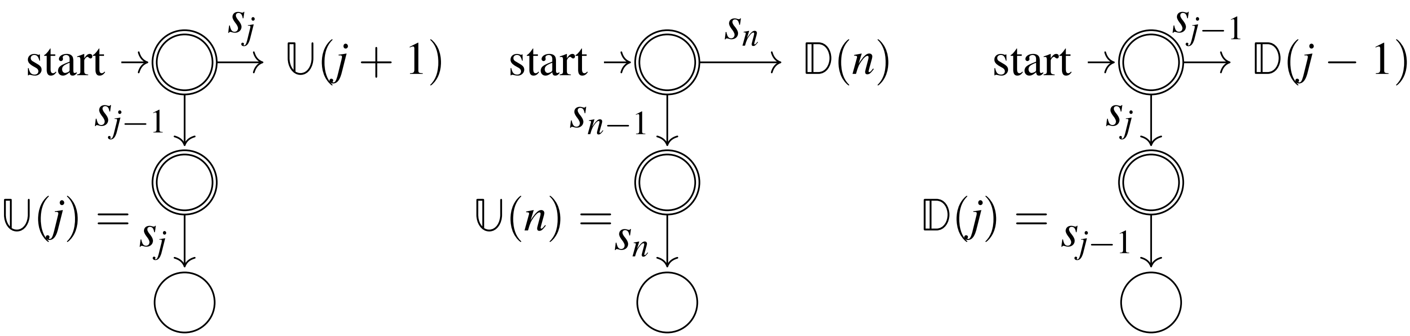


Figure 21: Recursive definition of the automata $\mathbb{U}(j)$ and $\mathbb{D}(j)$ in type B (above) and the corresponding automata that form $\mathbb{U}(2)$ in D_4 (below).