L'algèbre et le monoïde de biHecke d'un groupe de Coxeter fini et leurs représentations

Florent Hivert[1] Anne Schilling[2] Nicolas M. Thiéry[2,3]

[1]LITIS, Université Rouen, France

[2]University of California at Davis, USA

[3]Laboratoire de Mathématiques d'Orsay, Université Paris Sud, France

Caen, May 17th, 2011

arXiv:0711.1561v1 [math.RT]
arXiv:0804.3781v1 [math.RT]

Résumé

Dans cet exposé, nous présentons les résultats d'une série d'articles en collaboration avec Anne Schilling:
`arXiv:0711.1561`,`arXiv:0804.3781v3`,`arXiv:1012.1361`
Motivés par la théorie des représentations des algèbres de Hecke dégénérées et affine, ainsi que par le calcul de Schubert, nous associons à chaque groupe de Coxeter fini W une algèbre et un monoïde dits de biHecke. Leur construction s'appuie sur le modèle combinatoire usuel de la 0-algèbre de Hecke $H_0(W)$, i.e., pour le groupe symétrique, l'algèbre (ou le monoïde) engendré par les opérateurs de tri par bulles élémentaires. Plus précisément, ils sont obtenus comme algèbre ou monoïde engendrés conjointement par les opérateurs de tri et d'anti-tri.
Nous décrivons la théorie des représentations de l'algèbre de biHecke à l'aide de la combinatoire des descentes. De plus, nous montrons qu'il s'agit d'un quotient de l'algèbre de Hecke affine, expliquant par là les similarités de certaines représentations de cette dernière avec celles de $H_0(W)$. Finalement, la théorie des représentations du monoïde de biHecke étend celle de l'algèbre, grâce à une généralisation de la combinatoire des descentes relativement à tout intervale de W pour l'ordre faible.

Ces travaux s'appuient fortement sur l'exploration informatique. Nous illustrerons

notre démarche par quelques calculs typiques avec le logiciel Sage.

Schubert calculus, symmetric function

Divided differences operators:

$$\partial_i f(x_1, \dots, x_n) := \frac{f(\dots, x_i, x_{i+1}, \dots) - f(\dots, x_{i+1}, x_i, \dots)}{x_i - x_{i+1}}$$

$$\pi_i f := \partial_i(x_i f) \qquad \text{and} \qquad \hat{\pi}_i f := f - \pi_i f \qquad \dots$$

Problem

All these families satisfy the braids relations.
Describe the mixed relations ?

Applications:
appears in Iwahori-Hecke algebras, Schur symmetric functions, Schubert, Kazhdan-Lusztig polynomials, and Macdonald, (affine) Stanley symmetric functions, mathematical physics, Schur-Weyl duality for quantum groups, representations of $GL(\mathbb{F}_q)$, ...

Schubert calculus, symmetric function

Divided differences operators:

$$\partial_i f(x_1, \dots, x_n) := \frac{f(\dots, x_i, x_{i+1}, \dots) - f(\dots, x_{i+1}, x_i, \dots)}{x_i - x_{i+1}}$$

$$\pi_i f := \partial_i(x_i f) \qquad \text{and} \qquad \hat{\pi}_i f := f - \pi_i f \qquad \dots$$

Problem

All these families satisfy the braids relations.
Describe the mixed relations ?

Applications:
appears in Iwahori-Hecke algebras, Schur symmetric functions, Schubert, Kazhdan-Lusztig polynomials, and Macdonald, (affine) Stanley symmetric functions, mathematical physics, Schur-Weyl duality for quantum groups, representations of $GL(\mathbb{F}_q)$, ...

Representation theory: affine Hecke algebra

Theorem (Zelevinsky 1980)

Finite dim irrep of the affine Hecke algebra $\widetilde{\mathrm{H}}_n(q)$ are indexed by multisegments.

$$\widetilde{\mathrm{H}}_n(q) \simeq \mathrm{H}_n(q) \otimes \mathbb{C}[Y_1, \dots, Y_n]$$

$$Z(\widetilde{\mathrm{H}}_n(q)) = \mathbb{C}[Y_1, \dots, Y_n]^{\mathfrak{S}_{n+1}}$$

Principal specialization:

$$\mathcal{H}_n(q) := \widetilde{\mathrm{H}}_n(q) \,/\, \langle e_i(Y_1, \dots, Y_n) - e_i(1, q, \dots, q^{n-1}) \mid i = 1, \dots, n \rangle$$

- Bijection: Multisegments $\leftrightarrow$ subsets $I \subset \{1, \dots, n-1\}$.
- Base of irrep S_I indexed by descent classes of permutations.

Representation theory: affine Hecke algebra

Theorem (Zelevinsky 1980)

Finite dim irrep of the affine Hecke algebra $\widetilde{\mathrm{H}}_n(q)$ are indexed by multisegments.

$$\widetilde{\mathrm{H}}_n(q) \simeq \mathrm{H}_n(q) \otimes \mathbb{C}[Y_1, \ldots, Y_n]$$

$$Z(\widetilde{\mathrm{H}}_n(q)) = \mathbb{C}[Y_1, \ldots, Y_n]^{\mathfrak{S}_{n+1}}$$

Principal specialization:

$$\mathcal{H}_n(q) := \widetilde{\mathrm{H}}_n(q) \,/\, \langle e_i(Y_1, \ldots, Y_n) - e_i(1, q, \ldots, q^{n-1}) \mid i = 1, \ldots, n \rangle$$

- Bijection: Multisegments $\leftrightarrow$ subsets $I \subset \{1, \ldots, n-1\}$.
- Base of irrep S_I indexed by descent classes of permutations.

Representation theory: affine Hecke algebra

Theorem (Zelevinsky 1980)

Finite dim irrep of the affine Hecke algebra $\widetilde{\mathrm{H}}_n(q)$ are indexed by multisegments.

$$\widetilde{\mathrm{H}}_n(q) \simeq \mathrm{H}_n(q) \otimes \mathbb{C}[Y_1, \dots, Y_n]$$

$$Z(\widetilde{\mathrm{H}}_n(q)) = \mathbb{C}[Y_1, \dots, Y_n]^{\mathfrak{S}_{n+1}}$$

Principal specialization:

$$\mathcal{H}_n(q) := \widetilde{\mathrm{H}}_n(q) \,/\, \langle e_i(Y_1, \dots, Y_n) - e_i(1, q, \dots, q^{n-1}) \mid i = 1, \dots, n \rangle$$

- Bijection: Multisegments $\leftrightarrow$ subsets $I \subset \{1, \dots, n-1\}$.
- Base of irrep S_I indexed by descent classes of permutations.

Representation theory: 0-Hecke monoid

Definition (0-Hecke monoid $H_0(W)$ of a Coxeter group W)

Generators : $\langle \pi_1, \pi_2, \ldots \rangle$ (simple reflections)
Relations: $\pi_i^2 = \pi_i$ and braid relations

Theorem (Norton 1979-Carter 1986)

- *Simple modules S_I indexed by subsets $I \subset \{1, \ldots, n-1\}$.*
- *Base of indecomposable projective modules P_I indexed by descent classes of permutations.*

Representation theory: 0-Hecke monoid

Definition (0-Hecke monoid $H_0(W)$ of a Coxeter group W)

Generators : $\langle \pi_1, \pi_2, \dots \rangle$ (simple reflections)
Relations: $\pi_i^2 = \pi_i$ and braid relations

Theorem (Norton 1979-Carter 1986)

- *Simple modules S_I indexed by subsets $I \subset \{1, \dots, n-1\}$.*
- *Base of indecomposable projective modules P_I indexed by descent classes of permutations.*

Affine and 0 Hecke Algebras

- **Simple module** for the principal specialization $\mathcal{H}_n(q)$:
- **Indecomposable projective modules** for $H_0(W)$:

Combinatorics

- Indexed by subsets $I \subset \{1, \ldots, n-1\}$.
- Base indexed by descent classes.

Problem

Explain this coincidence of combinatorics

Affine and 0 Hecke Algebras

- **Simple module** for the principal specialization $\mathcal{H}_n(q)$:
- **Indecomposable projective modules** for $H_0(W)$:

Combinatorics

- Indexed by subsets $I \subset \{1, \ldots, n-1\}$.
- Base indexed by descent classes.

Problem

Explain this coincidence of combinatorics

Coxeter groups

Definition (Coxeter group W)

Generators : $(s_i)_{i\in S}$ (simple reflections)
Relations: $s_i^2 = 1$ and $\underbrace{s_i s_j \cdots}_{m_{i,j}} = \underbrace{s_j s_i \cdots}_{m_{i,j}}$, for $i \neq j$

Group algebra: $\mathbb{C}[W]$

Example (Type A_n: symmetric group $\mathfrak{S}_{n+1}$)

Generators: $(s_i)_{i=1,\ldots,n}$ (elementary transpositions)
Relations:

$$s_i^2 = 1 \qquad \text{for all } 1 \leq i \leq n,$$
$$s_i s_j = s_j s_i \qquad \text{for all } |i-j| > 1,$$
$$s_i s_{i+1} s_i = s_{i+1} s_i s_{i+1} \qquad \text{for all } 1 \leq i \leq n-1.$$

Coxeter groups

Definition (Coxeter group W)

Generators : $(s_i)_{i \in S}$ (simple reflections)
Relations: $s_i^2 = 1$ and $\underbrace{s_i s_j \cdots}_{m_{i,j}} = \underbrace{s_j s_i \cdots}_{m_{i,j}}$, for $i \neq j$

Group algebra: $\mathbb{C}[W]$

Example (Type A_n: symmetric group $\mathfrak{S}_{n+1}$)

Generators: $(s_i)_{i=1,\ldots,n}$ (elementary transpositions)
Relations:

$$
\begin{aligned}
s_i^2 &= 1 && \text{for all } 1 \leq i \leq n, \\
s_i s_j &= s_j s_i && \text{for all } |i-j| > 1, \\
s_i s_{i+1} s_i &= s_{i+1} s_i s_{i+1} && \text{for all } 1 \leq i \leq n-1.
\end{aligned}
$$

Definition (0-Hecke algebra $H(W)(0)$)

Generators : $(\pi_i)_{i\in S}$

Relations: $\pi_i^2 = \pi_i$ and $\underbrace{\pi_i\pi_j\cdots}_{m_{i,j}} = \underbrace{\pi_j\pi_i\cdots}_{m_{i,j}}$ for $i \neq j$

Basis: $(\pi_w)_{w\in W}$

Example (Type A_n)

Generators: $(\pi_i)_{i=1,\ldots,n}$ (adjacent comparators)

Relations:

$$\pi_i^2 = \pi_i \quad \text{for all } 1 \leq i \leq n,$$

$$\pi_i\pi_j = \pi_j\pi_i \quad \text{for all } |i-j| > 1,$$

$$\pi_i\pi_{i+1}\pi_i = \pi_{i+1}\pi_i\pi_{i+1} \quad \text{for all } 1 \leq i \leq n-1.$$

Definition (0-Hecke algebra H(W)(0))

Generators : $(\pi_i)_{i\in S}$
Relations: $\pi_i^2 = \pi_i$ and $\underbrace{\pi_i\pi_j\cdots}_{m_{i,j}} = \underbrace{\pi_j\pi_i\cdots}_{m_{i,j}}$ for $i \neq j$

Basis: $(\pi_w)_{w\in W}$

Example (Type A_n)

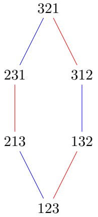
Generators: $(\pi_i)_{i=1,\ldots,n}$ (adjacent comparators)
Relations:

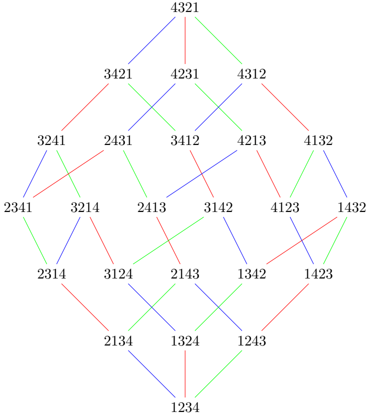
$$\begin{aligned} \pi_i^2 &= \pi_i && \text{for all } 1 \leq i \leq n, \\ \pi_i\pi_j &= \pi_j\pi_i && \text{for all } |i-j| > 1, \\ \pi_i\pi_{i+1}\pi_i &= \pi_{i+1}\pi_i\pi_{i+1} && \text{for all } 1 \leq i \leq n-1. \end{aligned}$$

0-Hecke algebras and bubble sort

Example (Right regular representation for type A)

13854627	13854627	13854627
$\downarrow s_2$ $\downarrow s_6$	$\downarrow \pi_2$ $\downarrow \pi_6$	$\downarrow \overline{\pi}_2$ $\downarrow \overline{\pi}_6$
18354267	18354627	13854267
$\downarrow s_2$ $\downarrow s_6$	$\downarrow \pi_2$ $\downarrow \pi_6$	$\downarrow \overline{\pi}_2$ $\downarrow \overline{\pi}_6$
13854627	18354627	13854267


321
231
312
213
132
123


4321
3421
4231
4312
3241
2431
3412
4213
4132
2341
3214
2413
3142
4123
1432
2314
3124
2143
1342
1423
2134
1324
1243
1234

Hecke algebras

Take q_1 and q_2 parameters, and set $q := -\frac{q_1}{q_2}$.

Definition (Hecke algebra $\mathrm{H}(W)(q_1, q_2)$)

Generators : $(T_i)_{i \in S}$ Relations: $(T_i - q_1)(T_i - q_2) = 0$ and $\underbrace{T_i T_j \cdots}_{m_{i,j}} = \underbrace{T_j T_i \cdots}_{m_{i,j}}$, for $i \neq j$

Basis: $(T_w)_{w \in W}$

- At $q = 1$: group algebra $\mathbb{C}[W]$
- At $q = 0$: 0-Hecke algebra $\mathrm{H}(W)(0)$
- At $q_1 = q_2 = 0$: nilCoxeter algebra
- At q not 0 nor a root of unity: isomorphic to $\mathbb{C}[W]$

Realization of T_i as operator in $\mathrm{End}(\mathbb{C}W)$:

$$T_i := (q_1 + q_2)\pi_i - q_1 s_i$$

Hecke algebras

Take q_1 and q_2 parameters, and set $q := -\frac{q_1}{q_2}$.

Definition (Hecke algebra $\mathrm{H}(W)(q_1, q_2)$)

Generators : $(T_i)_{i\in S}$ Relations: $(T_i - q_1)(T_i - q_2) = 0$ and

$$\underbrace{T_i T_j \cdots}_{m_{i,j}} = \underbrace{T_j T_i \cdots}_{m_{i,j}}, \text{ for } i \neq j$$

Basis: $(T_w)_{w\in W}$

- At $q = 1$: group algebra $\mathbb{C}[W]$
- At $q = 0$: 0-Hecke algebra $\mathrm{H}(W)(0)$
- At $q_1 = q_2 = 0$: nilCoxeter algebra
- At q not 0 nor a root of unity: isomorphic to $\mathbb{C}[W]$

Realization of T_i as operator in $\mathrm{End}(\mathbb{C}W)$:

$$T_i := (q_1 + q_2)\pi_i - q_1 s_i$$

Hecke algebras

Take q_1 and q_2 parameters, and set $q := -\frac{q_1}{q_2}$.

Definition (Hecke algebra $\mathrm{H}(W)(q_1, q_2)$)

Generators : $(T_i)_{i \in S}$ Relations: $(T_i - q_1)(T_i - q_2) = 0$ and $\underbrace{T_i T_j \cdots}_{m_{i,j}} = \underbrace{T_j T_i \cdots}_{m_{i,j}}$, for $i \neq j$

Basis: $(T_w)_{w \in W}$

- At $q = 1$: group algebra $\mathbb{C}[W]$
- At $q = 0$: 0-Hecke algebra $\mathrm{H}(W)(0)$
- At $q_1 = q_2 = 0$: nilCoxeter algebra
- At q not 0 nor a root of unity: isomorphic to $\mathbb{C}[W]$

Realization of T_i as operator in $\mathrm{End}(\mathbb{C}W)$:

$$T_i := (q_1 + q_2)\pi_i - q_1 s_i$$

Hecke algebras

Take q_1 and q_2 parameters, and set $q := -\frac{q_1}{q_2}$.

Definition (Hecke algebra $\mathrm{H}(W)(q_1, q_2)$)

Generators : $(T_i)_{i \in S}$ Relations: $(T_i - q_1)(T_i - q_2) = 0$ and $\underbrace{T_i T_j \cdots}_{m_{i,j}} = \underbrace{T_j T_i \cdots}_{m_{i,j}}$, for $i \neq j$

Basis: $(T_w)_{w \in W}$

- At $q = 1$: group algebra $\mathbb{C}[W]$
- At $q = 0$: 0-Hecke algebra $\mathrm{H}(W)(0)$
- At $q_1 = q_2 = 0$: nilCoxeter algebra
- At q not 0 nor a root of unity: isomorphic to $\mathbb{C}[W]$

Realization of T_i as operator in $\mathrm{End}(\mathbb{C}W)$:

$$T_i := (q_1 + q_2)\pi_i - q_1 s_i$$

BiHecke algebras

A silly idea during a brainstorm (Thibon, Novelli, H., T., 2003)

Definition (BiHecke algebra HW of a Coxeter group W)

Glue $\mathbb{C}[W]$ and $H(W)(0)$ on their right regular representations:

$$HW := \langle (\pi_i, s_i)_{i \in S} \rangle \quad \subset \quad \mathrm{End}(\mathbb{C}W)$$

- Any interesting structure?
- Contains all Hecke algebras by construction
- Type A: dimension and dimension of the radical in the Sloane!

BiHecke algebras

A silly idea during a brainstorm (Thibon, Novelli, H., T., 2003)

Definition (BiHecke algebra HW of a Coxeter group W)

Glue $\mathbb{C}[W]$ and $\mathrm{H}(W)(0)$ on their right regular representations:

$$HW := \langle(\pi_i, s_i)_{i \in S}\rangle \quad \subset \quad \mathrm{End}(\mathbb{C}W)$$

- Any interesting structure?
- Contains all Hecke algebras by construction
- Type A: dimension and dimension of the radical in the Sloane!

BiHecke algebras

A silly idea during a brainstorm (Thibon, Novelli, H., T., 2003)

Definition (BiHecke algebra HW of a Coxeter group W)

Glue $\mathbb{C}[W]$ and H(W)(0) on their right regular representations:

$$HW := \langle (\pi_i, s_i)_{i \in S} \rangle \quad \subset \quad \mathrm{End}(\mathbb{C}W)$$

- Any interesting structure?
- Contains all Hecke algebras by construction
- Type A: dimension and dimension of the radical in the Sloane!

BiHecke algebras

A silly idea during a brainstorm (Thibon, Novelli, H., T., 2003)

Definition (BiHecke algebra HW of a Coxeter group W)

Glue $\mathbb{C}[W]$ and H$(W)(0)$ on their right regular representations:

$$HW := \langle(\pi_i, s_i)_{i \in S}\rangle \quad \subset \quad \mathrm{End}(\mathbb{C}W)$$

- Any interesting structure?
- Contains all Hecke algebras by construction
- Type A: dimension and dimension of the radical in the Sloane!

The BiHecke algebra of rank 1

$$W := \{1, s\} \qquad \mathbb{C}W := \mathbb{C}.1 \oplus \mathbb{C}.s$$

$$\mathrm{id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad s = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \pi = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \quad \overline{\pi} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

Relations and some natural bases of HW

$$s\pi = \pi, \qquad \pi s = \overline{\pi}, \qquad \overline{\pi} + \pi = 1 + s$$

$$\{\mathrm{id}, s, \pi\} \qquad \text{or} \qquad \{\mathrm{id}, \pi, \overline{\pi}\}$$

Dimension 1 simple and projective modules

$$(1-s).\mathrm{id} = (1-s), \qquad (1-s).s = -(1-s), \qquad (1-s).\pi = 0$$

The BiHecke algebra of rank 1

$$W := \{1, s\} \qquad \mathbb{C}W := \mathbb{C}.1 \oplus \mathbb{C}.s$$

$$\mathrm{id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad s = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \pi = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \quad \overline{\pi} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

Relations and some natural bases of HW

$$s\pi = \pi, \qquad \pi s = \overline{\pi}, \qquad \overline{\pi} + \pi = 1 + s$$

$$\{\mathrm{id}, s, \pi\} \qquad \text{or} \qquad \{\mathrm{id}, \pi, \overline{\pi}\}$$

Dimension 1 simple and projective modules

$$(1-s).\mathrm{id} = (1-s), \qquad (1-s).s = -(1-s), \qquad (1-s).\pi = 0$$

The BiHecke algebra of rank 1

$$W := \{1, s\} \qquad \mathbb{C}W := \mathbb{C}.1 \oplus \mathbb{C}.s$$

$$\mathrm{id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad s = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \pi = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \quad \overline{\pi} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

Relations and some natural bases of HW

$$s\pi = \pi, \qquad \pi s = \overline{\pi}, \qquad \overline{\pi} + \pi = 1 + s$$

$$\{\mathrm{id}, s, \pi\} \qquad \text{or} \qquad \{\mathrm{id}, \pi, \overline{\pi}\}$$

Dimension 1 simple and projective modules

$$(1-s).\mathrm{id} = (1-s), \qquad (1-s).s = -(1-s), \qquad (1-s).\pi = 0$$

Structure of BiHecke algebras

Theorem (H.,T., 2005)

- HW *algebra of left antisymmetry preserving operators*
- HW^* *algebra of left symmetry preserving operators*
- *Basis of* HW*:* $\{w\pi_{w'} \mid \mathrm{D}_R(w) \cap \mathrm{D}_L(w') = \emptyset\}$
- *Rep. theory governed by the combinatorics of descents*
- HW *Morita equivalent to the poset algebra of boolean lattice*
- *Projective & simple modules indexed by parabolic subgroups*
  Restriction of simple:
  - *Exactly the Young's ribbon representation of* W
  - *Exactly the projective modules of* $\mathrm{H}(W)(0)$

Question (Thibon 2005)

Is there a link with the affine Hecke algebra?

Structure of BiHecke algebras

Theorem (H.,T., 2005)

- $\mathrm{H}W$ *algebra of left antisymmetry preserving operators*
- $\mathrm{H}W^*$ *algebra of left symmetry preserving operators*
- *Basis of* $\mathrm{H}W$*:* $\{w\pi_{w'} \mid \mathrm{D}_R(w) \cap \mathrm{D}_L(w') = \emptyset\}$
- *Rep. theory governed by the combinatorics of descents*
- $\mathrm{H}W$ *Morita equivalent to the poset algebra of boolean lattice*
- *Projective & simple modules indexed by parabolic subgroups*
  Restriction of simple:
  - *Exactly the Young's ribbon representation of* W
  - *Exactly the projective modules of* $\mathrm{H}(W)(0)$

Question (Thibon 2005)

Is there a link with the affine Hecke algebra?

Structure of BiHecke algebras

Theorem (H.,T., 2005)

- HW *algebra of left antisymmetry preserving operators*
- HW^* *algebra of left symmetry preserving operators*
- *Basis of* HW*:* $\{w\pi_{w'} \mid \mathrm{D}_R(w) \cap \mathrm{D}_L(w') = \emptyset\}$
- *Rep. theory governed by the combinatorics of descents*
- HW *Morita equivalent to the poset algebra of boolean lattice*
- *Projective & simple modules indexed by parabolic subgroups*
  Restriction of simple:
  - *Exactly the Young's ribbon representation of* W
  - *Exactly the projective modules of* $\mathrm{H}(W)(0)$

Question (Thibon 2005)

Is there a link with the affine Hecke algebra?

Structure of BiHecke algebras

Theorem (H.,T., 2005)

- HW *algebra of left antisymmetry preserving operators*
- HW^* *algebra of left symmetry preserving operators*
- *Basis of* HW*:* $\{w\pi_{w'} \mid \mathrm{D}_R(w) \cap \mathrm{D}_L(w') = \emptyset\}$
- *Rep. theory governed by the combinatorics of descents*
- HW *Morita equivalent to the poset algebra of boolean lattice*
- *Projective & simple modules indexed by parabolic subgroups*
  Restriction of simple:
    - *Exactly the Young's ribbon representation of* W
    - *Exactly the projective modules of* $\mathrm{H}(W)(0)$

Question (Thibon 2005)

Is there a link with the affine Hecke algebra?

Structure of BiHecke algebras

Theorem (H.,T., 2005)

- $\mathrm{H}W$ *algebra of left antisymmetry preserving operators*
- $\mathrm{H}W^*$ *algebra of left symmetry preserving operators*
- *Basis of* $\mathrm{H}W$*:* $\{w\pi_{w'} \mid \mathrm{D}_R(w) \cap \mathrm{D}_L(w') = \emptyset\}$
- *Rep. theory governed by the combinatorics of descents*
- $\mathrm{H}W$ *Morita equivalent to the poset algebra of boolean lattice*
- *Projective & simple modules indexed by parabolic subgroups*
  Restriction of simple:
  - *Exactly the Young's ribbon representation of* W
  - *Exactly the projective modules of* $\mathrm{H}(W)(0)$

Question (Thibon 2005)

Is there a link with the affine Hecke algebra?

Structure of BiHecke algebras

Theorem (H.,T., 2005)

- $\mathrm{H}W$ *algebra of left antisymmetry preserving operators*
- $\mathrm{H}W^*$ *algebra of left symmetry preserving operators*
- *Basis of* $\mathrm{H}W$*:* $\{w\pi_{w'} \mid \mathrm{D}_R(w) \cap \mathrm{D}_L(w') = \emptyset\}$
- *Rep. theory governed by the combinatorics of descents*
- $\mathrm{H}W$ *Morita equivalent to the poset algebra of boolean lattice*
- *Projective & simple modules indexed by parabolic subgroups*
  Restriction of simple:
  - *Exactly the Young's ribbon representation of* W
  - *Exactly the projective modules of* $\mathrm{H}(W)(0)$

Question (Thibon 2005)

Is there a link with the affine Hecke algebra?

Structure of BiHecke algebras

Theorem (H.,T., 2005)

- $\mathrm{H}W$ *algebra of left antisymmetry preserving operators*
- $\mathrm{H}W^*$ *algebra of left symmetry preserving operators*
- *Basis of* $\mathrm{H}W$*:* $\{w\pi_{w'} \mid \mathrm{D}_R(w) \cap \mathrm{D}_L(w') = \emptyset\}$
- *Rep. theory governed by the combinatorics of descents*
- $\mathrm{H}W$ *Morita equivalent to the poset algebra of boolean lattice*
- *Projective & simple modules indexed by parabolic subgroups*
  Restriction of simple:
  - *Exactly the Young's ribbon representation of* W
  - *Exactly the projective modules of* $\mathrm{H}(W)(0)$

Question (Thibon 2005)

Is there a link with the affine Hecke algebra?

Structure of BiHecke algebras

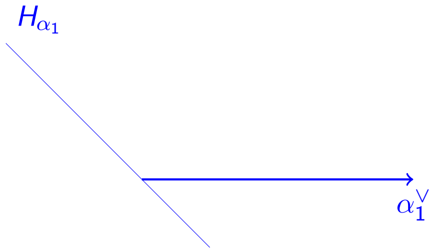
Theorem (H.,T., 2005)

- HW *algebra of left antisymmetry preserving operators*
- HW^* *algebra of left symmetry preserving operators*
- *Basis of* HW*:* $\{w\pi_{w'} \mid \mathrm{D}_R(w) \cap \mathrm{D}_L(w') = \emptyset\}$
- *Rep. theory governed by the combinatorics of descents*
- HW *Morita equivalent to the poset algebra of boolean lattice*
- *Projective & simple modules indexed by parabolic subgroups*
  Restriction of simple:
  - *Exactly the Young's ribbon representation of* W
  - *Exactly the projective modules of* H$(W)(0)$

Question (Thibon 2005)

Is there a link with the affine Hecke algebra?

Level 0 action of the type A_1^1 affine Hecke algebra



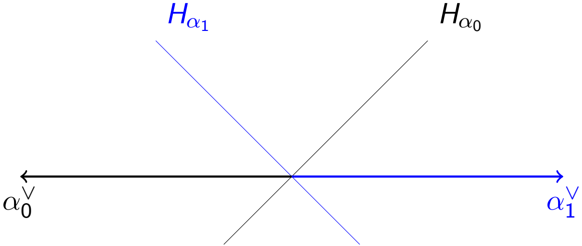
Remark (At level 0)

- W *degenerates trivially to* $\mathring{W}$ *of type* A_1; $W = \mathring{W} \ltimes T$

$$\mathrm{id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \pi_1 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad \pi_0 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

- π_0, π_1 *acts transitively on* $\mathring{W}$
- $\mathrm{H}(W)(0)$ *degenerates to* $\mathrm{H}\mathring{W}$, *not* $\mathrm{H}(\mathring{W})(0)$!

Level 0 action of the type A_1^1 affine Hecke algebra



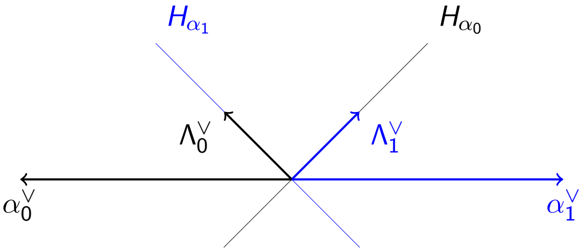
Remark (At level 0)

- *W degenerates trivially to $\mathring{W}$ of type A_1; $W = \mathring{W} \ltimes T$*

$$\mathrm{id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \pi_1 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad \pi_0 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

- *π_0, π_1 acts transitively on $\mathring{W}$*
- $\mathrm{H}(W)(0)$ *degenerates to* $\mathrm{H}\mathring{W}$, *not* $\mathrm{H}(\mathring{W})(0)$*!*

Level 0 action of the type A_1^1 affine Hecke algebra



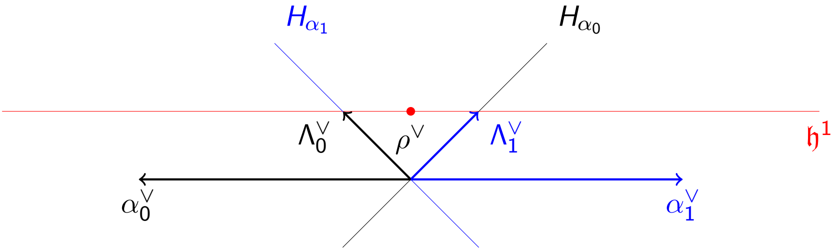
Remark (At level 0)

- *W degenerates trivially to $\mathring{W}$ of type A_1; $W = \mathring{W} \ltimes T$*

$$\mathrm{id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \pi_1 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad \pi_0 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

- *π_0, π_1 acts transitively on $\mathring{W}$*
- H(W)(0) *degenerates to* H$\mathring{W}$*, not* H($\mathring{W}$)(0)*!*

Level 0 action of the type A_1^1 affine Hecke algebra



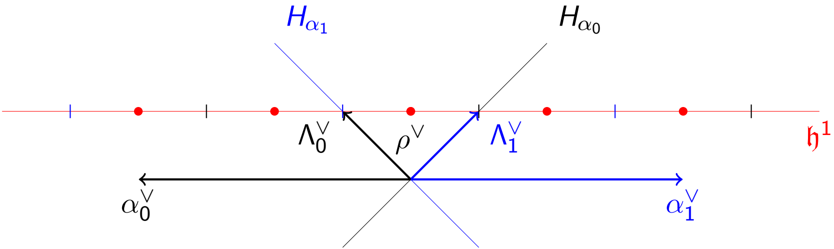
Remark (At level 0)

- *W degenerates trivially to $\mathring{W}$ of type A_1; $W = \mathring{W} \ltimes T$*

$$\mathrm{id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \pi_1 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad \pi_0 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

- *π_0, π_1 acts transitively on $\mathring{W}$*
- $\mathrm{H}(W)(0)$ *degenerates to* $\mathrm{H}\mathring{W}$*, not* $\mathrm{H}(\mathring{W})(0)$*!*

Level 0 action of the type A_1^1 affine Hecke algebra



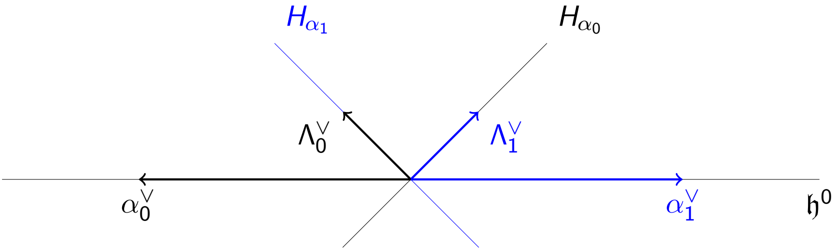
Remark (At level 0)

- *W degenerates trivially to $\mathring{W}$ of type A_1; $W = \mathring{W} \ltimes T$*

$$\mathrm{id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \pi_1 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad \pi_0 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

- *π_0, π_1 acts transitively on $\mathring{W}$*
- $\mathrm{H}(W)(0)$ *degenerates to* $\mathrm{H}\mathring{W}$*, not* $\mathrm{H}(\mathring{W})(0)$*!*

Level 0 action of the type A_1^1 affine Hecke algebra



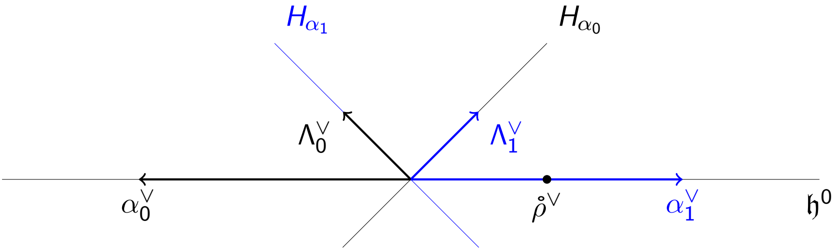
Remark (At level 0)

- *W degenerates trivially to $\mathring{W}$ of type A_1; $W = \mathring{W} \ltimes T$*

$$\mathrm{id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \pi_1 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad \pi_0 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

- *π_0, π_1 acts transitively on $\mathring{W}$*
- $\mathrm{H}(W)(0)$ *degenerates to* $\mathrm{H}\mathring{W}$*, not* $\mathrm{H}(\mathring{W})(0)$*!*

Level 0 action of the type A_1^1 affine Hecke algebra



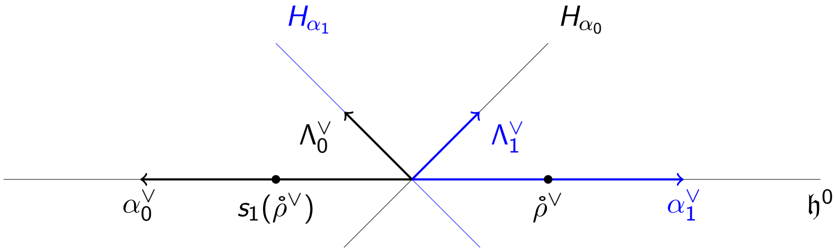
Remark (At level 0)

- *W degenerates trivially to $\mathring{W}$ of type A_1; $W = \mathring{W} \ltimes T$*

$$\mathrm{id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \pi_1 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad \pi_0 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

- *π_0, π_1 acts transitively on $\mathring{W}$*
- $\mathrm{H}(W)(0)$ *degenerates to* $\mathrm{H}\mathring{W}$, *not* $\mathrm{H}(\mathring{W})(0)$*!*

Level 0 action of the type A_1^1 affine Hecke algebra



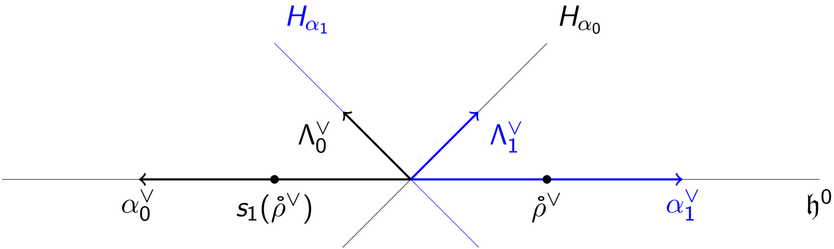
Remark (At level 0)

- *W degenerates trivially to $\mathring{W}$ of type A_1; $W = \mathring{W} \ltimes T$*

$$\mathrm{id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \pi_1 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad \pi_0 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

- *π_0, π_1 acts transitively on $\mathring{W}$*
- $\mathrm{H}(W)(0)$ *degenerates to* $\mathrm{H}\mathring{W}$, *not* $\mathrm{H}(\mathring{W})(0)$*!*

Level 0 action of the type A_1^1 affine Hecke algebra



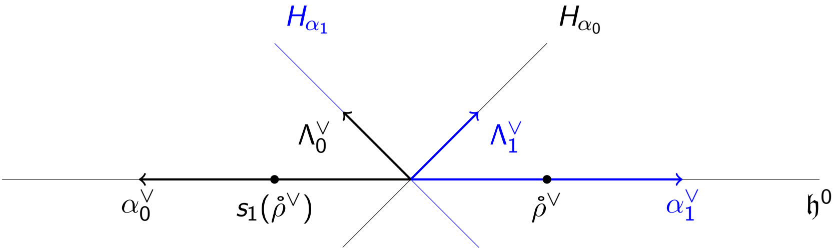
Remark (At level 0)

- *W degenerates trivially to $\mathring{W}$ of type A_1; $W = \mathring{W} \ltimes T$*

$$\mathrm{id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \pi_1 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad \pi_0 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

- *π_0, π_1 acts transitively on $\mathring{W}$*
- *$\mathrm{H}(W)(0)$ degenerates to $\mathrm{H}\mathring{W}$, not $\mathrm{H}(\mathring{W})(0)$!*

Level 0 action of the type A_1^1 affine Hecke algebra



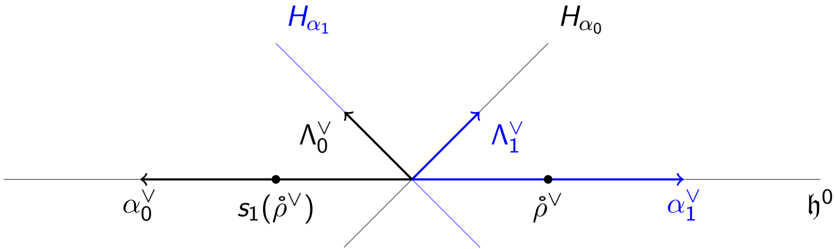
Remark (At level 0)

- *W degenerates trivially to $\mathring{W}$ of type A_1; $W = \mathring{W} \ltimes T$*

$$\mathrm{id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \pi_1 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad \pi_0 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

- *π_0, π_1 acts transitively on $\mathring{W}$*
- $\mathrm{H}(W)(0)$ *degenerates to* $\mathrm{H}\mathring{W}$, *not* $\mathrm{H}(\mathring{W})(0)$*!*

Level 0 action of the type A_1^1 affine Hecke algebra



Remark (At level 0)

- *W degenerates trivially to $\mathring{W}$ of type A_1; $W = \mathring{W} \ltimes T$*

$$\mathrm{id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \pi_1 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad \pi_0 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

- *π_0, π_1 acts transitively on $\mathring{W}$*
- *$\mathrm{H}(W)(0)$ degenerates to $\mathrm{H}\mathring{W}$, not $\mathrm{H}(\mathring{W})(0)$!*

Level 0 action of affine Hecke algebras

Theorem (H.,S.,T. 2008)

W: affine Weyl group
$\mathring{W}$: finite Weyl group induced by the level 0 action
At level 0:

- *W quotients trivially to $\mathring{W}$*
- $\mathrm{H}(W)(0)$ *quotients to* $\mathrm{H}\mathring{W}$
- $\mathrm{H}(W)(q)$ *quotients to* $\mathrm{H}\mathring{W}$, *for q generic*
- $\pi_0, \pi_1, \ldots, \pi_n$ *act transitively on* $\mathring{W}$

Level 0 action of affine Hecke algebras

Theorem (H.,S.,T. 2008)

W: affine Weyl group
$\mathring{W}$: finite Weyl group induced by the level 0 action
At level 0:

- *W quotients trivially to $\mathring{W}$*
- *$\mathrm{H}(W)(0)$ quotients to $\mathrm{H}\mathring{W}$*
- *$\mathrm{H}(W)(q)$ quotients to $\mathrm{H}\mathring{W}$, for q generic*
- *$\pi_0, \pi_1, \dots, \pi_n$ act transitively on $\mathring{W}$*

Level 0 action of affine Hecke algebras

Theorem (H.,S.,T. 2008)

W: affine Weyl group
$\mathring{W}$: finite Weyl group induced by the level 0 action
At level 0:

- *W quotients trivially to $\mathring{W}$*
- *$\mathrm{H}(W)(0)$ quotients to $\mathrm{H}\mathring{W}$*
- *$\mathrm{H}(W)(q)$ quotients to $\mathrm{H}\mathring{W}$, for q generic*
- *$\pi_0, \pi_1, \ldots, \pi_n$ act transitively on $\mathring{W}$*

Level 0 action of affine Hecke algebras

Theorem (H.,S.,T. 2008)

W: affine Weyl group
$\mathring{W}$: finite Weyl group induced by the level 0 action
At level 0:

- *W quotients trivially to $\mathring{W}$*
- $\mathrm{H}(W)(0)$ *quotients to* $\mathrm{H}\mathring{W}$
- $\mathrm{H}(W)(q)$ *quotients to* $\mathrm{H}\mathring{W}$*, for q generic*
- $\pi_0, \pi_1, \ldots, \pi_n$ *act transitively on* $\mathring{W}$

Type A: Bubble (anti) sort algorithm

1234

Type A: Bubble (anti) sort algorithm

1234

Type A: Bubble (anti) sort algorithm

1243

Type A: Bubble (anti) sort algorithm

1423

Type A: Bubble (anti) sort algorithm

4123

Type A: Bubble (anti) sort algorithm

4123

Type A: Bubble (anti) sort algorithm

4132

Type A: Bubble (anti) sort algorithm

4312

Type A: Bubble (anti) sort algorithm

4312

Type A: Bubble (anti) sort algorithm

4321

Type A: Bubble (anti) sort algorithm

4321

Type A: Bubble (anti) sort algorithm

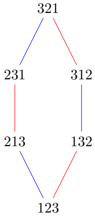
4321

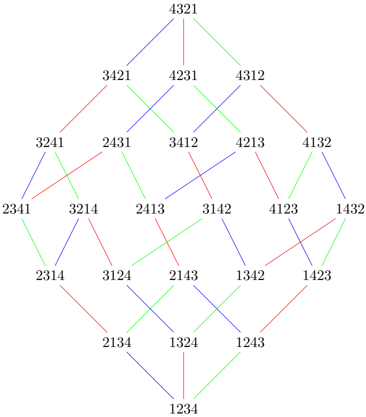
Underlying combinatorics: right permutohedron

Type A: Bubble (anti) sort algorithm

4321

Underlying combinatorics: right permutohedron

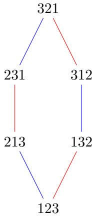


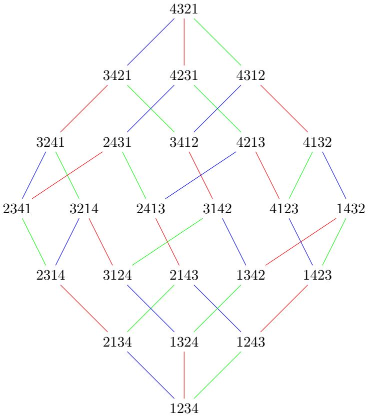


Type A: Bubble (anti) sort algorithm

4321

Underlying combinatorics: right permutohedron



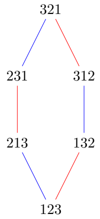


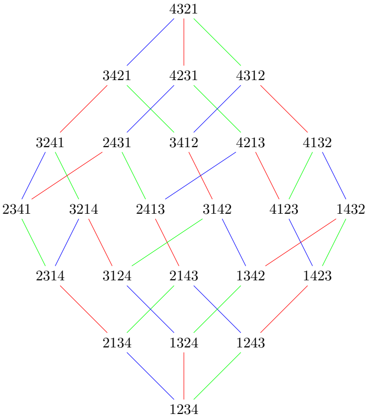
Elementary transpositions: $s_1, s_2, s_3, \ldots$

Type A: Bubble (anti) sort algorithm

4321

Underlying combinatorics: right permutohedron



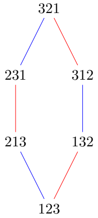


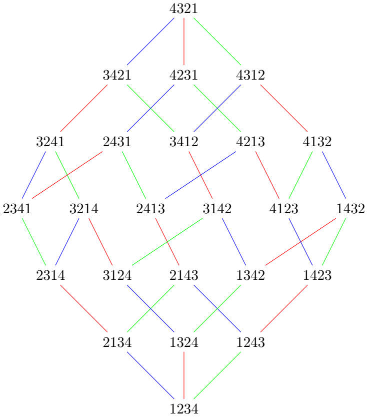
Elementary transpositions: $s_1, s_2, s_3, \ldots$

Elementary bubble antisort operators: $\pi_1, \pi_2, \pi_3, \ldots$

Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!





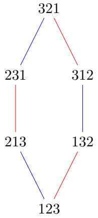
Proposition (HT'08)

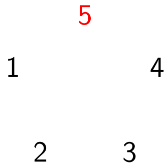
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

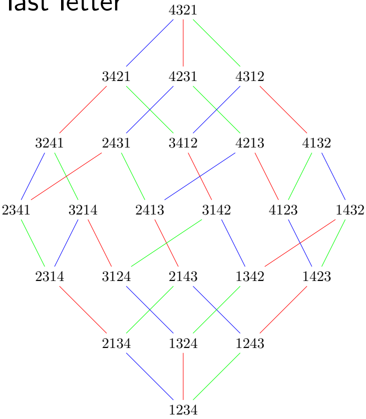
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







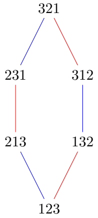
Proposition (HT'08)

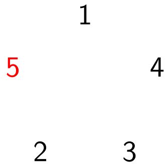
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

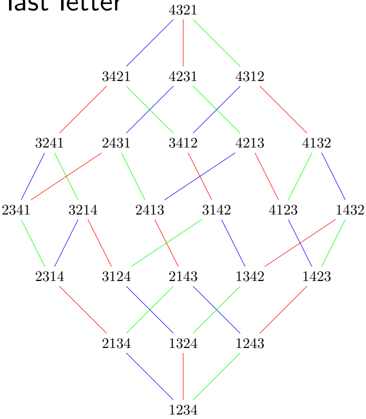
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







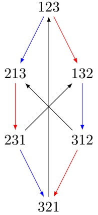
Proposition (HT'08)

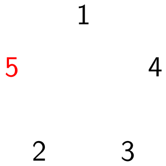
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

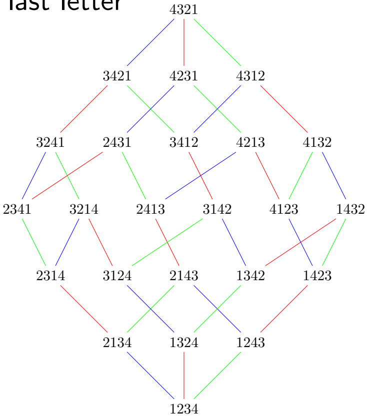
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







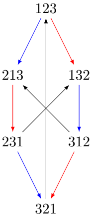
Proposition (HT'08)

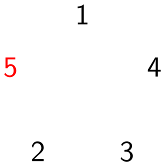
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

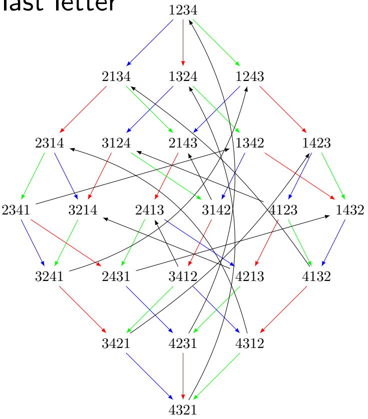
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







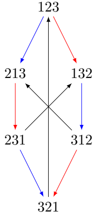
Proposition (HT'08)

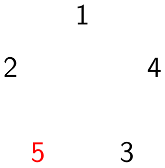
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

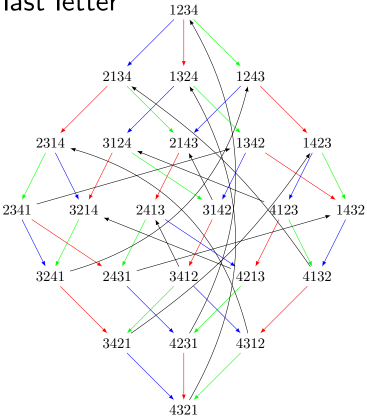
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







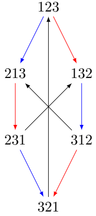
Proposition (HT'08)

$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

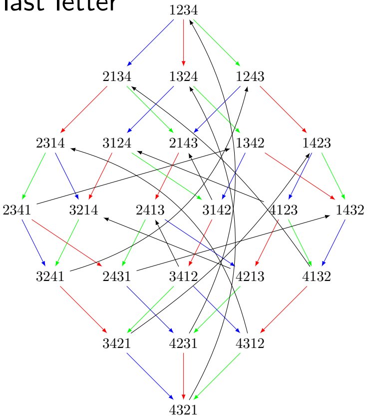
New operator π_0: acts between first and last letter



1

2 4

3 5



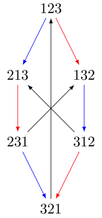
Proposition (HT'08)

$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

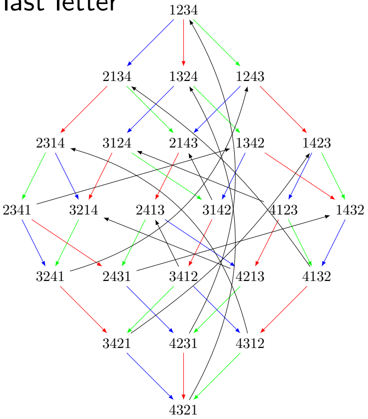
New operator π_0: acts between first and last letter



1

2 5

3 4



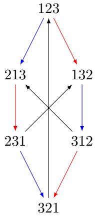
Proposition (HT'08)

$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

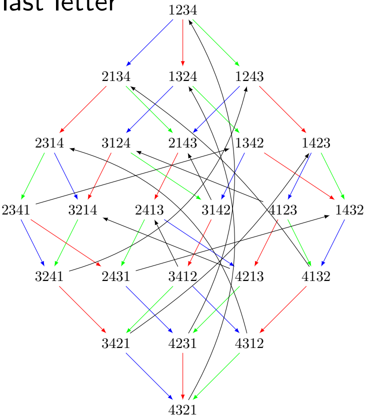
New operator π_0: acts between first and last letter



5

2 1

3 4



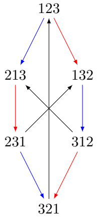
Proposition (HT'08)

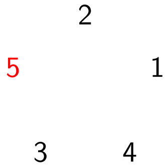
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

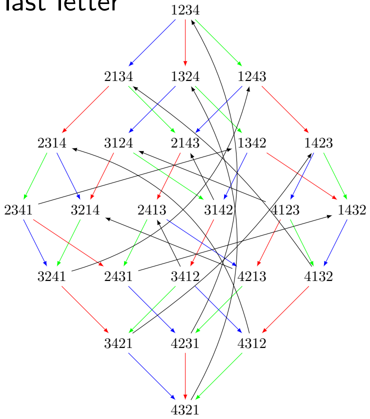
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







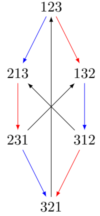
Proposition (HT'08)

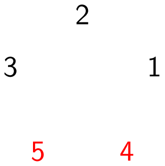
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

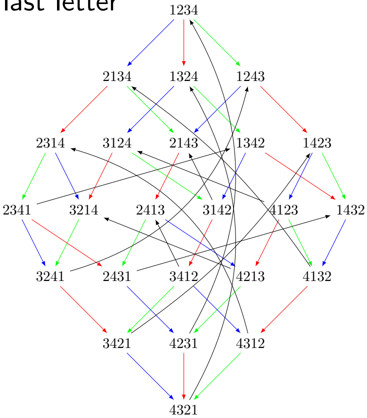
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







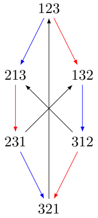
Proposition (HT'08)

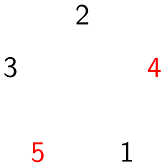
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

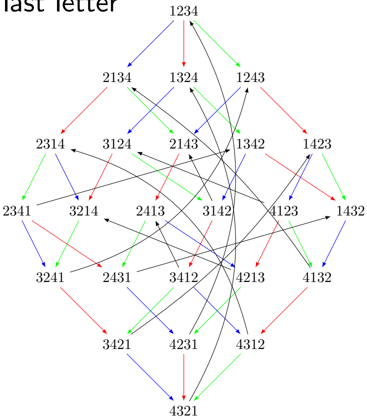
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







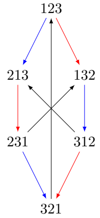
Proposition (HT'08)

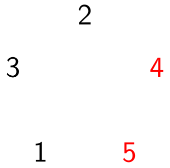
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

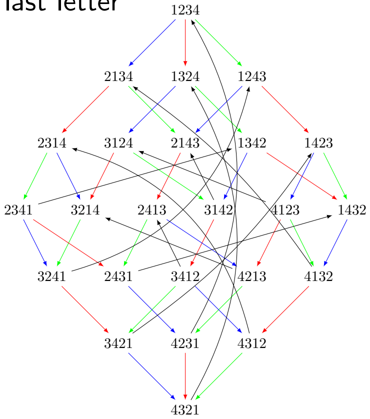
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







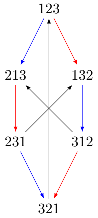
Proposition (HT'08)

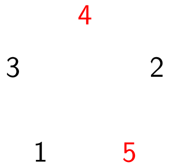
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

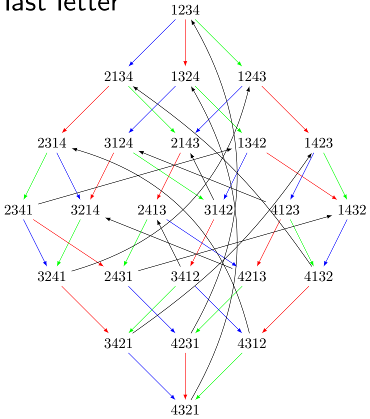
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







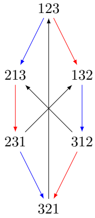
Proposition (HT'08)

$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

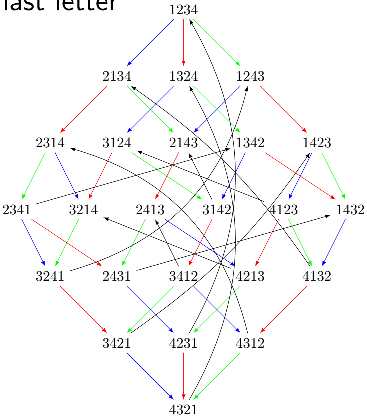
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter



4
3 5
1 2



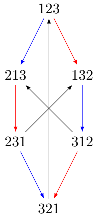
Proposition (HT'08)

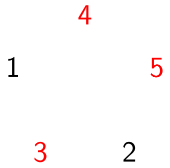
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

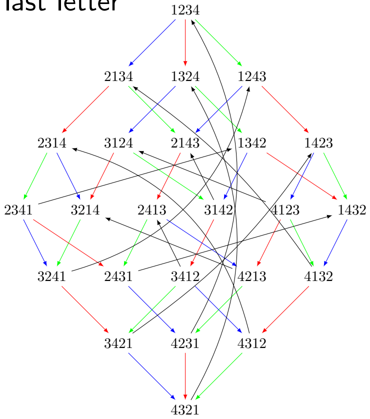
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







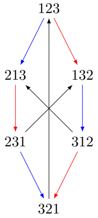
Proposition (HT'08)

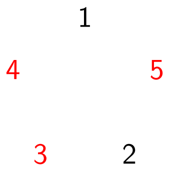
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

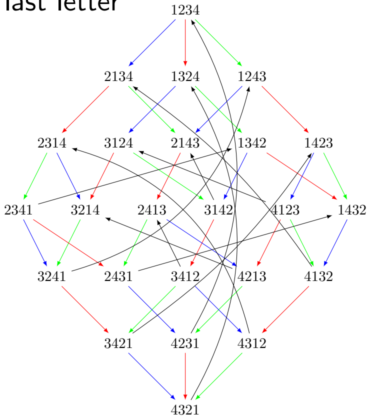
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







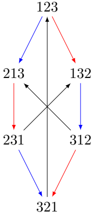
Proposition (HT'08)

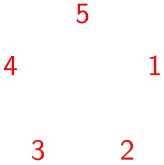
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

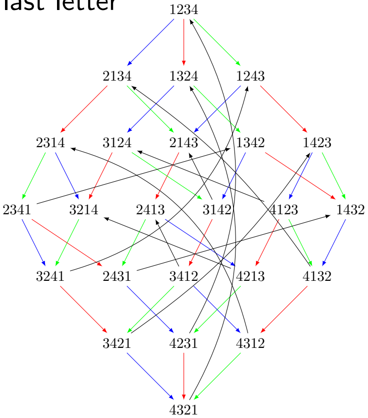
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







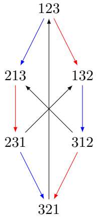
Proposition (HT'08)

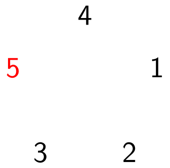
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

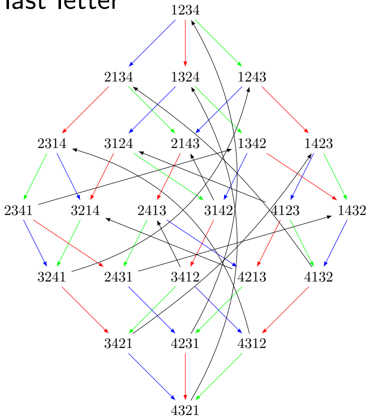
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







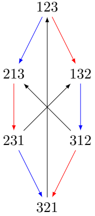
Proposition (HT'08)

$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

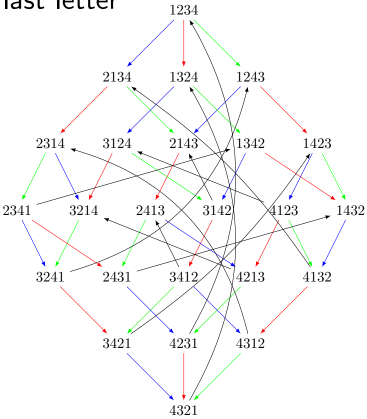
New operator π_0: acts between first and last letter



4

3 1

5 2



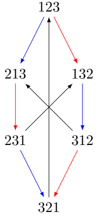
Proposition (HT'08)

$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

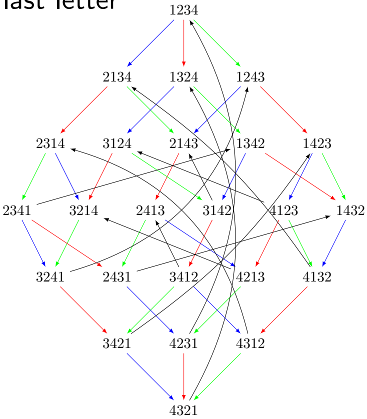
New operator π_0: acts between first and last letter



4

3 1

2 5



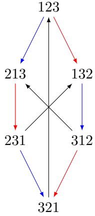
Proposition (HT'08)

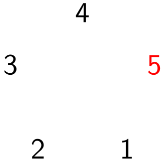
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

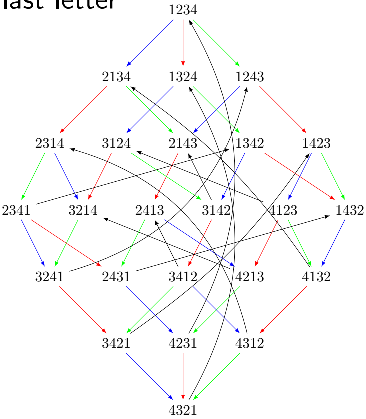
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







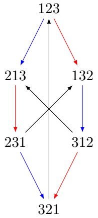
Proposition (HT'08)

$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

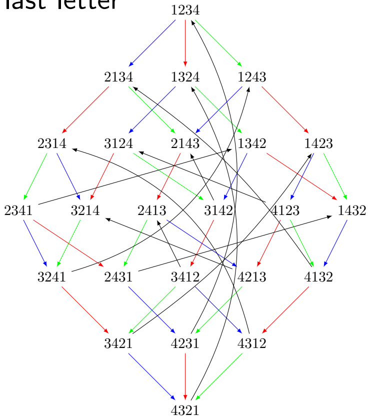
New operator π_0: acts between first and last letter



5

3 4

2 1



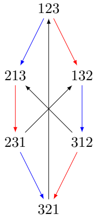
Proposition (HT'08)

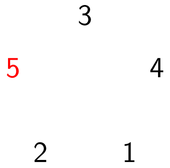
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

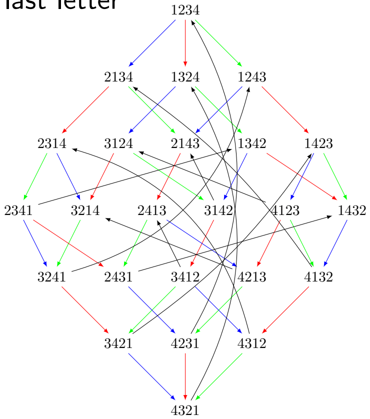
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







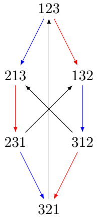
Proposition (HT'08)

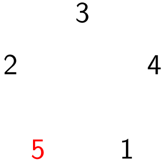
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

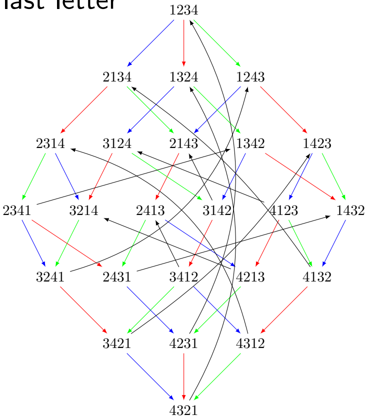
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







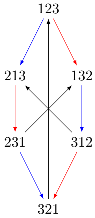
Proposition (HT'08)

$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

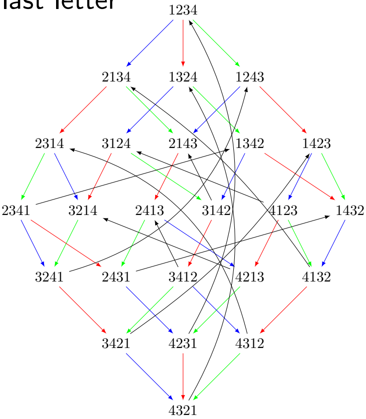
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321   There is no return!

New operator π_0: acts between first and last letter



3
2 4
1 5



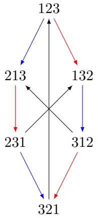
Proposition (HT'08)

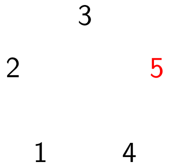
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

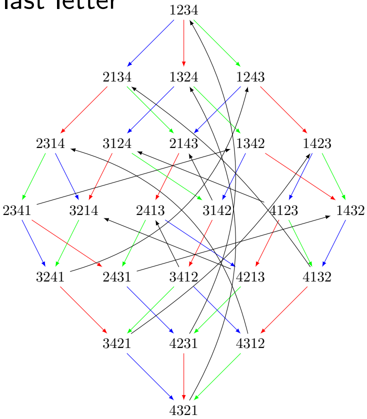
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







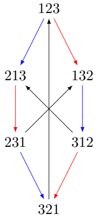
Proposition (HT'08)

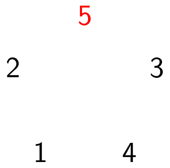
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

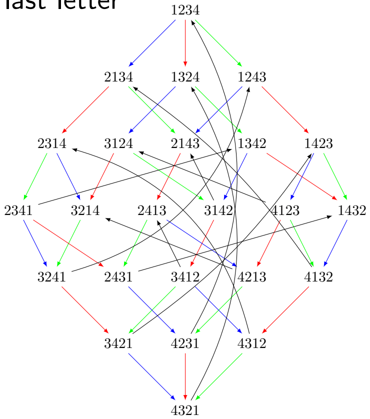
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







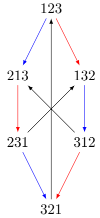
Proposition (HT'08)

$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

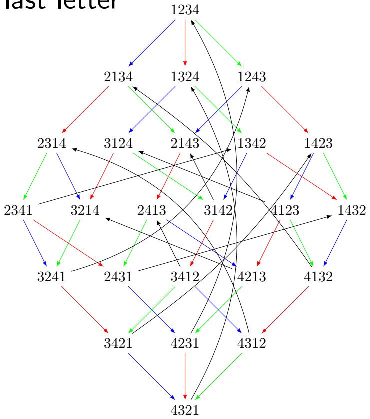
New operator π_0: acts between first and last letter



2

5 3

1 4



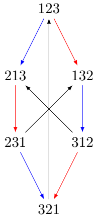
Proposition (HT'08)

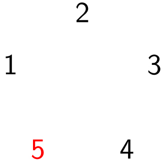
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

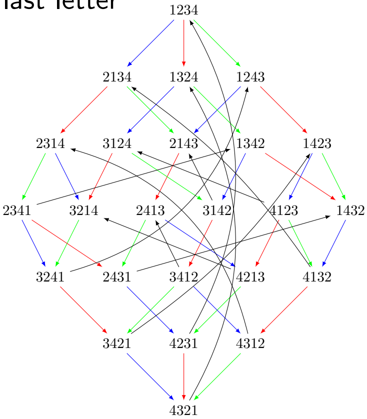
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







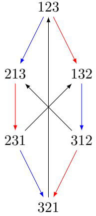
Proposition (HT'08)

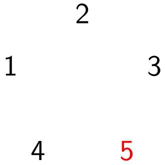
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

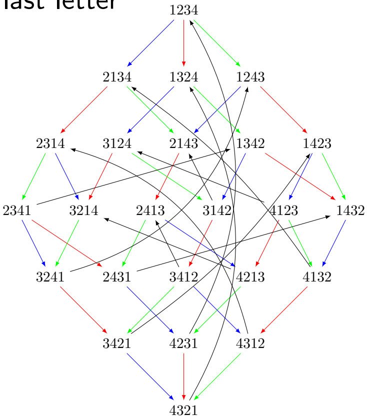
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







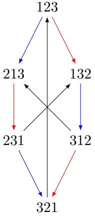
Proposition (HT'08)

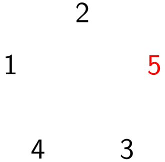
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

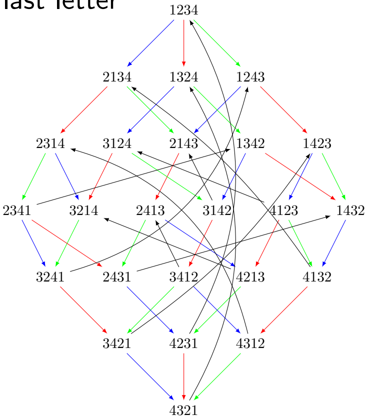
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







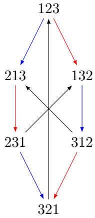
Proposition (HT'08)

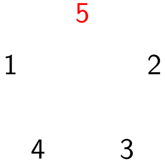
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

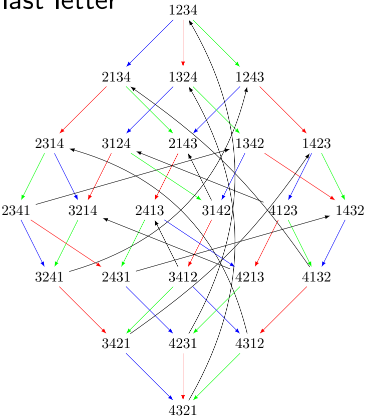
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







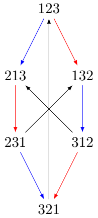
Proposition (HT'08)

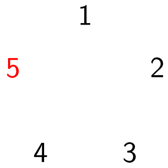
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

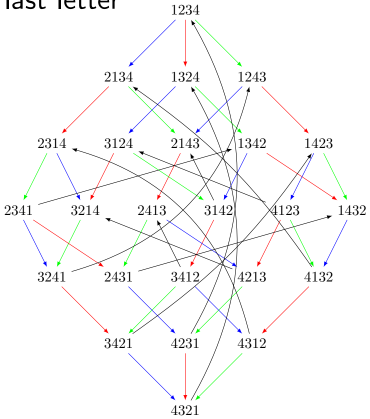
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







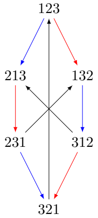
Proposition (HT'08)

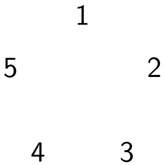
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

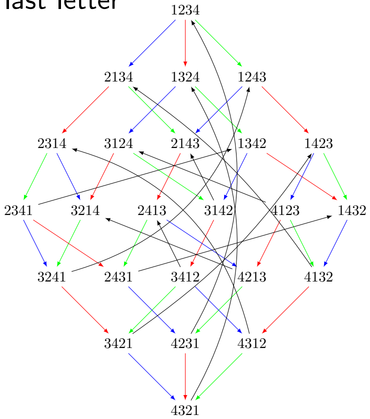
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







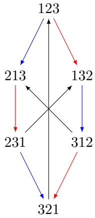
Proposition (HT'08)

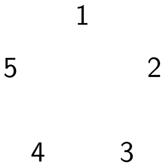
$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ act transitively on permutations

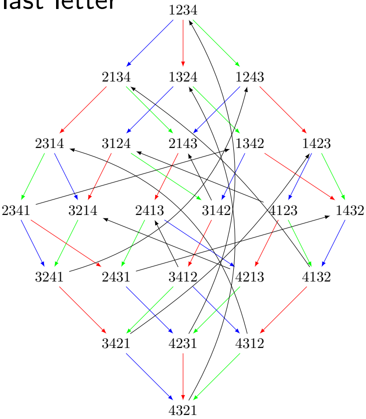
Type A: Cyclic bubble sort

$\pi_1, \pi_2, \pi_3, \ldots$ antisort 12345 into 54321 There is no return!

New operator π_0: acts between first and last letter







Proposition (HT'08)

$\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ *act transitively on permutations*

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types* B, C, D
- *Existence for all types (including twisted)*

Proof

- Type B:

$$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$$

$$\underline{1234}$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{12}\underline{34}$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$21\underline{34}$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

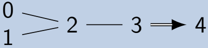
Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:



$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$21\underline{34}$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$2\underline{3}1\underline{4}$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$2\underline{34}1$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$2\underline{34}1$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{3}2\underline{4}1$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{34}21$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{34}21$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

4321

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

4321

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$432\underline{1}$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$43\underline{1}2$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$4\underline{1}32$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{1}432$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{1}43$2

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

 $1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$$\underline{1}43\underline{2}$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{1}4\underline{2}3$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:  $1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$$\underline{1}\underline{2}43$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$$

$$\underline{12}43$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$$\underline{1}\underline{2}4\underline{3}$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: $\begin{smallmatrix}0\\1\end{smallmatrix} > 2 - 3 \Rightarrow 4$ $\quad 1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$$\underline{1}\underline{2}\underline{3}4$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{123}4$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$21\underline{3}4$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$21\underline{3}4$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$2\underline{3}14$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$2\underline{3}14$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{3}214$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{3}214$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{3}241$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{3}24\underline{1}$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{3}2\underline{1}4$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{3}\underline{1}24$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

  $1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

  $\underline{31}24$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

1324

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

1324

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

1342

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$134\underline{2}$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$13\underline{2}4$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$1\underline{2}34$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{2}134$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{2}134$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{2}314$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{2}341$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{2}34\underline{1}$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

$\underline{2}3\underline{1}4$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$$

$$\underline{2}\underline{1}34$$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

  $1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

  $\underline{21}34$

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

1234

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

1234

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

- Type B:

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

1234

- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Recursive sorting algorithms in other types

Proposition (S.,T.,2007)

- *Similar algorithms for types B, C, D*
- *Existence for all types (including twisted)*

Proof

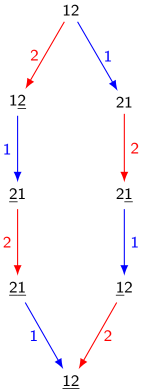
- Type B: 

$1 < 2 < 3 < 4 < \underline{4} < \underline{3} < \underline{2} < \underline{1}$

1234

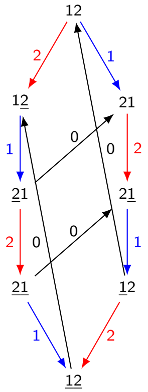
- Type-free induction strategy
- Case by case induction step
  Brute force on computer for the exceptional types (E_8!)

Type free geometric argument (I)



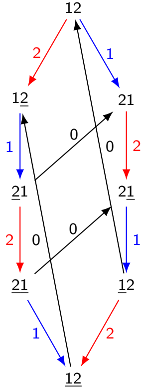
π_1, π_2 on C_2

Type free geometric argument (I)

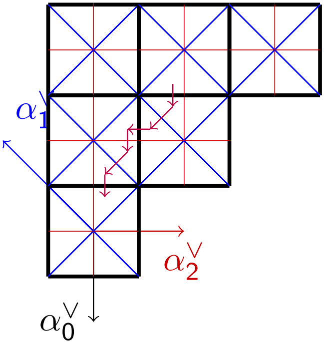


π_0, π_1, π_2 on C_2

Type free geometric argument (I)

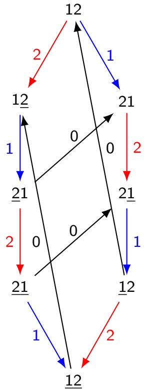


π_0, π_1, π_2 on C_2

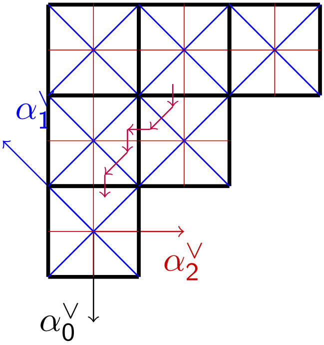


Alcove picture at level 1

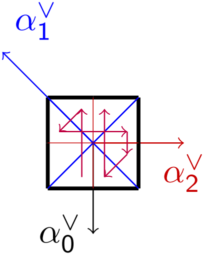
Type free geometric argument (I)



π_0, π_1, π_2 on C_2

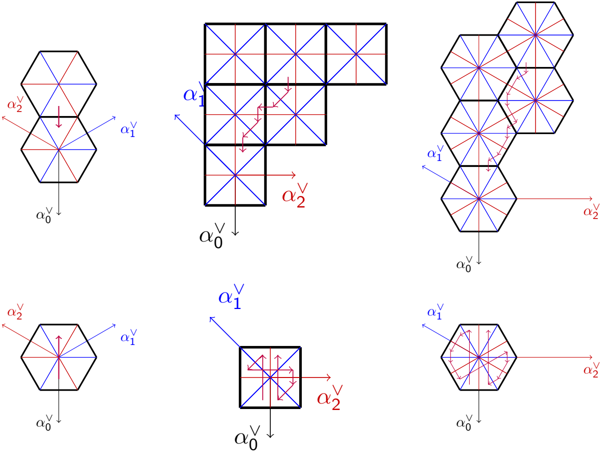


Alcove picture at level 1



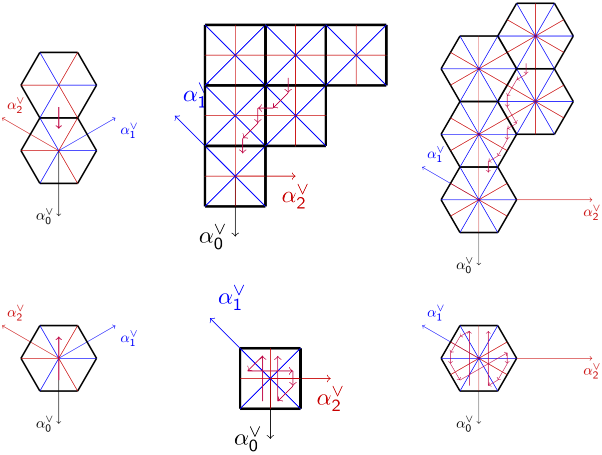
Quotient at level 0
(Steinberg torus)

Type free geometric argument (II)



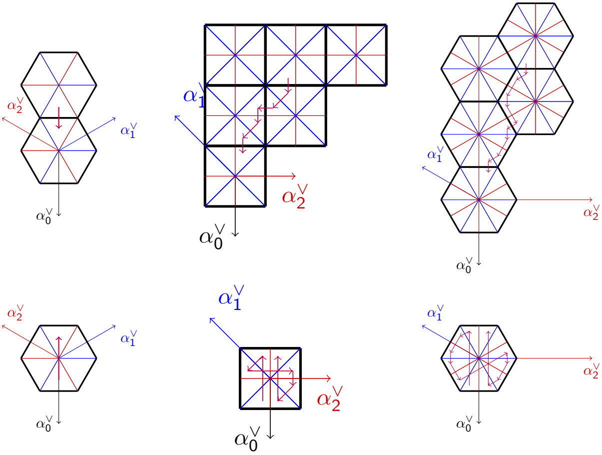
- Covers all rank 2 affine Weyl groups
- Generalization to I_n? Meaning of π_0?

Type free geometric argument (II)



- Covers all rank 2 affine Weyl groups
- Generalization to I_n? Meaning of π_0?

Type free geometric argument (II)



- Covers all rank 2 affine Weyl groups
- Generalization to I_n? Meaning of π_0?

BiHecke algebras and principal series representations

- $Y^{\lambda^\vee} \in \mathrm{H}(W)(q)$ (analog of translations in W)
- $\mathbb{C}[Y]$: commutative algebra, $\mathbb{C}[Y]^{\mathring{W}}$: center of $\mathrm{H}(W)(q)$
- t: character on $\mathbb{C}[Y]$
  - Central specialization of $\mathbb{C}[Y]^W$ on t:
    Quotient $\mathcal{H}(q,t)$ of $\mathrm{H}(W)(q)$ of dimension $|W|^2$
  - Representation $\rho_t := t\uparrow_{\mathbb{C}[Y]}^{\mathrm{H}(W)(q)}$ of $\mathrm{H}(W)(q)$ on $\mathbb{C}\mathring{W}$:
    Quotient of $\mathcal{H}(q,t)$, generically trivial

Theorem (S.,T., 2008)

Take q non zero and non root of unity, and $t : Y^{\lambda^\vee} \mapsto q^{-\mathrm{ht}(\lambda^\vee)}$
Then, $\rho_t(\mathrm{H}(W)(q)) = \mathrm{H}\mathring{W}$
I.e. $\mathrm{H}\mathring{W}$ (non trivial!) quotient of $\mathcal{H}(q,t)$ and of $\mathrm{H}(W)(q)$

- Proof: diagonalization of the action of Y on $\mathbb{C}\mathring{W}$, using alcove walks and the intertwining operators τ_i
- What happens at roots of unity? (Nicolas Borie)

BiHecke algebras and principal series representations

- $Y^{\lambda^\vee} \in \mathrm{H}(W)(q)$ (analog of translations in W)
- $\mathbb{C}[Y]$: commutative algebra, $\mathbb{C}[Y]^{\mathring{W}}$: center of $\mathrm{H}(W)(q)$
- t: character on $\mathbb{C}[Y]$
  - Central specialization of $\mathbb{C}[Y]^{W}$ on t:
    Quotient $\mathcal{H}(q,t)$ of $\mathrm{H}(W)(q)$ of dimension $|W|^2$
  - Representation $\rho_t := t \uparrow_{\mathbb{C}[Y]}^{\mathrm{H}(W)(q)}$ of $\mathrm{H}(W)(q)$ on $\mathbb{C}\mathring{W}$:
    Quotient of $\mathcal{H}(q,t)$, generically trivial

Theorem (S.,T., 2008)

Take q non zero and non root of unity, and $t : Y^{\lambda^\vee} \mapsto q^{-\operatorname{ht}(\lambda^\vee)}$
Then, $\rho_t(\mathrm{H}(W)(q)) = \mathrm{H}\mathring{W}$
I.e. $\mathrm{H}\mathring{W}$ (non trivial!) quotient of $\mathcal{H}(q,t)$ and of $\mathrm{H}(W)(q)$

- Proof: diagonalization of the action of Y on $\mathbb{C}\mathring{W}$, using alcove walks and the intertwining operators τ_i
- What happens at roots of unity? (Nicolas Borie)

BiHecke algebras and principal series representations

- $Y^{\lambda^\vee} \in \mathrm{H}(W)(q)$ (analog of translations in W)
- $\mathbb{C}[Y]$: commutative algebra, $\mathbb{C}[Y]^{\mathring{W}}$: center of $\mathrm{H}(W)(q)$
- t: character on $\mathbb{C}[Y]$
  - Central specialization of $\mathbb{C}[Y]^{W}$ on t:
    Quotient $\mathcal{H}(q,t)$ of $\mathrm{H}(W)(q)$ of dimension $|W|^2$
  - Representation $\rho_t := t \uparrow_{\mathbb{C}[Y]}^{\mathrm{H}(W)(q)}$ of $\mathrm{H}(W)(q)$ on $\mathbb{C}\mathring{W}$:
    Quotient of $\mathcal{H}(q,t)$, generically trivial

Theorem (S.,T., 2008)

Take q non zero and non root of unity, and $t : Y^{\lambda^\vee} \mapsto q^{-\operatorname{ht}(\lambda^\vee)}$
Then, $\rho_t(\mathrm{H}(W)(q)) = \mathrm{H}\mathring{W}$
I.e. $\mathrm{H}\mathring{W}$ (non trivial!) quotient of $\mathcal{H}(q,t)$ and of $\mathrm{H}(W)(q)$

- Proof: diagonalization of the action of Y on $\mathbb{C}\mathring{W}$, using alcove walks and the intertwining operators τ_i
- What happens at roots of unity? (Nicolas Borie)

BiHecke algebras and principal series representations

- $Y^{\lambda^\vee} \in \mathrm{H}(W)(q)$ (analog of translations in W)
- $\mathbb{C}[Y]$: commutative algebra, $\mathbb{C}[Y]^{\mathring{W}}$: center of $\mathrm{H}(W)(q)$
- t: character on $\mathbb{C}[Y]$
  - Central specialization of $\mathbb{C}[Y]^W$ on t:
    Quotient $\mathcal{H}(q,t)$ of $\mathrm{H}(W)(q)$ of dimension $|W|^2$
  - Representation $\rho_t := t \uparrow_{\mathbb{C}[Y]}^{\mathrm{H}(W)(q)}$ of $\mathrm{H}(W)(q)$ on $\mathbb{C}\mathring{W}$:
    Quotient of $\mathcal{H}(q,t)$, generically trivial

Theorem (S.,T., 2008)

Take q non zero and non root of unity, and $t : Y^{\lambda^\vee} \mapsto q^{-\operatorname{ht}(\lambda^\vee)}$
Then, $\rho_t(\mathrm{H}(W)(q)) = \mathrm{H}\mathring{W}$
I.e. $\mathrm{H}\mathring{W}$ (non trivial!) quotient of $\mathcal{H}(q,t)$ and of $\mathrm{H}(W)(q)$

- Proof: diagonalization of the action of Y on $\mathbb{C}\mathring{W}$, using alcove walks and the intertwining operators τ_i
- What happens at roots of unity? (Nicolas Borie)

BiHecke algebras and principal series representations

- $Y^{\lambda^\vee} \in \mathrm{H}(W)(q)$ (analog of translations in W)
- $\mathbb{C}[Y]$: commutative algebra, $\mathbb{C}[Y]^{\mathring{W}}$: center of $\mathrm{H}(W)(q)$
- t: character on $\mathbb{C}[Y]$
  - Central specialization of $\mathbb{C}[Y]^W$ on t:
    Quotient $\mathcal{H}(q,t)$ of $\mathrm{H}(W)(q)$ of dimension $|W|^2$
  - Representation $\rho_t := t \uparrow_{\mathbb{C}[Y]}^{\mathrm{H}(W)(q)}$ of $\mathrm{H}(W)(q)$ on $\mathbb{C}\mathring{W}$:
    Quotient of $\mathcal{H}(q,t)$, generically trivial

Theorem (S.,T., 2008)

Take q non zero and non root of unity, and $t : Y^{\lambda^\vee} \mapsto q^{-\operatorname{ht}(\lambda^\vee)}$
Then, $\rho_t(\mathrm{H}(W)(q)) = \mathrm{H}\mathring{W}$
I.e. $\mathrm{H}\mathring{W}$ (non trivial!) quotient of $\mathcal{H}(q,t)$ and of $\mathrm{H}(W)(q)$

- Proof: diagonalization of the action of Y on $\mathbb{C}\mathring{W}$, using alcove walks and the intertwining operators τ_i
- What happens at roots of unity? (Nicolas Borie)

BiHecke algebras and principal series representations

- $Y^{\lambda^\vee} \in \mathrm{H}(W)(q)$ (analog of translations in W)
- $\mathbb{C}[Y]$: commutative algebra, $\mathbb{C}[Y]^{\mathring{W}}$: center of $\mathrm{H}(W)(q)$
- t: character on $\mathbb{C}[Y]$
  - Central specialization of $\mathbb{C}[Y]^W$ on t:
    Quotient $\mathcal{H}(q,t)$ of $\mathrm{H}(W)(q)$ of dimension $|W|^2$
  - Representation $\rho_t := t \uparrow_{\mathbb{C}[Y]}^{\mathrm{H}(W)(q)}$ of $\mathrm{H}(W)(q)$ on $\mathbb{C}\mathring{W}$:
    Quotient of $\mathcal{H}(q,t)$, generically trivial

Theorem (S.,T., 2008)

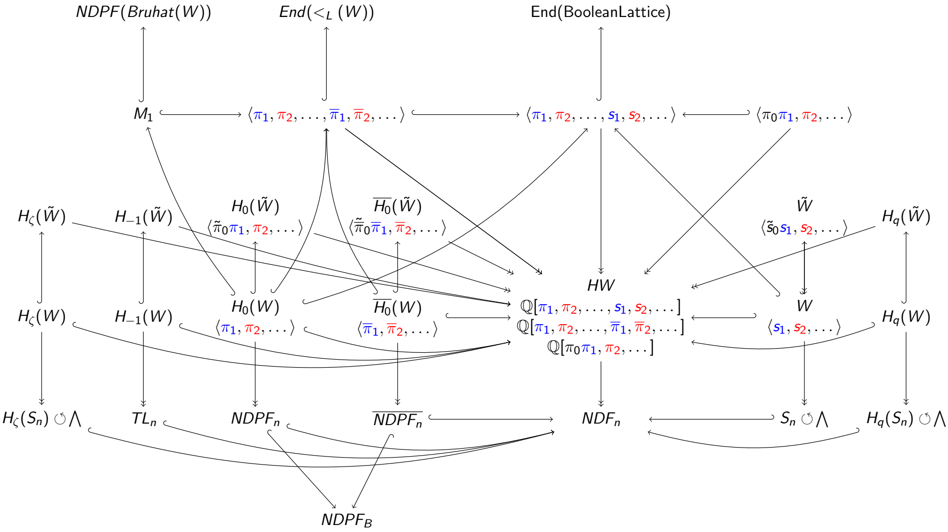
Take q non zero and non root of unity, and $t : Y^{\lambda^\vee} \mapsto q^{-\mathrm{ht}(\lambda^\vee)}$
Then, $\rho_t(\mathrm{H}(W)(q)) = \mathrm{H}\mathring{W}$
I.e. $\mathrm{H}\mathring{W}$ (non trivial!) quotient of $\mathcal{H}(q,t)$ and of $\mathrm{H}(W)(q)$

- Proof: diagonalization of the action of Y on $\mathbb{C}\mathring{W}$, using alcove walks and the intertwining operators τ_i
- What happens at roots of unity? (Nicolas Borie)

Partial conclusion

- BiHecke algebras:
  - Many equivalent definitions
  - Nice structure and representation theory
  - (type A) Connections with NCSF, parking functions
  - Connections with 0-Hecke and affine Hecke algebras
- Where does this structure come from?
- Is it useful?

The Big Picture



The bi-Hecke monoid

Question

Size of $M(W) = \langle \pi_1, \pi_2, \dots, \overline{\pi}_1, \overline{\pi}_2, \dots \rangle$

$|M(S_n)| = 1, 3, 23, 477, 31103, ?$

- How to attack such a problem?
- Generators and relations?
- Representation theory?

Theorem (HST08)

$M(W)$ admits $|W|$ simple / indecomposable projective modules

- Why do we care?

$$|M(W)| = \sum_{w \in W} \dim S_w . \dim P_w$$

The bi-Hecke monoid

Question

Size of $M(W) = \langle \pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$
$|M(S_n)| = 1, 3, 23, 477, 31103, ?$

- How to attack such a problem?
- Generators and relations?
- Representation theory?

Theorem (HST08)

$M(W)$ admits $|W|$ simple / indecomposable projective modules

- Why do we care?

$$|M(W)| = \sum_{w \in W} \dim S_w . \dim P_w$$

The bi-Hecke monoid

Question

Size of $M(W) = \langle \pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$
$|M(S_n)| = 1, 3, 23, 477, 31103, ?$

- How to attack such a problem?
- Generators and relations?
- Representation theory?

Theorem (HST08)

$M(W)$ admits $|W|$ simple / indecomposable projective modules

- Why do we care?

$$|M(W)| = \sum_{w \in W} \dim S_w . \dim P_w$$

The bi-Hecke monoid

Question

Size of $M(W) = \langle \pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$
$|M(S_n)| = 1, 3, 23, 477, 31103, ?$

- How to attack such a problem?
- Generators and relations?
- Representation theory?

Theorem (HST08)

$M(W)$ admits $|W|$ simple / indecomposable projective modules

- Why do we care?

$$|M(W)| = \sum_{w \in W} \dim S_w . \dim P_w$$

The bi-Hecke monoid

Question

Size of $M(W) = \langle \pi_1, \pi_2, \dots, \overline{\pi}_1, \overline{\pi}_2, \dots \rangle$
$|M(S_n)| = 1, 3, 23, 477, 31103, ?$

- How to attack such a problem?
- Generators and relations?
- Representation theory?

Theorem (HST08)

$M(W)$ admits $|W|$ simple / indecomposable projective modules

- Why do we care?

$$|M(W)| = \sum_{w \in W} \dim S_w . \dim P_w$$

The bi-Hecke monoid

Question

Size of $M(W) = \langle \pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$
$|M(S_n)| = 1, 3, 23, 477, 31103, ?$

- How to attack such a problem?
- Generators and relations?
- Representation theory?

Theorem (HST08)

$M(W)$ admits $|W|$ simple / indecomposable projective modules

- Why do we care?

$$|M(W)| = \sum_{w \in W} \dim S_w . \dim P_w$$

The bi-Hecke monoid

Question

Size of $M(W) = \langle \pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots \rangle$
$|M(S_n)| = 1, 3, 23, 477, 31103, ?$

- How to attack such a problem?
- Generators and relations?
- Representation theory?

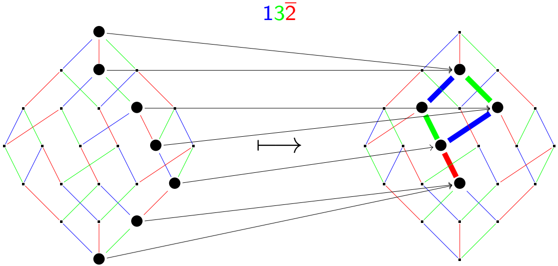
Theorem (HST08)

$M(W)$ admits $|W|$ simple / indecomposable projective modules

- Why do we care?

$$|M(W)| = \sum_{w \in W} \dim S_w . \dim P_w$$

Key combinatorial lemma



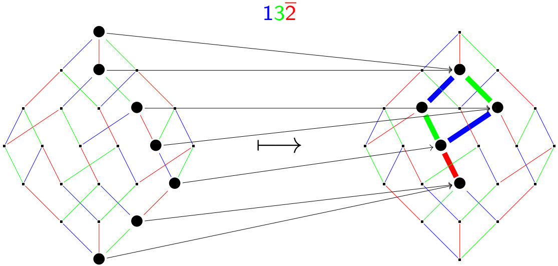
Lemma

For $f \in M(W)$ and $w \in W$: $(s_i w).f = w.f$ or $s_i\,(w.f)$

Proof.

Exchange property / associativity □

Key combinatorial lemma



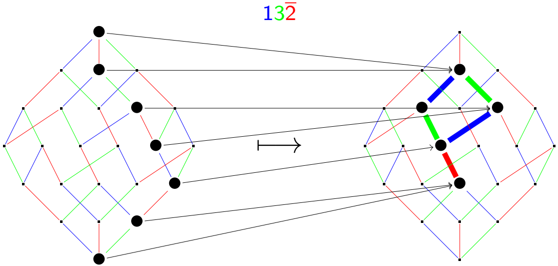
Lemma

For $f \in M(W)$ and $w \in W$: $\quad (s_i w).f = w.f \quad$ *or* $\quad s_i\ (w.f)$

Proof.

Exchange property / associativity □

Key combinatorial lemma



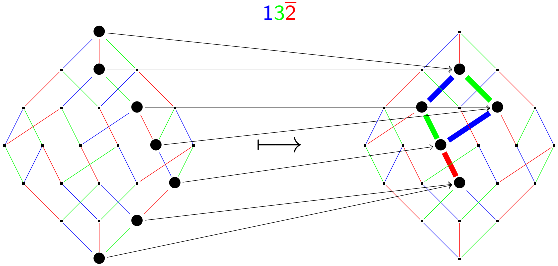
Lemma

For $f \in M(W)$ and $w \in W$: $\quad (s_i w).f = w.f \quad$ *or* $\quad s_i\,(w.f)$

Proof.

Exchange property / associativity □

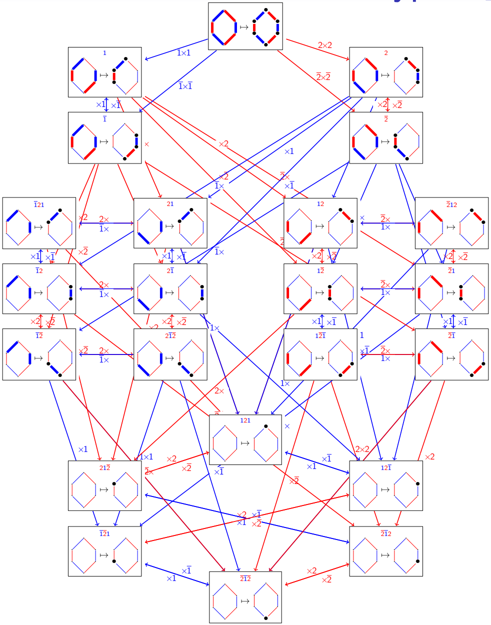
Key combinatorial lemma



Corollary

- *If* $w = uv$*, then* $(uv).f = u' \, (v.f)$*, where* $u' <_B u$
- *Preservation of left order:* $u \leq_L v \Longrightarrow u.f \leq_L v.f$
- *Preservation of Bruhat order:* $u \leq_B v \Longrightarrow u.f \leq_B v.f$
- f *in* $M(W)$ *is determined by its fibers and* $f(1)$

The biHecke monoid for type A_2



Representation theory of $M(W)$

Theorem (HST'08)

$M(W)$ admits $|W|$ simple modules

Sketch of proof.

- M acts transitively on intervals $[u, v]_L$
- The image set of an idempotent is an interval $[u, v]_L$
- $\exists!$ e_w idempotent with image set $[1, w]_L$, for any $w \in W$
- $(e_w)_{w \in W}$: transversal of the regular J-classes
    - $f = uev$ if and only if $\mathrm{im}(f)$ is a subinterval of $\mathrm{im}(e)$

□

Problem

Dimension of simple and projective modules?

Representation theory of $M(W)$

Theorem (HST'08)

$M(W)$ admits $|W|$ simple modules

Sketch of proof.

- M acts transitively on intervals $[u, v]_L$
- The image set of an idempotent is an interval $[u, v]_L$
- $\exists!\ e_w$ idempotent with image set $[1, w]_L$, for any $w \in W$
- $(e_w)_{w \in W}$: transversal of the regular J-classes
  - $f = uev$ if and only if $\mathrm{im}(f)$ is a subinterval of $\mathrm{im}(e)$

□

Problem

Dimension of simple and projective modules?

Representation theory of $M(W)$

Theorem (HST'08)

$M(W)$ admits $|W|$ simple modules

Sketch of proof.

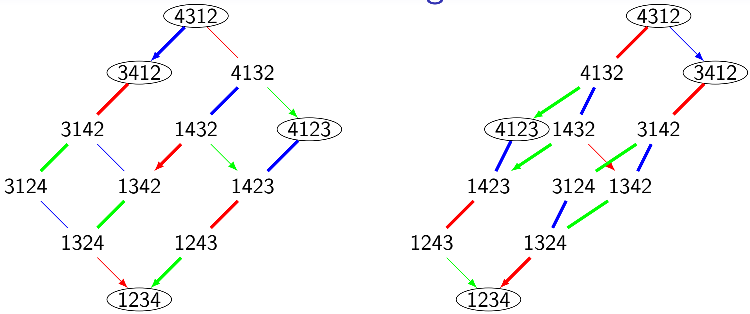
- M acts transitively on intervals $[u, v]_L$
- The image set of an idempotent is an interval $[u, v]_L$
- $\exists!$ e_w idempotent with image set $[1, w]_L$, for any $w \in W$
- $(e_w)_{w \in W}$: transversal of the regular J-classes
  - $f = uev$ if and only if $\mathrm{im}(f)$ is a subinterval of $\mathrm{im}(e)$

□

Problem

Dimension of simple and projective modules?

Translation algebras



Definition (Translation algebra)

$T_w := \mathbb{Q}[\pi_1, \pi_2, \ldots, \overline{\pi}_1, \overline{\pi}_2, \ldots]$ acting on $\mathbb{Q}.[1, w]_R$

- Blocks: $J = \{\}, \{1, 2\}, \{3\}, \{1, 2, 3\} \quad \Longrightarrow \quad$ Submodules P_J
- T_w: max. algebra stabilizing all $P_J \quad \Longrightarrow \quad$ Repr. theory
- T_w quotient of $\mathbb{Q}[M(W)]$; top: simple module S_w of M
- Dimension: inclusion-exclusion along the cutting poset
- Generating series calculation?

Conclusion

General strategy:

- Find combinatorial models for algebras and representations
- As simple as possible, but no simpler
- Concrete and effective
- Find the right point of view where proofs become trivial
- Use representation theory and computer exploration as a guide

Conclusion

General strategy:

- Find combinatorial models for algebras and representations
- As simple as possible, but no simpler
- Concrete and effective
- Find the right point of view where proofs become trivial
- Use representation theory and computer exploration as a guide

Conclusion

General strategy:

- Find combinatorial models for algebras and representations
- As simple as possible, but no simpler
- Concrete and effective
- Find the right point of view where proofs become trivial
- Use representation theory and computer exploration as a guide

Conclusion

General strategy:

- Find combinatorial models for algebras and representations
- As simple as possible, but no simpler
- Concrete and effective
- Find the right point of view where proofs become trivial
- Use representation theory and computer exploration as a guide

Conclusion

General strategy:

- Find combinatorial models for algebras and representations
- As simple as possible, but no simpler
- Concrete and effective
- Find the right point of view where proofs become trivial
- Use representation theory and computer exploration as a guide

Le projet Sage-Combinat (2000-2011)

Mission

« Améliorer MuPAD/Sage comme boîte à outils extensible pour l'exploration informatique en combinatoire algébrique, en fédérant et mutualisant les efforts de développements des chercheurs »

Stratégie

- Licence libre pour partager avec le plus grand nombre
  En restant pragmatique dans les collaborations
- Développement décentralisé et international
  Garantie d'indépendance vis-à-vis des tutelles
- Développé par des chercheurs pour des chercheurs
  Avec un usage plus large en vue
- Coeur du développement par des permanents
  Les doctorants se concentrent sur leurs propres besoins
- Chaque ligne de code justifiée par un projet de recherche
  Avec une vision à long terme (développement agile)
- Inspiration des informaticiens:
  Concepts et méthodologies de programmation
  Outils de développement coopératif

Le projet Sage-Combinat (2000-2011)

Mission

« Améliorer MuPAD/Sage comme boîte à outils extensible pour l'exploration informatique en combinatoire algébrique, en fédérant et mutualisant les efforts de développements des chercheurs »

Stratégie

- Licence libre pour partager avec le plus grand nombre
  En restant pragmatique dans les collaborations
- Développement décentralisé et international
  Garantie d'indépendance vis-à-vis des tutelles
- Développé par des chercheurs pour des chercheurs
  Avec un usage plus large en vue
- Coeur du développement par des permanents
  Les doctorants se concentrent sur leurs propres besoins
- Chaque ligne de code justifiée par un projet de recherche
  Avec une vision à long terme (développement agile)
- Inspiration des informaticiens:
  Concepts et méthodologies de programmation
  Outils de développement coopératif

Le projet Sage-Combinat: 10 ans après

En quelques chiffres

- MuPAD/C++: 600 classes, 5000 méthodes, 160k lignes de code+doc+tests
- Sage (Python): 300 tickets / 200k lignes de code+doc+tests
- Financements: ANR, PEPS, NSF, Google Summer of Code
- **70+ articles de recherche:**
  - Combinatoire algébrique, énumérative, des mots, ...
  - Test de programmes (LRI: Denise,Gaudel,Gouraud,Oudinet)

Une communauté internationale

- Barcelonne, Davis, Lyon, Marne, Marseille, Montpellier, Orsay, Paris, Rouen, Philadelphie, Seattle, Stanford, Montreal, Toronto, ...
- Abbad, Berg, Borie, Bump, Bandlow, Boussicault, Chapoton, Delecroix, Dehaye, Denton, Descouens, Drake, Gomez Diaz, Feray, Hansen, Hemmecke, Hivert, Jones, Labbé, Laigle-Chapuy, Laugerotte, Lemeur, Mathas, Molinero, Monteil, Musiker, Novelli, Nzeutchap, Pon, Rubey, Saliola, Schilling, Shimozono, Stump, Tevlin, Thiéry, Walker, Wang, Zabrocki, ...

Le projet Sage-Combinat: 10 ans après

En quelques chiffres

- MuPAD/C++: 600 classes, 5000 méthodes, 160k lignes de code+doc+tests
- Sage (Python): 300 tickets / 200k lignes de code+doc+tests
- Financements: ANR, PEPS, NSF, Google Summer of Code
- **70+ articles de recherche:**
  - Combinatoire algébrique, énumérative, des mots, ...
  - Test de programmes (LRI: Denise,Gaudel,Gouraud,Oudinet)

Une communauté internationale

- Barcelonne, Davis, Lyon, Marne, Marseille, Montpellier, Orsay, Paris, Rouen, Philadelphie, Seattle, Stanford, Montreal, Toronto, ...
- Abbad, Berg, Borie, Bump, Bandlow, Boussicault, Chapoton, Delecroix, Dehaye, Denton, Descouens, Drake, Gomez Diaz, Feray, Hansen, Hemmecke, Hivert, Jones, Labbé, Laigle-Chapuy, Laugerotte, Lemeur, Mathas, Molinero, Monteil, Musiker, Novelli, Nzeutchap, Pon, Rubey, Saliola, Schilling, Shimozono, Stump, Tevlin, Thiéry, Walker, Wang, Zabrocki, ...

Le projet Sage-Combinat: 10 ans après

En quelques chiffres

- `MuPAD/C++`: 600 classes, 5000 méthodes, 160k lignes de code+doc+tests
- `Sage (Python)`: 300 tickets / 200k lignes de code+doc+tests
- Financements: ANR, PEPS, NSF, Google Summer of Code
- **70+ articles de recherche:**
  - Combinatoire algébrique, énumérative, des mots, ...
  - Test de programmes (LRI: Denise,Gaudel,Gouraud,Oudinet)

Une communauté internationale

- Barcelonne, Davis, Lyon, Marne, Marseille, Montpellier, Orsay, Paris, Rouen, Philadelphie, Seattle, Stanford, Montreal, Toronto, ...
- Abbad, Berg, Borie, Bump, Bandlow, Boussicault, Chapoton, Delecroix, Dehaye, Denton, Descouens, Drake, Gomez Diaz, Feray, Hansen, Hemmecke, Hivert, Jones, Labbé, Laigle-Chapuy, Laugerotte, Lemeur, Mathas, Molinero, Monteil, Musiker, Novelli, Nzeutchap, Pon, Rubey, Saliola, Schilling, Shimozono, Stump, Tevlin, Thiéry, Walker, Wang, Zabrocki, ...

Le projet Sage-Combinat: 10 ans après

En quelques chiffres

- `MuPAD/C++`: 600 classes, 5000 méthodes, 160k lignes de code+doc+tests
- `Sage (Python)`: 300 tickets / 200k lignes de code+doc+tests
- Financements: ANR, PEPS, NSF, Google Summer of Code
- **70+ articles de recherche:**
  - Combinatoire algébrique, énumérative, des mots, ...
  - Test de programmes (LRI: Denise,Gaudel,Gouraud,Oudinet)

Une communauté internationale

- Barcelonne, Davis, Lyon, Marne, Marseille, Montpellier, Orsay, Paris, Rouen, Philadelphie, Seattle, Stanford, Montreal, Toronto, ...
- Abbad, Berg, Borie, Bump, Bandlow, Boussicault, Chapoton, Delecroix, Dehaye, Denton, Descouens, Drake, Gomez Diaz, Feray, Hansen, Hemmecke, Hivert, Jones, Labbé, Laigle-Chapuy, Laugerotte, Lemeur, Mathas, Molinero, Monteil, Musiker, Novelli, Nzeutchap, Pon, Rubey, Saliola, Schilling, Shimozono, Stump, Tevlin, Thiéry, Walker, Wang, Zabrocki, ...