An invitation to Hecke-Kiselman monoids

Alessandro D'Andrea

Università di Roma "La Sapienza"

Second Antipode Workshop
September 12 2022

Coxeter systems

Coxeter system: combinatorial information contained in a (unoriented) graph Γ

- vertices: generators (of order 2) of a group
- edges: relations (e.g., commutation, braid) between generators

One obtains a presentation

$$\begin{aligned}\langle s_i \mid s_i^2 &= 1 \\ s_is_j &= s_js_i \text{ if } i \text{ and } j \text{ are not connected by an edge} \\ s_is_js_i &= s_js_is_j \text{ if } i \text{ and } j \text{ are connected by an edge } \rangle\end{aligned}$$

which yields a group W_Γ.

In this talk, everything is simply laced.

Coxeter systems

Coxeter system: combinatorial information contained in a (unoriented) graph Γ

- vertices: generators (of order 2) of a group
- edges: relations (e.g., commutation, braid) between generators

One obtains a presentation

$$\begin{aligned}\langle s_i \mid s_i^2 &= 1 \\ s_i s_j &= s_j s_i \text{ if } i \text{ and } j \text{ are not connected by an edge} \\ s_i s_j s_i &= s_j s_i s_j \text{ if } i \text{ and } j \text{ are connected by an edge } \rangle\end{aligned}$$

which yields a group W_Γ.

In this talk, everything is simply laced.

Coxeter systems

Coxeter system: combinatorial information contained in a (unoriented) graph Γ

- vertices: generators (of order 2) of a group
- edges: relations (e.g., commutation, braid) between generators

One obtains a presentation

$$\begin{aligned} \langle s_i \mid s_i^2 &= 1 \\ s_i s_j &= s_j s_i \text{ if } i \text{ and } j \text{ are not connected by an edge} \\ s_i s_j s_i &= s_j s_i s_j \text{ if } i \text{ and } j \text{ are connected by an edge } \rangle \end{aligned}$$

which yields a group W_Γ.

In this talk, everything is simply laced.

Coxeter systems

Coxeter system: combinatorial information contained in a (unoriented) graph Γ

- vertices: generators (of order 2) of a group
- edges: relations (e.g., commutation, braid) between generators

One obtains a presentation

$$\begin{aligned} \langle s_i \mid s_i^2 &= 1 \\ s_i s_j &= s_j s_i \text{ if } i \text{ and } j \text{ are not connected by an edge} \\ s_i s_j s_i &= s_j s_i s_j \text{ if } i \text{ and } j \text{ are connected by an edge } \rangle \end{aligned}$$

which yields a group W_Γ.

In this talk, everything is simply laced.

Coxeter systems

Coxeter system: combinatorial information contained in a (unoriented) graph Γ

- vertices: generators (of order 2) of a group
- edges: relations (e.g., commutation, braid) between generators

One obtains a presentation

$$\begin{aligned}\langle s_i \mid s_i^2 = 1 \\ & s_i s_j = s_j s_i \text{ if } i \text{ and } j \text{ are not connected by an edge} \\ & s_i s_j s_i = s_j s_i s_j \text{ if } i \text{ and } j \text{ are connected by an edge } \rangle\end{aligned}$$

which yields a group W_Γ.

In this talk, everything is simply laced.

Coxeter systems

Coxeter system: combinatorial information contained in a (unoriented) graph Γ

- vertices: generators (of order 2) of a group
- edges: relations (e.g., commutation, braid) between generators

One obtains a presentation

$$\begin{aligned}\langle s_i \mid s_i^2 &= 1 \\ s_i s_j &= s_j s_i \text{ if } i \text{ and } j \text{ are not connected by an edge} \\ s_i s_j s_i &= s_j s_i s_j \text{ if } i \text{ and } j \text{ are connected by an edge } \rangle\end{aligned}$$

which yields a group W_Γ.

In this talk, everything is simply laced.

Coxeter systems

Coxeter system: combinatorial information contained in a (unoriented) graph Γ

- vertices: generators (of order 2) of a group
- edges: relations (e.g., commutation, braid) between generators

One obtains a presentation

$$\begin{aligned} \langle s_i \mid s_i^2 &= 1 \\ s_i s_j &= s_j s_i \text{ if } i \text{ and } j \text{ are not connected by an edge} \\ s_i s_j s_i &= s_j s_i s_j \text{ if } i \text{ and } j \text{ are connected by an edge } \rangle \end{aligned}$$

which yields a group W_Γ.

In this talk, everything is simply laced.

Coxeter systems

Coxeter system: combinatorial information contained in a (unoriented) graph Γ

- vertices: generators (of order 2) of a group
- edges: relations (e.g., commutation, braid) between generators

One obtains a presentation

$$\begin{aligned} \langle s_i \mid s_i^2 &= 1 \\ s_i s_j &= s_j s_i \text{ if } i \text{ and } j \text{ are not connected by an edge} \\ s_i s_j s_i &= s_j s_i s_j \text{ if } i \text{ and } j \text{ are connected by an edge } \rangle \end{aligned}$$

which yields a group W_Γ.

In this talk, everything is simply laced.

Coxeter monoids

The above presentation can be deformed (in an associative algebra context) to:

$$s_i^2 = 1 \rightsquigarrow (s_i + 1)(s_i - q) = 0.$$

One obtains a $\mathbb{Z}[q^{\pm 1}]$-algebra (Iwahori-Hecke) which can be specialized to complex values of q.

- $q = 1 \to$ group algebra of the Coxeter group W_Γ
- generic values of $q \to$ generators s_i do not close under product
- $q = 0 \to$ monoid algebra of the Coxeter monoid (generated by $a_i = -s_i$)   "0-Hecke algebras", Norton 1979

Coxeter monoids

The above presentation can be deformed (in an associative algebra context) to:

$$s_i^2 = 1 \rightsquigarrow (s_i + 1)(s_i - q) = 0.$$

One obtains a $\mathbb{Z}[q^{\pm 1}]$-algebra (Iwahori-Hecke) which can be specialized to complex values of q.

- $q = 1 \to$ group algebra of the Coxeter group W_Γ
- generic values of $q \to$ generators s_i do not close under product
- $q = 0 \to$ monoid algebra of the Coxeter monoid (generated by $a_i = -s_i$) "0-Hecke algebras", Norton 1979

Coxeter monoids

The above presentation can be deformed (in an associative algebra context) to:

$$s_i^2 = 1 \rightsquigarrow (s_i + 1)(s_i - q) = 0.$$

One obtains a $\mathbb{Z}[q^{\pm 1}]$-algebra (Iwahori-Hecke) which can be specialized to complex values of q.

- $q = 1 \to$ group algebra of the Coxeter group W_Γ
- generic values of $q \to$ generators s_i do not close under product
- $q = 0 \to$ monoid algebra of the Coxeter monoid (generated by $a_i = -s_i$) "0-Hecke algebras", Norton 1979

Coxeter monoids

The above presentation can be deformed (in an associative algebra context) to:

$$s_i^2 = 1 \rightsquigarrow (s_i + 1)(s_i - q) = 0.$$

One obtains a $\mathbb{Z}[q^{\pm 1}]$-algebra (Iwahori-Hecke) which can be specialized to complex values of q.

- $q = 1 \rightarrow$ group algebra of the Coxeter group W_Γ
- generic values of $q \rightarrow$ generators s_i do not close under product
- $q = 0 \rightarrow$ monoid algebra of the Coxeter monoid (generated by $a_i = -s_i$) "0-Hecke algebras", Norton 1979

Coxeter monoids

The above presentation can be deformed (in an associative algebra context) to:

$$s_i^2 = 1 \rightsquigarrow (s_i + 1)(s_i - q) = 0.$$

One obtains a $\mathbb{Z}[q^{\pm 1}]$-algebra (Iwahori-Hecke) which can be specialized to complex values of q.

- $q = 1 \to$ group algebra of the Coxeter group W_Γ
- generic values of $q \to$ generators s_i do not close under product
- $q = 0 \to$ monoid algebra of the Coxeter monoid (generated by $a_i = -s_i$) "0-Hecke algebras", Norton 1979

Coxeter monoids

The above presentation can be deformed (in an associative algebra context) to:

$$s_i^2 = 1 \rightsquigarrow (s_i + 1)(s_i - q) = 0.$$

One obtains a $\mathbb{Z}[q^{\pm 1}]$-algebra (Iwahori-Hecke) which can be specialized to complex values of q.

- $q = 1 \to$ group algebra of the Coxeter group W_Γ
- generic values of $q \to$ generators s_i do not close under product
- $q = 0 \to$ monoid algebra of the Coxeter monoid (generated by $a_i = -s_i$) "0-Hecke algebras", Norton 1979

Coxeter monoids

The above presentation can be deformed (in an associative algebra context) to:

$$s_i^2 = 1 \rightsquigarrow (s_i + 1)(s_i - q) = 0.$$

One obtains a $\mathbb{Z}[q^{\pm 1}]$-algebra (Iwahori-Hecke) which can be specialized to complex values of q.

- $q = 1 \to$ group algebra of the Coxeter group W_Γ
- generic values of $q \to$ generators s_i do not close under product
- $q = 0 \to$ monoid algebra of the Coxeter monoid (generated by $a_i = -s_i$) "0-Hecke algebras", Norton 1979

Coxeter monoids

The above presentation can be deformed (in an associative algebra context) to:

$$s_i^2 = 1 \rightsquigarrow (s_i + 1)(s_i - q) = 0.$$

One obtains a $\mathbb{Z}[q^{\pm 1}]$-algebra (Iwahori-Hecke) which can be specialized to complex values of q.

- $q = 1 \to$ group algebra of the Coxeter group W_Γ
- generic values of $q \to$ generators s_i do not close under product
- $q = 0 \to$ monoid algebra of the Coxeter monoid (generated by $a_i = -s_i$) "0-Hecke algebras", Norton 1979

Coxeter monoids

The Coxeter monoid has the same order as the Coxeter group (it can be viewed as a different product on the same set). It also appears as the Richardson-Springer monoid (when dealing with combinatorics of B-orbits in spherical varieties).

Coxeter monoids are also known as 0-Hecke monoids.

Knowledge of both the Coxeter group and the Coxeter monoid up to isomorphism determines the Coxeter system.
"Coxeter groups, Coxeter monoids and the Bruhat order" Kenney 2014

Coxeter monoids

The Coxeter monoid has the same order as the Coxeter group (it can be viewed as a different product on the same set). It also appears as the Richardson-Springer monoid (when dealing with combinatorics of B-orbits in spherical varieties).

Coxeter monoids are also known as 0-Hecke monoids.

Knowledge of both the Coxeter group and the Coxeter monoid up to isomorphism determines the Coxeter system.
"Coxeter groups, Coxeter monoids and the Bruhat order" Kenney 2014

Coxeter monoids

The Coxeter monoid has the same order as the Coxeter group (it can be viewed as a different product on the same set). It also appears as the Richardson-Springer monoid (when dealing with combinatorics of B-orbits in spherical varieties).

Coxeter monoids are also known as 0-Hecke monoids.

Knowledge of both the Coxeter group and the Coxeter monoid up to isomorphism determines the Coxeter system.
"Coxeter groups, Coxeter monoids and the Bruhat order" Kenney 2014

Coxeter monoids

The Coxeter monoid has the same order as the Coxeter group (it can be viewed as a different product on the same set). It also appears as the Richardson-Springer monoid (when dealing with combinatorics of B-orbits in spherical varieties).

Coxeter monoids are also known as 0-Hecke monoids.

Knowledge of both the Coxeter group and the Coxeter monoid up to isomorphism determines the Coxeter system.
"Coxeter groups, Coxeter monoids and the Bruhat order" Kenney 2014

Quotients of Coxeter monoids

(Quotients of) Coxeter monoids appear in the literature. Examples:

- Kiselman's semigroup and its generalizations
- Catalan monoid

Quotients of Coxeter monoids

(Quotients of) Coxeter monoids appear in the literature. Examples:

- Kiselman's semigroup and its generalizations
- Catalan monoid

Quotients of Coxeter monoids

(Quotients of) Coxeter monoids appear in the literature. Examples:

- Kiselman's semigroup and its generalizations
- Catalan monoid

Quotients of Coxeter monoids

(Quotients of) Coxeter monoids appear in the literature. Examples:

- Kiselman's semigroup and its generalizations
- Catalan monoid

Kiselman's semigroup

In convexity theory, one may attach to a function $f : E \to \mathbb{R} \cup \{\pm\infty\}$

- $c(f)$: the largest convex function not exceeding f
- $l(f)$: the largest lower semicontinuous function not exceeding f
- $m(f) = f$ if $f > -\infty$ everywhere; $m(f) \equiv -\infty$ otherwise

Then c, l, m are idempotent operators, and satisfy

$$clc = lcl = lc$$
$$cmc = mcm = mc$$
$$lml = mlm = ml.$$

The monoid $\langle c, l, m\rangle$ has at most 18 elements. Indeed exactly 18 when E is a real infinite-dimensional normed space, and in this case the above relations provide a presentation.
The Kiselman monoid K_n generalizes the above presentation but admits n generators.

Kiselman's semigroup

In convexity theory, one may attach to a function $f : E \to \mathbb{R} \cup \{\pm\infty\}$

- $c(f)$: the largest convex function not exceeding f
- $l(f)$: the largest lower semicontinuous function not exceeding f
- $m(f) = f$ if $f > -\infty$ everywhere; $m(f) \equiv -\infty$ otherwise

Then c, l, m are idempotent operators, and satisfy

$$clc = lcl = lc$$
$$cmc = mcm = mc$$
$$lml = mlm = ml.$$

The monoid $\langle c, l, m\rangle$ has at most 18 elements. Indeed exactly 18 when E is a real infinite-dimensional normed space, and in this case the above relations provide a presentation.

The Kiselman monoid K_n generalizes the above presentation but admits n generators.

Kiselman's semigroup

In convexity theory, one may attach to a function $f : E \to \mathbb{R} \cup \{\pm\infty\}$

- $c(f)$: the largest convex function not exceeding f
- $l(f)$: the largest lower semicontinuous function not exceeding f
- $m(f) = f$ if $f > -\infty$ everywhere; $m(f) \equiv -\infty$ otherwise

Then c, l, m are idempotent operators, and satisfy

$$clc = lcl = lc$$
$$cmc = mcm = mc$$
$$lml = mlm = ml.$$

The monoid $\langle c, l, m\rangle$ has at most 18 elements. Indeed exactly 18 when E is a real infinite-dimensional normed space, and in this case the above relations provide a presentation.

The Kiselman monoid K_n generalizes the above presentation but admits n generators.

Kiselman's semigroup

In convexity theory, one may attach to a function $f : E \to \mathbb{R} \cup \{\pm\infty\}$

- $c(f)$: the largest convex function not exceeding f
- $l(f)$: the largest lower semicontinuous function not exceeding f
- $m(f) = f$ if $f > -\infty$ everywhere; $m(f) \equiv -\infty$ otherwise

Then c, l, m are idempotent operators, and satisfy

$$clc = lcl = lc$$
$$cmc = mcm = mc$$
$$lml = mlm = ml.$$

The monoid $\langle c, l, m \rangle$ has at most 18 elements. Indeed exactly 18 when E is a real infinite-dimensional normed space, and in this case the above relations provide a presentation.

The Kiselman monoid K_n generalizes the above presentation but admits n generators.

Kiselman's semigroup

In convexity theory, one may attach to a function $f : E \to \mathbb{R} \cup \{\pm\infty\}$

- $c(f)$: the largest convex function not exceeding f
- $l(f)$: the largest lower semicontinuous function not exceeding f
- $m(f) = f$ if $f > -\infty$ everywhere; $m(f) \equiv -\infty$ otherwise

Then c, l, m are idempotent operators, and satisfy

$$clc = lcl = lc$$
$$cmc = mcm = mc$$
$$lml = mlm = ml.$$

The monoid $\langle c, l, m \rangle$ has at most 18 elements. Indeed exactly 18 when E is a real infinite-dimensional normed space, and in this case the above relations provide a presentation.

The Kiselman monoid K_n generalizes the above presentation but admits n generators.

Kiselman's semigroup

In convexity theory, one may attach to a function $f : E \to \mathbb{R} \cup \{\pm\infty\}$

- $c(f)$: the largest convex function not exceeding f
- $l(f)$: the largest lower semicontinuous function not exceeding f
- $m(f) = f$ if $f > -\infty$ everywhere; $m(f) \equiv -\infty$ otherwise

Then c, l, m are idempotent operators, and satisfy

$$clc = lcl = lc$$
$$cmc = mcm = mc$$
$$lml = mlm = ml.$$

The monoid $\langle c, l, m \rangle$ has at most 18 elements. Indeed exactly 18 when E is a real infinite-dimensional normed space, and in this case the above relations provide a presentation.

The Kiselman monoid K_n generalizes the above presentation but admits n generators.

Kiselman's semigroup

In convexity theory, one may attach to a function $f : E \to \mathbb{R} \cup \{\pm\infty\}$

- $c(f)$: the largest convex function not exceeding f
- $l(f)$: the largest lower semicontinuous function not exceeding f
- $m(f) = f$ if $f > -\infty$ everywhere; $m(f) \equiv -\infty$ otherwise

Then c, l, m are idempotent operators, and satisfy

$$clc = lcl = lc$$
$$cmc = mcm = mc$$
$$lml = mlm = ml.$$

The monoid $\langle c, l, m\rangle$ has at most 18 elements. Indeed exactly 18 when E is a real infinite-dimensional normed space, and in this case the above relations provide a presentation.

The Kiselman monoid K_n generalizes the above presentation but admits n generators.

Kiselman's semigroup

In convexity theory, one may attach to a function $f : E \to \mathbb{R} \cup \{\pm\infty\}$

- $c(f)$: the largest convex function not exceeding f
- $l(f)$: the largest lower semicontinuous function not exceeding f
- $m(f) = f$ if $f > -\infty$ everywhere; $m(f) \equiv -\infty$ otherwise

Then c, l, m are idempotent operators, and satisfy

$$clc = lcl = lc$$
$$cmc = mcm = mc$$
$$lml = mlm = ml.$$

The monoid $\langle c, l, m\rangle$ has at most 18 elements. Indeed exactly 18 when E is a real infinite-dimensional normed space, and in this case the above relations provide a presentation.

The Kiselman monoid K_n generalizes the above presentation but admits n generators.

Kiselman's semigroup

In convexity theory, one may attach to a function $f : E \to \mathbb{R} \cup \{\pm\infty\}$

- $c(f)$: the largest convex function not exceeding f
- $l(f)$: the largest lower semicontinuous function not exceeding f
- $m(f) = f$ if $f > -\infty$ everywhere; $m(f) \equiv -\infty$ otherwise

Then c, l, m are idempotent operators, and satisfy

$$clc = lcl = lc$$
$$cmc = mcm = mc$$
$$lml = mlm = ml.$$

The monoid $\langle c, l, m\rangle$ has at most 18 elements. Indeed exactly 18 when E is a real infinite-dimensional normed space, and in this case the above relations provide a presentation.

The Kiselman monoid K_n generalizes the above presentation but admits n generators.

Kiselman's semigroup

In convexity theory, one may attach to a function $f : E \to \mathbb{R} \cup \{\pm\infty\}$

- $c(f)$: the largest convex function not exceeding f
- $l(f)$: the largest lower semicontinuous function not exceeding f
- $m(f) = f$ if $f > -\infty$ everywhere; $m(f) \equiv -\infty$ otherwise

Then c, l, m are idempotent operators, and satisfy

$$clc = lcl = lc$$
$$cmc = mcm = mc$$
$$lml = mlm = ml.$$

The monoid $\langle c, l, m \rangle$ has at most 18 elements. Indeed exactly 18 when E is a real infinite-dimensional normed space, and in this case the above relations provide a presentation.
The Kiselman monoid K_n generalizes the above presentation but admits n generators.

Reduced expressions in Kiselman's semigroups

Kiselman's semigroup K_n is generated by n idempotents $a_i, i = 1, \ldots, n$.

When $1 \leq i < j \leq n$ one has relations $a_i a_j a_i = a_j a_i a_j = a_i a_j$.

It is easy to show that if between two a_i only $a_j, j > i$, occur, then one may delete the rightmost a_i (similarly if only lower indices occur, one may remove the leftmost occurrence).

The only possible reduced expressions are such that between two identical generators, both higher and lower indices must occur. Using some old results on confluence (Newman 1942; also Huet 1980) one may show that

- Such words are all reduced
- All choices of cancellations from a given word lead to the same (hence unique) reduced expression. (Kudryavtseva, Mazorchuk 2009)

Reduced expressions in Kiselman's semigroups

Kiselman's semigroup K_n is generated by n idempotents $a_i, i = 1, \ldots, n$.

When $1 \leq i < j \leq n$ one has relations $a_i a_j a_i = a_j a_i a_j = a_i a_j$.

It is easy to show that if between two a_i only $a_j, j > i$, occur, then one may delete the rightmost a_i (similarly if only lower indices occur, one may remove the leftmost occurrence).

The only possible reduced expressions are such that between two identical generators, both higher and lower indices must occur. Using some old results on confluence (Newman 1942; also Huet 1980) one may show that

- Such words are all reduced
- All choices of cancellations from a given word lead to the same (hence unique) reduced expression. (Kudryavtseva, Mazorchuk 2009)

Reduced expressions in Kiselman's semigroups

Kiselman's semigroup K_n is generated by n idempotents $a_i, i = 1, \dots, n$.

When $1 \leq i < j \leq n$ one has relations $a_i a_j a_i = a_j a_i a_j = a_i a_j$.

It is easy to show that if between two a_i only $a_j, j > i$, occur, then one may delete the rightmost a_i (similarly if only lower indices occur, one may remove the leftmost occurrence).

The only possible reduced expressions are such that between two identical generators, both higher and lower indices must occur. Using some old results on confluence (Newman 1942; also Huet 1980) one may show that

- Such words are all reduced
- All choices of cancellations from a given word lead to the same (hence unique) reduced expression. (Kudryavtseva, Mazorchuk 2009)

Reduced expressions in Kiselman's semigroups

Kiselman's semigroup K_n is generated by n idempotents $a_i, i = 1, \ldots, n$.

When $1 \leq i < j \leq n$ one has relations $a_i a_j a_i = a_j a_i a_j = a_i a_j$.

It is easy to show that if between two a_i only $a_j, j > i$, occur, then one may delete the rightmost a_i (similarly if only lower indices occur, one may remove the leftmost occurrence).

The only possible reduced expressions are such that between two identical generators, both higher and lower indices must occur. Using some old results on confluence (Newman 1942; also Huet 1980) one may show that

- Such words are all reduced
- All choices of cancellations from a given word lead to the same (hence unique) reduced expression. (Kudryavtseva, Mazorchuk 2009)

Reduced expressions in Kiselman's semigroups

Kiselman's semigroup K_n is generated by n idempotents $a_i, i = 1, \ldots, n$.

When $1 \leq i < j \leq n$ one has relations $a_i a_j a_i = a_j a_i a_j = a_i a_j$.

It is easy to show that if between two a_i only $a_j, j > i$, occur, then one may delete the rightmost a_i (similarly if only lower indices occur, one may remove the leftmost occurrence).

The only possible reduced expressions are such that between two identical generators, both higher and lower indices must occur. Using some old results on confluence (Newman 1942; also Huet 1980) one may show that

- Such words are all reduced
- All choices of cancellations from a given word lead to the same (hence unique) reduced expression. (Kudryavtseva, Mazorchuk 2009)

Reduced expressions in Kiselman's semigroups

Kiselman's semigroup K_n is generated by n idempotents $a_i, i = 1, \ldots, n$.

When $1 \leq i < j \leq n$ one has relations $a_i a_j a_i = a_j a_i a_j = a_i a_j$.

It is easy to show that if between two a_i only $a_j, j > i$, occur, then one may delete the rightmost a_i (similarly if only lower indices occur, one may remove the leftmost occurrence).

The only possible reduced expressions are such that between two identical generators, both higher and lower indices must occur. Using some old results on confluence (Newman 1942; also Huet 1980) one may show that

- Such words are all reduced
- All choices of cancellations from a given word lead to the same (hence unique) reduced expression. (Kudryavtseva, Mazorchuk 2009)

Reduced expressions in Kiselman's semigroups

Kiselman's semigroup K_n is generated by n idempotents $a_i, i = 1, \dots, n$.

When $1 \le i < j \le n$ one has relations $a_i a_j a_i = a_j a_i a_j = a_i a_j$.

It is easy to show that if between two a_i only $a_j, j > i$, occur, then one may delete the rightmost a_i (similarly if only lower indices occur, one may remove the leftmost occurrence).

The only possible reduced expressions are such that between two identical generators, both higher and lower indices must occur. Using some old results on confluence (Newman 1942; also Huet 1980) one may show that

- Such words are all reduced
- All choices of cancellations from a given word lead to the same (hence unique) reduced expression. (Kudryavtseva, Mazorchuk 2009)

Cardinality of Kiselman's semigroups

K_n always has finitely many elements, but its cardinality is not well understood. (A125625: 1, 2, 5, 18, 115, 1710, 83973...) it grows quickly!

- A closed or recursive formula for the cardinality of K_n is missing
- The only concrete estimate (Kudryavtseva, Mazorchuk 2009) in the literature is $|K_n| \leq n^{L(n)}$ where

$$L(n) = \begin{cases} 2^{k+1} - 2 & \text{if } n = 2k \\ 3 \cdot 2^k - 2 & \text{if } n = 2k+1 \end{cases}$$

- Indeed, $\log |K_n| \simeq c2^{n/2}$, separately for even and odd values of n. (joint with Stella)

Cardinality of Kiselman's semigroups

K_n always has finitely many elements, but its cardinality is not well understood. (A125625: 1, 2, 5, 18, 115, 1710, 83973...) it grows quickly!

- A closed or recursive formula for the cardinality of K_n is missing
- The only concrete estimate (Kudryavtseva, Mazorchuk 2009) in the literature is $|K_n| \leq n^{L(n)}$ where

$$L(n) = \begin{cases} 2^{k+1} - 2 & \text{if } n = 2k \\ 3 \cdot 2^k - 2 & \text{if } n = 2k+1 \end{cases}$$

- Indeed, $\log |K_n| \simeq c2^{n/2}$, separately for even and odd values of n. (joint with Stella)

Cardinality of Kiselman's semigroups

K_n always has finitely many elements, but its cardinality is not well understood. (A125625: 1, 2, 5, 18, 115, 1710, 83973...) it grows quickly!

- A closed or recursive formula for the cardinality of K_n is missing
- The only concrete estimate (Kudryavtseva, Mazorchuk 2009) in the literature is $|K_n| \leq n^{L(n)}$ where

$$L(n) = \begin{cases} 2^{k+1} - 2 & \text{if } n = 2k \\ 3 \cdot 2^k - 2 & \text{if } n = 2k+1 \end{cases}$$

- Indeed, $\log |K_n| \simeq c2^{n/2}$, separately for even and odd values of n. (joint with Stella)

Cardinality of Kiselman's semigroups

K_n always has finitely many elements, but its cardinality is not well understood. (A125625: 1, 2, 5, 18, 115, 1710, 83973...) it grows quickly!

- A closed or recursive formula for the cardinality of K_n is missing
- The only concrete estimate (Kudryavtseva, Mazorchuk 2009) in the literature is $|K_n| \leq n^{L(n)}$ where

$$L(n) = \begin{cases} 2^{k+1} - 2 & \text{if } n = 2k \\ 3 \cdot 2^k - 2 & \text{if } n = 2k+1 \end{cases}$$

- Indeed, $\log |K_n| \simeq c2^{n/2}$, separately for even and odd values of n. (joint with Stella)

Cardinality of Kiselman's semigroups

K_n always has finitely many elements, but its cardinality is not well understood. (A125625: 1, 2, 5, 18, 115, 1710, 83973...) it grows quickly!

- A closed or recursive formula for the cardinality of K_n is missing
- The only concrete estimate (Kudryavtseva, Mazorchuk 2009) in the literature is $|K_n| \leq n^{L(n)}$ where

$$L(n) = \begin{cases} 2^{k+1} - 2 & \text{if } n = 2k \\ 3 \cdot 2^k - 2 & \text{if } n = 2k+1 \end{cases}$$

- Indeed, $\log |K_n| \simeq c2^{n/2}$, separately for even and odd values of n. (joint with Stella)

Cardinality of Kiselman's semigroups

K_n always has finitely many elements, but its cardinality is not well understood. (A125625: 1, 2, 5, 18, 115, 1710, 83973...) it grows quickly!

- A closed or recursive formula for the cardinality of K_n is missing
- The only concrete estimate (Kudryavtseva, Mazorchuk 2009) in the literature is $|K_n| \leq n^{L(n)}$ where

$$L(n) = \begin{cases} 2^{k+1} - 2 & \text{if } n = 2k \\ 3 \cdot 2^k - 2 & \text{if } n = 2k+1 \end{cases}$$

- Indeed, $\log |K_n| \simeq c2^{n/2}$, separately for even and odd values of n. (joint with Stella)

Cardinality of Kiselman's semigroups

K_n always has finitely many elements, but its cardinality is not well understood. (A125625: 1, 2, 5, 18, 115, 1710, 83973...) it grows quickly!

- A closed or recursive formula for the cardinality of K_n is missing
- The only concrete estimate (Kudryavtseva, Mazorchuk 2009) in the literature is $|K_n| \leq n^{L(n)}$ where

$$L(n) = \begin{cases} 2^{k+1} - 2 & \text{if } n = 2k \\ 3 \cdot 2^k - 2 & \text{if } n = 2k + 1 \end{cases}$$

- Indeed, $\log |K_n| \simeq c2^{n/2}$, separately for even and odd values of n. (joint with Stella)

Catalan monoid

Order decreasing, order preserving functions $f : \{1, \ldots, n\} \to \{1, \ldots, n\}$ form a monoid C_n with respect to composition.

The cardinality of C_n is given by the n-th Catalan number.

C_n has been considered in computer science in the context of hashing and storing/retrieval of information.

$$
\begin{aligned}
C_n = \langle a_i, i = 1, \ldots, n-1 \mid\, & a_i^2 = a_i \\
& a_i a_j = a_j a_i \text{ if } |i-j| > 1 \\
& a_i a_{i+1} a_i = a_{i+1} a_i a_{i+1} = a_i a_{i+1} \rangle
\end{aligned}
$$

Here a_i is the function mapping $i+1$ to i and fixing all other elements.

Catalan monoid

Order decreasing, order preserving functions $f : \{1, \dots, n\} \to \{1, \dots, n\}$ form a monoid C_n with respect to composition.

The cardinality of C_n is given by the n-th Catalan number.

C_n has been considered in computer science in the context of hashing and storing/retrieval of information.

$$
\begin{aligned}
C_n = \langle a_i, i = 1, \dots, n-1 \mid\ & a_i^2 = a_i \\
& a_i a_j = a_j a_i \text{ if } |i-j| > 1 \\
& a_i a_{i+1} a_i = a_{i+1} a_i a_{i+1} = a_i a_{i+1} \rangle
\end{aligned}
$$

Here a_i is the function mapping $i+1$ to i and fixing all other elements.

Catalan monoid

Order decreasing, order preserving functions $f : \{1, \dots, n\} \to \{1, \dots, n\}$ form a monoid C_n with respect to composition.

The cardinality of C_n is given by the n-th Catalan number.

C_n has been considered in computer science in the context of hashing and storing/retrieval of information.

$$C_n = \langle a_i, i = 1, \dots, n-1 \mid a_i^2 = a_i \\ a_i a_j = a_j a_i \text{ if } |i-j| > 1 \\ a_i a_{i+1} a_i = a_{i+1} a_i a_{i+1} = a_i a_{i+1} \rangle$$

Here a_i is the function mapping $i+1$ to i and fixing all other elements.

Catalan monoid

Order decreasing, order preserving functions $f : \{1, \ldots, n\} \to \{1, \ldots, n\}$ form a monoid C_n with respect to composition.

The cardinality of C_n is given by the n-th Catalan number.

C_n has been considered in computer science in the context of hashing and storing/retrieval of information.

$$
\begin{aligned}
C_n = \langle a_i, i = 1, \ldots, n-1 \mid\, & a_i^2 = a_i \\
& a_i a_j = a_j a_i \text{ if } |i-j| > 1 \\
& a_i a_{i+1} a_i = a_{i+1} a_i a_{i+1} = a_i a_{i+1} \rangle
\end{aligned}
$$

Here a_i is the function mapping $i+1$ to i and fixing all other elements.

Catalan monoid

Order decreasing, order preserving functions $f : \{1, \ldots, n\} \to \{1, \ldots, n\}$ form a monoid C_n with respect to composition.

The cardinality of C_n is given by the n-th Catalan number.

C_n has been considered in computer science in the context of hashing and storing/retrieval of information.

$$\begin{aligned} C_n = \langle a_i, i = 1, \ldots, n-1 \mid\, & a_i^2 = a_i \\ & a_i a_j = a_j a_i \text{ if } |i-j| > 1 \\ & a_i a_{i+1} a_i = a_{i+1} a_i a_{i+1} = a_i a_{i+1} \rangle \end{aligned}$$

Here a_i is the function mapping $i+1$ to i and fixing all other elements.

Hecke-Kiselman monoids

A general setting generalizing all above examples is that of Kiselman quotients of 0-Hecke monoids or Hecke-Kiselman monoids, for short.

- The combinatorial informations is contained in a digraph (with both oriented and unoriented edges) Γ yielding a presentation of a monoid HK_Γ:
- one has an idempotent generator a_i for each vertex i;
- $a_i a_j a_i = a_j a_i a_j$ if i and j are connected by an unoriented edge = side,
- $a_i a_j a_i = a_j a_i a_j = a_i a_j$ if there is an oriented edge = arrow, connecting i to j.
- $a_i a_j = a_j a_i$ if i and j are not connected,

There is at most one edge between any two vertices.

Reduced expressions are as before, once one takes commutation relations into account. (joint with Aragona, 2020)

Hecke-Kiselman monoids

A general setting generalizing all above examples is that of Kiselman quotients of 0-Hecke monoids or Hecke-Kiselman monoids, for short.

- The combinatorial informations is contained in a digraph (with both oriented and unoriented edges) Γ yielding a presentation of a monoid HK_Γ:
- one has an idempotent generator a_i for each vertex i;
- $a_i a_j a_i = a_j a_i a_j$ if i and j are connected by an unoriented edge = side,
- $a_i a_j a_i = a_j a_i a_j = a_i a_j$ if there is an oriented edge = arrow, connecting i to j.
- $a_i a_j = a_j a_i$ if i and j are not connected,

There is at most one edge between any two vertices.

Reduced expressions are as before, once one takes commutation relations into account. (joint with Aragona, 2020)

Hecke-Kiselman monoids

A general setting generalizing all above examples is that of Kiselman quotients of 0-Hecke monoids or Hecke-Kiselman monoids, for short.

- The combinatorial informations is contained in a digraph (with both oriented and unoriented edges) Γ yielding a presentation of a monoid HK_Γ:
  - one has an idempotent generator a_i for each vertex i;
  - $a_i a_j a_i = a_j a_i a_j$ if i and j are connected by an unoriented edge = side,
  - $a_i a_j a_i = a_j a_i a_j = a_i a_j$ if there is an oriented edge = arrow, connecting i to j.
  - $a_i a_j = a_j a_i$ if i and j are not connected,

There is at most one edge between any two vertices.

Reduced expressions are as before, once one takes commutation relations into account. (joint with Aragona, 2020)

Hecke-Kiselman monoids

A general setting generalizing all above examples is that of Kiselman quotients of 0-Hecke monoids or Hecke-Kiselman monoids, for short.

- The combinatorial informations is contained in a digraph (with both oriented and unoriented edges) Γ yielding a presentation of a monoid HK_Γ:
  - one has an idempotent generator a_i for each vertex i;
  - $a_i a_j a_i = a_j a_i a_j$ if i and j are connected by an unoriented edge = side,
  - $a_i a_j a_i = a_j a_i a_j = a_i a_j$ if there is an oriented edge = arrow, connecting i to j.
  - $a_i a_j = a_j a_i$ if i and j are not connected,

There is at most one edge between any two vertices.

Reduced expressions are as before, once one takes commutation relations into account. (joint with Aragona, 2020)

Hecke-Kiselman monoids

A general setting generalizing all above examples is that of Kiselman quotients of 0-Hecke monoids or Hecke-Kiselman monoids, for short.

- The combinatorial informations is contained in a digraph (with both oriented and unoriented edges) Γ yielding a presentation of a monoid HK_Γ:
- one has an idempotent generator a_i for each vertex i;
- $a_i a_j a_i = a_j a_i a_j$ if i and j are connected by an unoriented edge = side,
- $a_i a_j a_i = a_j a_i a_j = a_i a_j$ if there is an oriented edge = arrow, connecting i to j.
- $a_i a_j = a_j a_i$ if i and j are not connected,

There is at most one edge between any two vertices.

Reduced expressions are as before, once one takes commutation relations into account. (joint with Aragona, 2020)

Hecke-Kiselman monoids

A general setting generalizing all above examples is that of Kiselman quotients of 0-Hecke monoids or Hecke-Kiselman monoids, for short.

- The combinatorial informations is contained in a digraph (with both oriented and unoriented edges) Γ yielding a presentation of a monoid HK_Γ:
- one has an idempotent generator a_i for each vertex i;
- $a_i a_j a_i = a_j a_i a_j$ if i and j are connected by an unoriented edge = side,
- $a_i a_j a_i = a_j a_i a_j = a_i a_j$ if there is an oriented edge = arrow, connecting i to j.
- $a_i a_j = a_j a_i$ if i and j are not connected,

There is at most one edge between any two vertices.

Reduced expressions are as before, once one takes commutation relations into account. (joint with Aragona, 2020)

Hecke-Kiselman monoids

A general setting generalizing all above examples is that of Kiselman quotients of 0-Hecke monoids or Hecke-Kiselman monoids, for short.

- The combinatorial informations is contained in a digraph (with both oriented and unoriented edges) Γ yielding a presentation of a monoid HK_Γ:
- one has an idempotent generator a_i for each vertex i;
- $a_i a_j a_i = a_j a_i a_j$ if i and j are connected by an unoriented edge = side,
- $a_i a_j a_i = a_j a_i a_j = a_i a_j$ if there is an oriented edge = arrow, connecting i to j.
- $a_i a_j = a_j a_i$ if i and j are not connected,

There is at most one edge between any two vertices.

Reduced expressions are as before, once one takes commutation relations into account. (joint with Aragona, 2020)

Hecke-Kiselman monoids

A general setting generalizing all above examples is that of Kiselman quotients of 0-Hecke monoids or Hecke-Kiselman monoids, for short.

- The combinatorial informations is contained in a digraph (with both oriented and unoriented edges) Γ yielding a presentation of a monoid HK_Γ:
- one has an idempotent generator a_i for each vertex i;
- $a_i a_j a_i = a_j a_i a_j$ if i and j are connected by an unoriented edge = side,
- $a_i a_j a_i = a_j a_i a_j = a_i a_j$ if there is an oriented edge = arrow, connecting i to j.
- $a_i a_j = a_j a_i$ if i and j are not connected,

There is at most one edge between any two vertices.

Reduced expressions are as before, once one takes commutation relations into account. (joint with Aragona, 2020)

Hecke-Kiselman monoids

A general setting generalizing all above examples is that of Kiselman quotients of 0-Hecke monoids or Hecke-Kiselman monoids, for short.

- The combinatorial informations is contained in a digraph (with both oriented and unoriented edges) Γ yielding a presentation of a monoid HK_Γ:
- one has an idempotent generator a_i for each vertex i;
- $a_i a_j a_i = a_j a_i a_j$ if i and j are connected by an unoriented edge = side,
- $a_i a_j a_i = a_j a_i a_j = a_i a_j$ if there is an oriented edge = arrow, connecting i to j.
- $a_i a_j = a_j a_i$ if i and j are not connected,

There is at most one edge between any two vertices.

Reduced expressions are as before, once one takes commutation relations into account. (joint with Aragona, 2020)

Hecke-Kiselman monoids

A general setting generalizing all above examples is that of Kiselman quotients of 0-Hecke monoids or Hecke-Kiselman monoids, for short.

- The combinatorial informations is contained in a digraph (with both oriented and unoriented edges) Γ yielding a presentation of a monoid HK_Γ:
- one has an idempotent generator a_i for each vertex i;
- $a_i a_j a_i = a_j a_i a_j$ if i and j are connected by an unoriented edge = side,
- $a_i a_j a_i = a_j a_i a_j = a_i a_j$ if there is an oriented edge = arrow, connecting i to j.
- $a_i a_j = a_j a_i$ if i and j are not connected,

There is at most one edge between any two vertices.

Reduced expressions are as before, once one takes commutation relations into account. (joint with Aragona, 2020)

Hecke-Kiselman monoids

A general setting generalizing all above examples is that of Kiselman quotients of 0-Hecke monoids or Hecke-Kiselman monoids, for short.

- The combinatorial informations is contained in a digraph (with both oriented and unoriented edges) Γ yielding a presentation of a monoid HK_Γ:
- one has an idempotent generator a_i for each vertex i;
- $a_i a_j a_i = a_j a_i a_j$ if i and j are connected by an unoriented edge = side,
- $a_i a_j a_i = a_j a_i a_j = a_i a_j$ if there is an oriented edge = arrow, connecting i to j.
- $a_i a_j = a_j a_i$ if i and j are not connected,

There is at most one edge between any two vertices.

Reduced expressions are as before, once one takes commutation relations into account. (joint with Aragona, 2020)

Hecke-Kiselman monoids

A general setting generalizing all above examples is that of Kiselman quotients of 0-Hecke monoids or Hecke-Kiselman monoids, for short.

- The combinatorial informations is contained in a digraph (with both oriented and unoriented edges) Γ yielding a presentation of a monoid HK_Γ:
- one has an idempotent generator a_i for each vertex i;
- $a_i a_j a_i = a_j a_i a_j$ if i and j are connected by an unoriented edge = side,
- $a_i a_j a_i = a_j a_i a_j = a_i a_j$ if there is an oriented edge = arrow, connecting i to j.
- $a_i a_j = a_j a_i$ if i and j are not connected,

There is at most one edge between any two vertices.

Reduced expressions are as before, once one takes commutation relations into account. (joint with Aragona, 2020)

Hecke-Kiselman monoids

A general setting generalizing all above examples is that of Kiselman quotients of 0-Hecke monoids or Hecke-Kiselman monoids, for short.

- The combinatorial informations is contained in a digraph (with both oriented and unoriented edges) Γ yielding a presentation of a monoid HK_Γ:
- one has an idempotent generator a_i for each vertex i;
- $a_i a_j a_i = a_j a_i a_j$ if i and j are connected by an unoriented edge = side,
- $a_i a_j a_i = a_j a_i a_j = a_i a_j$ if there is an oriented edge = arrow, connecting i to j.
- $a_i a_j = a_j a_i$ if i and j are not connected,

There is at most one edge between any two vertices.

Reduced expressions are as before, once one takes commutation relations into account. (joint with Aragona, 2020)

Hecke-Kiselman monoids

A general setting generalizing all above examples is that of Kiselman quotients of 0-Hecke monoids or Hecke-Kiselman monoids, for short.

- The combinatorial informations is contained in a digraph (with both oriented and unoriented edges) Γ yielding a presentation of a monoid HK_Γ:
- one has an idempotent generator a_i for each vertex i;
- $a_i a_j a_i = a_j a_i a_j$ if i and j are connected by an unoriented edge = side,
- $a_i a_j a_i = a_j a_i a_j = a_i a_j$ if there is an oriented edge = arrow, connecting i to j.
- $a_i a_j = a_j a_i$ if i and j are not connected,

There is at most one edge between any two vertices.

Reduced expressions are as before, once one takes commutation relations into account. (joint with Aragona, 2020)

Finiteness of Hecke-Kiselman monoids

Commutation $ab = ba$ implies $aba = bab = ab$, thus $HK_{\Gamma'}$ is a quotient of HK_Γ if Γ' if obtained from Γ by:

- removing an arrow
- making a side into an arrow
- removing a side

If Γ' is obtained from Γ by means of a finite sequence of such moves, and HK_Γ is finite, then $HK_{\Gamma'}$ must be finite too. (as it is a quotient)

If Γ has no arrows, HK_Γ is finite iff Γ is a finite disjoint union of finite Dynking diagrams (simply laced $\implies$ ADE classification).

Hence, if HK_Γ is finite, then Γ is obtained by adding arrows to a finite disjoint union of ADE graphs. The converse is false, and apparently very involved.

Finiteness of Hecke-Kiselman monoids

Commutation $ab = ba$ implies $aba = bab = ab$, thus $HK_{\Gamma'}$ is a quotient of HK_Γ if Γ' if obtained from Γ by:

- removing an arrow
- making a side into an arrow
- removing a side

If Γ' is obtained from Γ by means of a finite sequence of such moves, and HK_Γ is finite, then $HK_{\Gamma'}$ must be finite too. (as it is a quotient)

If Γ has no arrows, HK_Γ is finite iff Γ is a finite disjoint union of finite Dynking diagrams (simply laced $\implies$ ADE classification).

Hence, if HK_Γ is finite, then Γ is obtained by adding arrows to a finite disjoint union of ADE graphs. The converse is false, and apparently very involved.

Finiteness of Hecke-Kiselman monoids

Commutation $ab = ba$ implies $aba = bab = ab$, thus $HK_{\Gamma'}$ is a quotient of HK_Γ if Γ' if obtained from Γ by:

- removing an arrow
- making a side into an arrow
- removing a side

If Γ' is obtained from Γ by means of a finite sequence of such moves, and HK_Γ is finite, then $HK_{\Gamma'}$ must be finite too. (as it is a quotient)

If Γ has no arrows, HK_Γ is finite iff Γ is a finite disjoint union of finite Dynking diagrams (simply laced $\implies$ ADE classification).

Hence, if HK_Γ is finite, then Γ is obtained by adding arrows to a finite disjoint union of ADE graphs. The converse is false, and apparently very involved.

Finiteness of Hecke-Kiselman monoids

Commutation $ab = ba$ implies $aba = bab = ab$, thus $HK_{\Gamma'}$ is a quotient of HK_Γ if Γ' if obtained from Γ by:

- removing an arrow
- making a side into an arrow
- removing a side

If Γ' is obtained from Γ by means of a finite sequence of such moves, and HK_Γ is finite, then $HK_{\Gamma'}$ must be finite too. (as it is a quotient)

If Γ has no arrows, HK_Γ is finite iff Γ is a finite disjoint union of finite Dynking diagrams (simply laced $\implies$ ADE classification).

Hence, if HK_Γ is finite, then Γ is obtained by adding arrows to a finite disjoint union of ADE graphs. The converse is false, and apparently very involved.

Finiteness of Hecke-Kiselman monoids

Commutation $ab = ba$ implies $aba = bab = ab$, thus $HK_{\Gamma'}$ is a quotient of HK_Γ if Γ' if obtained from Γ by:

- removing an arrow
- making a side into an arrow
- removing a side

If Γ' is obtained from Γ by means of a finite sequence of such moves, and HK_Γ is finite, then $HK_{\Gamma'}$ must be finite too. (as it is a quotient)

If Γ has no arrows, HK_Γ is finite iff Γ is a finite disjoint union of finite Dynking diagrams (simply laced $\implies$ ADE classification).

Hence, if HK_Γ is finite, then Γ is obtained by adding arrows to a finite disjoint union of ADE graphs. The converse is false, and apparently very involved.

Finiteness of Hecke-Kiselman monoids

Commutation $ab = ba$ implies $aba = bab = ab$, thus $HK_{\Gamma'}$ is a quotient of HK_Γ if Γ' if obtained from Γ by:

- removing an arrow
- making a side into an arrow
- removing a side

If Γ' is obtained from Γ by means of a finite sequence of such moves, and HK_Γ is finite, then $HK_{\Gamma'}$ must be finite too. (as it is a quotient)

If Γ has no arrows, HK_Γ is finite iff Γ is a finite disjoint union of finite Dynking diagrams (simply laced $\implies$ ADE classification).

Hence, if HK_Γ is finite, then Γ is obtained by adding arrows to a finite disjoint union of ADE graphs. The converse is false, and apparently very involved.

Finiteness of Hecke-Kiselman monoids

Commutation $ab = ba$ implies $aba = bab = ab$, thus $HK_{\Gamma'}$ is a quotient of HK_Γ if Γ' if obtained from Γ by:

- removing an arrow
- making a side into an arrow
- removing a side

If Γ' is obtained from Γ by means of a finite sequence of such moves, and HK_Γ is finite, then $HK_{\Gamma'}$ must be finite too. (as it is a quotient)

If Γ has no arrows, HK_Γ is finite iff Γ is a finite disjoint union of finite Dynking diagrams (simply laced $\implies$ ADE classification).

Hence, if HK_Γ is finite, then Γ is obtained by adding arrows to a finite disjoint union of ADE graphs. The converse is false, and apparently very involved.

Finiteness of Hecke-Kiselman monoids

Commutation $ab = ba$ implies $aba = bab = ab$, thus $HK_{\Gamma'}$ is a quotient of HK_Γ if Γ' if obtained from Γ by:

- removing an arrow
- making a side into an arrow
- removing a side

If Γ' is obtained from Γ by means of a finite sequence of such moves, and HK_Γ is finite, then $HK_{\Gamma'}$ must be finite too. (as it is a quotient)

If Γ has no arrows, HK_Γ is finite iff Γ is a finite disjoint union of finite Dynking diagrams (simply laced $\implies$ ADE classification).

Hence, if HK_Γ is finite, then Γ is obtained by adding arrows to a finite disjoint union of ADE graphs. The converse is false, and apparently very involved.

Finiteness of Hecke-Kiselman monoids

Commutation $ab = ba$ implies $aba = bab = ab$, thus $HK_{\Gamma'}$ is a quotient of HK_Γ if Γ' if obtained from Γ by:

- removing an arrow
- making a side into an arrow
- removing a side

If Γ' is obtained from Γ by means of a finite sequence of such moves, and HK_Γ is finite, then $HK_{\Gamma'}$ must be finite too. (as it is a quotient)

If Γ has no arrows, HK_Γ is finite iff Γ is a finite disjoint union of finite Dynking diagrams (simply laced $\implies$ ADE classification).

Hence, if HK_Γ is finite, then Γ is obtained by adding arrows to a finite disjoint union of ADE graphs. The converse is false, and apparently very involved.

Finiteness of Hecke-Kiselman monoids

Commutation $ab = ba$ implies $aba = bab = ab$, thus $HK_{\Gamma'}$ is a quotient of HK_Γ if Γ' if obtained from Γ by:

- removing an arrow
- making a side into an arrow
- removing a side

If Γ' is obtained from Γ by means of a finite sequence of such moves, and HK_Γ is finite, then $HK_{\Gamma'}$ must be finite too. (as it is a quotient)

If Γ has no arrows, HK_Γ is finite iff Γ is a finite disjoint union of finite Dynking diagrams (simply laced $\implies$ ADE classification).

Hence, if HK_Γ is finite, then Γ is obtained by adding arrows to a finite disjoint union of ADE graphs. The converse is false, and apparently very involved.

Update systems on graphs

Let Γ be an oriented graph with at most one arrow between any two vertices. An update system on Γ is a choice of:

- a set S_i of local states for each vertex i;
- a local update function $f_i : \prod_{i \to j} S_j \to S_i$.

If $S = \prod_i S_i$ is the set of global states, each f_i induces a global update function $F_i : S \to S$ given by

$$(F_i(s))_k = \begin{cases} s_k & \text{if } k \neq i \\ f_i(s_j, i \to j) & \text{if } k = i \end{cases}$$

Every word in the vertices of Γ yields a corresponding composition of the F_i.

The image of the natural homomorphism $F(V) \to \mathrm{End}(S)$ is the dynamics monoid of the update system.

Update systems on graphs

Let Γ be an oriented graph with at most one arrow between any two vertices. An update system on Γ is a choice of:

- a set S_i of local states for each vertex i;
- a local update function $f_i : \prod_{i \to j} S_j \to S_i$.

If $S = \prod_i S_i$ is the set of global states, each f_i induces a global update function $F_i : S \to S$ given by

$$(F_i(s))_k = \begin{cases} s_k & \text{if } k \neq i \\ f_i(s_j, i \to j) & \text{if } k = i \end{cases}$$

Every word in the vertices of Γ yields a corresponding composition of the F_i.

The image of the natural homomorphism $F(V) \to \mathrm{End}(S)$ is the dynamics monoid of the update system.

Update systems on graphs

Let Γ be an oriented graph with at most one arrow between any two vertices. An update system on Γ is a choice of:

- a set S_i of local states for each vertex i;
- a local update function $f_i : \prod_{i \to j} S_j \to S_i$.

If $S = \prod_i S_i$ is the set of global states, each f_i induces a global update function $F_i : S \to S$ given by

$$(F_i(s))_k = \begin{cases} s_k & \text{if } k \neq i \\ f_i(s_j, i \to j) & \text{if } k = i \end{cases}$$

Every word in the vertices of Γ yields a corresponding composition of the F_i.

The image of the natural homomorphism $F(V) \to \mathrm{End}(S)$ is the dynamics monoid of the update system.

Update systems on graphs

Let Γ be an oriented graph with at most one arrow between any two vertices. An update system on Γ is a choice of:

- a set S_i of local states for each vertex i;
- a local update function $f_i : \prod_{i \to j} S_j \to S_i$.

If $S = \prod_i S_i$ is the set of global states, each f_i induces a global update function $F_i : S \to S$ given by

$$(F_i(s))_k = \begin{cases} s_k & \text{if } k \neq i \\ f_i(s_j, i \to j) & \text{if } k = i \end{cases}$$

Every word in the vertices of Γ yields a corresponding composition of the F_i.

The image of the natural homomorphism $F(V) \to \mathrm{End}(S)$ is the dynamics monoid of the update system.

Update systems on graphs

Let Γ be an oriented graph with at most one arrow between any two vertices. An update system on Γ is a choice of:

- a set S_i of local states for each vertex i;
- a local update function $f_i : \prod_{i \to j} S_j \to S_i$.

If $S = \prod_i S_i$ is the set of global states, each f_i induces a global update function $F_i : S \to S$ given by

$$(F_i(s))_k = \begin{cases} s_k & \text{if } k \neq i \\ f_i(s_j, i \to j) & \text{if } k = i \end{cases}$$

Every word in the vertices of Γ yields a corresponding composition of the F_i.

The image of the natural homomorphism $F(V) \to \mathrm{End}(S)$ is the dynamics monoid of the update system.

Update systems on graphs

Let Γ be an oriented graph with at most one arrow between any two vertices. An update system on Γ is a choice of:

- a set S_i of local states for each vertex i;
- a local update function $f_i : \prod_{i \to j} S_j \to S_i$.

If $S = \prod_i S_i$ is the set of global states, each f_i induces a global update function $F_i : S \to S$ given by

$$(F_i(s))_k = \begin{cases} s_k & \text{if } k \neq i \\ f_i(s_j, i \to j) & \text{if } k = i \end{cases}$$

Every word in the vertices of Γ yields a corresponding composition of the F_i.

The image of the natural homomorphism $F(V) \to \mathrm{End}(S)$ is the dynamics monoid of the update system.

Update systems on graphs

Let Γ be an oriented graph with at most one arrow between any two vertices. An update system on Γ is a choice of:

- a set S_i of local states for each vertex i;
- a local update function $f_i : \prod_{i\to j} S_j \to S_i$.

If $S = \prod_i S_i$ is the set of global states, each f_i induces a global update function $F_i : S \to S$ given by

$$(F_i(s))_k = \begin{cases} s_k & \text{if } k \neq i \\ f_i(s_j, i \to j) & \text{if } k = i \end{cases}$$

Every word in the vertices of Γ yields a corresponding composition of the F_i.

The image of the natural homomorphism $F(V) \to \mathrm{End}(S)$ is the dynamics monoid of the update system.

Update systems on graphs

Let Γ be an oriented graph with at most one arrow between any two vertices. An update system on Γ is a choice of:

- a set S_i of local states for each vertex i;
- a local update function $f_i : \prod_{i \to j} S_j \to S_i$.

If $S = \prod_i S_i$ is the set of global states, each f_i induces a global update function $F_i : S \to S$ given by

$$(F_i(s))_k = \begin{cases} s_k & \text{if } k \neq i \\ f_i(s_j, i \to j) & \text{if } k = i \end{cases}$$

Every word in the vertices of Γ yields a corresponding composition of the F_i.

The image of the natural homomorphism $F(V) \to \mathrm{End}(S)$ is the dynamics monoid of the update system.

A finiteness argument

- If Γ has no self-loops, then every F_i is idempotent. Henceforth: no self-loops!
- If i and j are not connected, then F_i and F_j commute.
- If $i \to j$, BUT $j \not\to i$, then $F_iF_jF_i = F_jF_iF_j = F_iF_j$.

If Γ has no cycles of length 1 or 2, then the natural homomorphism $F(V) \to \mathrm{End}(S)$ factor through HK_Γ.

Example: $\mathrm{Cyc}_n = \mathbb{Z}/n\mathbb{Z}$ with arrows $i \to i+1$; $S_i = \mathbb{Z}$, for all i; $f_i(s_{i+1}) = s_{i+1} + 1$. Then powers of $F_1F_2 \dots F_n$ are all distinct.

Consequently HK_{Cyc_n} is infinite. We learn that if HK_Γ is finite, it contains no oriented cycle.

A finiteness argument

- If Γ has no self-loops, then every F_i is idempotent. Henceforth: no self-loops!
- If i and j are not connected, then F_i and F_j commute.
- If $i \to j$, BUT $j \not\to i$, then $F_iF_jF_i = F_jF_iF_j = F_iF_j$.

If Γ has no cycles of length 1 or 2, then the natural homomorphism $F(V) \to \mathrm{End}(S)$ factor through HK_Γ.

Example: $\mathrm{Cyc}_n = \mathbb{Z}/n\mathbb{Z}$ with arrows $i \to i+1$; $S_i = \mathbb{Z}$, for all i; $f_i(s_{i+1}) = s_{i+1} + 1$. Then powers of $F_1F_2 \dots F_n$ are all distinct.

Consequently HK_{Cyc_n} is infinite. We learn that if HK_Γ is finite, it contains no oriented cycle.

A finiteness argument

- If Γ has no self-loops, then every F_i is idempotent. Henceforth: no self-loops!
- If i and j are not connected, then F_i and F_j commute.
- If $i \to j$, BUT $j \not\to i$, then $F_iF_jF_i = F_jF_iF_j = F_iF_j$.

If Γ has no cycles of length 1 or 2, then the natural homomorphism $F(V) \to \mathrm{End}(S)$ factor through HK_Γ.

Example: $\mathrm{Cyc}_n = \mathbb{Z}/n\mathbb{Z}$ with arrows $i \to i+1$; $S_i = \mathbb{Z}$, for all i; $f_i(s_{i+1}) = s_{i+1} + 1$. Then powers of $F_1F_2 \dots F_n$ are all distinct.

Consequently HK_{Cyc_n} is infinite. We learn that if HK_Γ is finite, it contains no oriented cycle.

A finiteness argument

- If Γ has no self-loops, then every F_i is idempotent. Henceforth: no self-loops!
- If i and j are not connected, then F_i and F_j commute.
- If $i \to j$, BUT $j \not\to i$, then $F_i F_j F_i = F_j F_i F_j = F_i F_j$.

If Γ has no cycles of length 1 or 2, then the natural homomorphism $F(V) \to \mathrm{End}(S)$ factor through HK_Γ.

Example: $\mathrm{Cyc}_n = \mathbb{Z}/n\mathbb{Z}$ with arrows $i \to i+1$; $S_i = \mathbb{Z}$, for all i; $f_i(s_{i+1}) = s_{i+1} + 1$. Then powers of $F_1 F_2 \dots F_n$ are all distinct.

Consequently HK_{Cyc_n} is infinite. We learn that if HK_Γ is finite, it contains no oriented cycle.

A finiteness argument

- If Γ has no self-loops, then every F_i is idempotent. Henceforth: no self-loops!
- If i and j are not connected, then F_i and F_j commute.
- If $i \to j$, BUT $j \not\to i$, then $F_i F_j F_i = F_j F_i F_j = F_i F_j$.

If Γ has no cycles of length 1 or 2, then the natural homomorphism $F(V) \to \mathrm{End}(S)$ factor through HK_Γ.

Example: $\mathrm{Cyc}_n = \mathbb{Z}/n\mathbb{Z}$ with arrows $i \to i+1$; $S_i = \mathbb{Z}$, for all i; $f_i(s_{i+1}) = s_{i+1} + 1$. Then powers of $F_1 F_2 \ldots F_n$ are all distinct.

Consequently HK_{Cyc_n} is infinite. We learn that if HK_Γ is finite, it contains no oriented cycle.

A finiteness argument

- If Γ has no self-loops, then every F_i is idempotent. Henceforth: no self-loops!
- If i and j are not connected, then F_i and F_j commute.
- If $i \to j$, BUT $j \not\to i$, then $F_i F_j F_i = F_j F_i F_j = F_i F_j$.

If Γ has no cycles of length 1 or 2, then the natural homomorphism $F(V) \to \mathrm{End}(S)$ factor through HK_Γ.

Example: $\mathrm{Cyc}_n = \mathbb{Z}/n\mathbb{Z}$ with arrows $i \to i+1$; $S_i = \mathbb{Z}$, for all i; $f_i(s_{i+1}) = s_{i+1} + 1$. Then powers of $F_1 F_2 \dots F_n$ are all distinct.

Consequently HK_{Cyc_n} is infinite. We learn that if HK_Γ is finite, it contains no oriented cycle.

A finiteness argument

- If Γ has no self-loops, then every F_i is idempotent. Henceforth: no self-loops!
- If i and j are not connected, then F_i and F_j commute.
- If $i \to j$, BUT $j \not\to i$, then $F_i F_j F_i = F_j F_i F_j = F_i F_j$.

If Γ has no cycles of length 1 or 2, then the natural homomorphism $F(V) \to \mathrm{End}(S)$ factor through HK_Γ.

Example: $\mathrm{Cyc}_n = \mathbb{Z}/n\mathbb{Z}$ with arrows $i \to i+1$; $S_i = \mathbb{Z}$, for all i; $f_i(s_{i+1}) = s_{i+1} + 1$. Then powers of $F_1 F_2 \dots F_n$ are all distinct.

Consequently HK_{Cyc_n} is infinite. We learn that if HK_Γ is finite, it contains no oriented cycle.

A finiteness argument

- If Γ has no self-loops, then every F_i is idempotent. Henceforth: no self-loops!
- If i and j are not connected, then F_i and F_j commute.
- If $i \to j$, BUT $j \not\to i$, then $F_i F_j F_i = F_j F_i F_j = F_i F_j$.

If Γ has no cycles of length 1 or 2, then the natural homomorphism $F(V) \to \mathrm{End}(S)$ factor through HK_Γ.

Example: $\mathrm{Cyc}_n = \mathbb{Z}/n\mathbb{Z}$ with arrows $i \to i+1$; $S_i = \mathbb{Z}$, for all i; $f_i(s_{i+1}) = s_{i+1} + 1$. Then powers of $F_1 F_2 \dots F_n$ are all distinct.

Consequently HK_{Cyc_n} is infinite. We learn that if HK_Γ is finite, it contains no oriented cycle.

A finiteness argument

- If Γ has no self-loops, then every F_i is idempotent. Henceforth: no self-loops!
- If i and j are not connected, then F_i and F_j commute.
- If $i \to j$, BUT $j \not\to i$, then $F_i F_j F_i = F_j F_i F_j = F_i F_j$.

If Γ has no cycles of length 1 or 2, then the natural homomorphism $F(V) \to \mathrm{End}(S)$ factor through HK_Γ.

Example: $\mathrm{Cyc}_n = \mathbb{Z}/n\mathbb{Z}$ with arrows $i \to i+1$; $S_i = \mathbb{Z}$, for all i; $f_i(s_{i+1}) = s_{i+1} + 1$. Then powers of $F_1 F_2 \ldots F_n$ are all distinct.

Consequently HK_{Cyc_n} is infinite. We learn that if HK_Γ is finite, it contains no oriented cycle.

A finiteness argument

- If Γ has no self-loops, then every F_i is idempotent. Henceforth: no self-loops!
- If i and j are not connected, then F_i and F_j commute.
- If $i \to j$, BUT $j \not\to i$, then $F_i F_j F_i = F_j F_i F_j = F_i F_j$.

If Γ has no cycles of length 1 or 2, then the natural homomorphism $F(V) \to \mathrm{End}(S)$ factor through HK_Γ.

Example: $\mathrm{Cyc}_n = \mathbb{Z}/n\mathbb{Z}$ with arrows $i \to i+1$; $S_i = \mathbb{Z}$, for all i; $f_i(s_{i+1}) = s_{i+1} + 1$. Then powers of $F_1 F_2 \dots F_n$ are all distinct.

Consequently HK_{Cyc_n} is infinite. We learn that if HK_Γ is finite, it contains no oriented cycle.

Finiteness of HK_Γ

What do we know of Γ if HK_Γ is finite?

- Γ has no oriented (or orientable) cycles
- If Γ has only unoriented edges, then it is a disjoint union of finite Dynkin graphs
- If Γ_n is the graph with vertices $v_1, \dots, v_n$ connected by arrows $v_i \to v_j$ iff $i < j$, then $HK_{\Gamma_n} = K_n$ is finite
- If Γ has only arrows, HK_Γ is finite iff Γ is acyclic (if it is acyclic, it is a quotient of some HK_{Γ_n}, which is finite)

The mixed case is complicated and exhibits not well understood interactions between ADE components and arrows between them: acyclicity and ADE components do not suffice to ensure finiteness.

Finiteness of HK_Γ

What do we know of Γ if HK_Γ is finite?

- Γ has no oriented (or orientable) cycles
- If Γ has only unoriented edges, then it is a disjoint union of finite Dynkin graphs
- If Γ_n is the graph with vertices $v_1, \dots, v_n$ connected by arrows $v_i \to v_j$ iff $i < j$, then $HK_{\Gamma_n} = K_n$ is finite
- If Γ has only arrows, HK_Γ is finite iff Γ is acyclic (if it is acyclic, it is a quotient of some HK_{Γ_n}, which is finite)

The mixed case is complicated and exhibits not well understood interactions between ADE components and arrows between them: acyclicity and ADE components do not suffice to ensure finiteness.

Finiteness of HK_Γ

What do we know of Γ if HK_Γ is finite?

- Γ has no oriented (or orientable) cycles
- If Γ has only unoriented edges, then it is a disjoint union of finite Dynkin graphs
- If Γ_n is the graph with vertices $v_1, \dots, v_n$ connected by arrows $v_i \to v_j$ iff $i < j$, then $HK_{\Gamma_n} = K_n$ is finite
- If Γ has only arrows, HK_Γ is finite iff Γ is acyclic (if it is acyclic, it is a quotient of some HK_{Γ_n}, which is finite)

The mixed case is complicated and exhibits not well understood interactions between ADE components and arrows between them: acyclicity and ADE components do not suffice to ensure finiteness.

Finiteness of HK_Γ

What do we know of Γ if HK_Γ is finite?

- Γ has no oriented (or orientable) cycles
- If Γ has only unoriented edges, then it is a disjoint union of finite Dynkin graphs
- If Γ_n is the graph with vertices $v_1, \ldots, v_n$ connected by arrows $v_i \to v_j$ iff $i < j$, then $HK_{\Gamma_n} = K_n$ is finite
- If Γ has only arrows, HK_Γ is finite iff Γ is acyclic (if it is acyclic, it is a quotient of some HK_{Γ_n}, which is finite)

The mixed case is complicated and exhibits not well understood interactions between ADE components and arrows between them: acyclicity and ADE components do not suffice to ensure finiteness.

Finiteness of HK_Γ

What do we know of Γ if HK_Γ is finite?

- Γ has no oriented (or orientable) cycles
- If Γ has only unoriented edges, then it is a disjoint union of finite Dynkin graphs
- If Γ_n is the graph with vertices $v_1, \ldots, v_n$ connected by arrows $v_i \to v_j$ iff $i < j$, then $HK_{\Gamma_n} = K_n$ is finite
- If Γ has only arrows, HK_Γ is finite iff Γ is acyclic (if it is acyclic, it is a quotient of some HK_{Γ_n}, which is finite)

The mixed case is complicated and exhibits not well understood interactions between ADE components and arrows between them: acyclicity and ADE components do not suffice to ensure finiteness.

Finiteness of HK_Γ

What do we know of Γ if HK_Γ is finite?

- Γ has no oriented (or orientable) cycles
- If Γ has only unoriented edges, then it is a disjoint union of finite Dynkin graphs
- If Γ_n is the graph with vertices $v_1, \ldots, v_n$ connected by arrows $v_i \to v_j$ iff $i < j$, then $HK_{\Gamma_n} = K_n$ is finite
- If Γ has only arrows, HK_Γ is finite iff Γ is acyclic (if it is acyclic, it is a quotient of some HK_{Γ_n}, which is finite)

The mixed case is complicated and exhibits not well understood interactions between ADE components and arrows between them: acyclicity and ADE components do not suffice to ensure finiteness.

Finiteness of HK_Γ

What do we know of Γ if HK_Γ is finite?

- Γ has no oriented (or orientable) cycles
- If Γ has only unoriented edges, then it is a disjoint union of finite Dynkin graphs
- If Γ_n is the graph with vertices $v_1, \dots, v_n$ connected by arrows $v_i \to v_j$ iff $i < j$, then $HK_{\Gamma_n} = K_n$ is finite
- If Γ has only arrows, HK_Γ is finite iff Γ is acyclic (if it is acyclic, it is a quotient of some HK_{Γ_n}, which is finite)

The mixed case is complicated and exhibits not well understood interactions between ADE components and arrows between them: acyclicity and ADE components do not suffice to ensure finiteness.

Finiteness of HK_Γ

What do we know of Γ if HK_Γ is finite?

- Γ has no oriented (or orientable) cycles
- If Γ has only unoriented edges, then it is a disjoint union of finite Dynkin graphs
- If Γ_n is the graph with vertices $v_1, \dots, v_n$ connected by arrows $v_i \to v_j$ iff $i < j$, then $HK_{\Gamma_n} = K_n$ is finite
- If Γ has only arrows, HK_Γ is finite iff Γ is acyclic (if it is acyclic, it is a quotient of some HK_{Γ_n}, which is finite)

The mixed case is complicated and exhibits not well understood interactions between ADE components and arrows between them: acyclicity and ADE components do not suffice to ensure finiteness.

Finiteness of HK_Γ

What do we know of Γ if HK_Γ is finite?

- Γ has no oriented (or orientable) cycles
- If Γ has only unoriented edges, then it is a disjoint union of finite Dynkin graphs
- If Γ_n is the graph with vertices $v_1, \dots, v_n$ connected by arrows $v_i \to v_j$ iff $i < j$, then $HK_{\Gamma_n} = K_n$ is finite
- If Γ has only arrows, HK_Γ is finite iff Γ is acyclic (if it is acyclic, it is a quotient of some HK_{Γ_n}, which is finite)

The mixed case is complicated and exhibits not well understood interactions between ADE components and arrows between them: acyclicity and ADE components do not suffice to ensure finiteness.

Finiteness of Hecke-Kiselman monoids

There is a unique acyclic digraph on four vertices with ADE connected components which yields an infinite Hecke-Kiselman monoid:

$$\begin{tikzcd} a \arrow[r] \arrow[d, dash] \arrow[dr] & c \arrow[d, dash] \\ b \arrow[r] \arrow[ur] & d. \end{tikzcd}$$

This is proved by making it act "transitively" on an infinite set.

joint with Aragona 2013

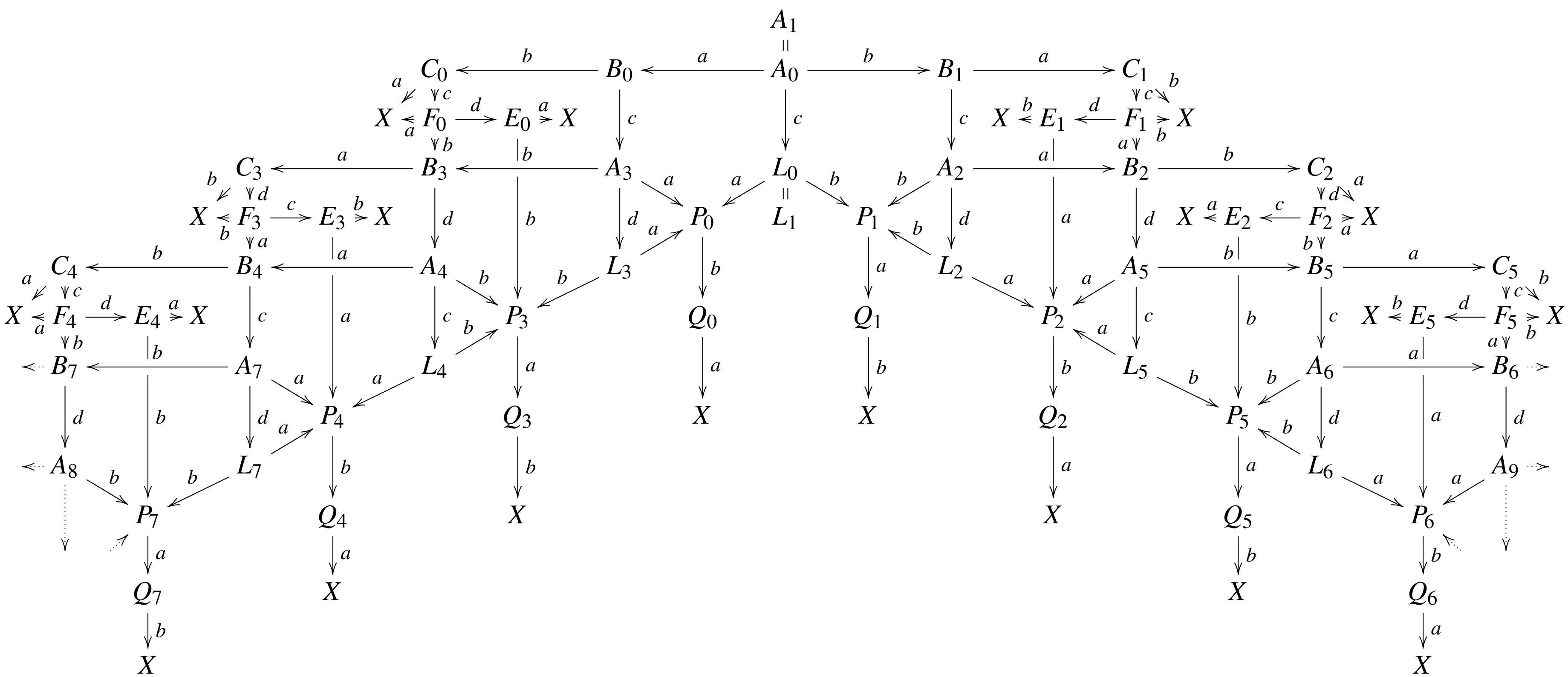


FIGURE 1

Is universal dynamics possible?

- Is it possible to set up an update system on the graph Γ so that $HK_\Gamma \to \mathrm{End}(S)$ be injective?

We already know that maps F_i satisfy the Hecke-Kiselman relations, but there might be further relations we failed to spot so far.

In order to show there are no further universal relation is to set up an update system on Γ in which the F_i generate a monoid isomorphic to HK_Γ.

Idea: set up local functions that (combinatorially?) recover a word (= update sequence) inducing the information found on outward vertices.

Is universal dynamics possible?

- Is it possible to set up an update system on the graph Γ so that $HK_\Gamma \to \mathrm{End}(S)$ be injective?

We already know that maps F_i satisfy the Hecke-Kiselman relations, but there might be further relations we failed to spot so far.

In order to show there are no further universal relation is to set up an update system on Γ in which the F_i generate a monoid isomorphic to HK_Γ.

Idea: set up local functions that (combinatorially?) recover a word (= update sequence) inducing the information found on outward vertices.

Is universal dynamics possible?

- Is it possible to set up an update system on the graph Γ so that $HK_\Gamma \to \mathrm{End}(S)$ be injective?

We already know that maps F_i satisfy the Hecke-Kiselman relations, but there might be further relations we failed to spot so far.

In order to show there are no further universal relation is to set up an update system on Γ in which the F_i generate a monoid isomorphic to HK_Γ.

Idea: set up local functions that (combinatorially?) recover a word (= update sequence) inducing the information found on outward vertices.

Is universal dynamics possible?

- Is it possible to set up an update system on the graph Γ so that $HK_\Gamma \to \mathrm{End}(S)$ be injective?

We already know that maps F_i satisfy the Hecke-Kiselman relations, but there might be further relations we failed to spot so far.

In order to show there are no further universal relation is to set up an update system on Γ in which the F_i generate a monoid isomorphic to HK_Γ.

Idea: set up local functions that (combinatorially?) recover a word (= update sequence) inducing the information found on outward vertices.

Kiselman case

The most convenient case to treat is when $\Gamma = \Gamma_n$ is the complete (acyclic, ordered) graph. It is convenient since:

- one has an explicit characterization of reduced words in $HK_{\Gamma_n} = K_n$;
- simplifications from non reduced to reduced words are always monotone: one may simplify any given word to its reduced form by a sequence of length-reducing steps and. . .
- . . . every such sequence ends on the same reduced word.

Kiselman case

The most convenient case to treat is when $\Gamma = \Gamma_n$ is the complete (acyclic, ordered) graph. It is convenient since:

- one has an explicit characterization of reduced words in $HK_{\Gamma_n} = K_n$;
- simplifications from non reduced to reduced words are always monotone: one may simplify any given word to its reduced form by a sequence of length-reducing steps and. . .
- . . . every such sequence ends on the same reduced word.

Kiselman case

The most convenient case to treat is when $\Gamma = \Gamma_n$ is the complete (acyclic, ordered) graph. It is convenient since:

- one has an explicit characterization of reduced words in $HK_{\Gamma_n} = K_n$;
- simplifications from non reduced to reduced words are always monotone: one may simplify any given word to its reduced form by a sequence of length-reducing steps and...
- ...every such sequence ends on the same reduced word.

Kiselman case

The most convenient case to treat is when $\Gamma = \Gamma_n$ is the complete (acyclic, ordered) graph. It is convenient since:

- one has an explicit characterization of reduced words in $HK_{\Gamma_n} = K_n$;
- simplifications from non reduced to reduced words are always monotone: one may simplify any given word to its reduced form by a sequence of length-reducing steps and...
- ...every such sequence ends on the same reduced word.

Kiselman case

The most convenient case to treat is when $\Gamma = \Gamma_n$ is the complete (acyclic, ordered) graph. It is convenient since:

- one has an explicit characterization of reduced words in $HK_{\Gamma_n} = K_n$;
- simplifications from non reduced to reduced words are always monotone: one may simplify any given word to its reduced form by a sequence of length-reducing steps and…
- …every such sequence ends on the same reduced word.

A linking operation

If $u, v \in F(A)$ are words in the alphabet A, we define $[u, v]$ to be the shortest word that

- has v as a suffix
- admits u as a subword

How to compute $[u, v]$:

- Factor $u = u_1 u_2$ so that u_2 is longest suffix of u which is a subword of v.
- Then $[u, v] = u_1 v$.

E.g.: $[abcab, babc] = abcb\underline{ab}c$.

The product $[\ ,\ ]$ is neither commutative nor associative. It seems to lack good properties, but solves the universal dynamics problem for the graph Γ_n.

A linking operation

If $u, v \in F(A)$ are words in the alphabet A, we define $[u, v]$ to be the shortest word that

- has v as a suffix
- admits u as a subword

How to compute $[u, v]$:

- Factor $u = u_1 u_2$ so that u_2 is longest suffix of u which is a subword of v.
- Then $[u, v] = u_1 v$.

E.g.: $[abcab, babc] = abc\underline{ba}bc$.

The product $[\ ,\ ]$ is neither commutative nor associative. It seems to lack good properties, but solves the universal dynamics problem for the graph Γ_n.

A linking operation

If $u, v \in F(A)$ are words in the alphabet A, we define $[u, v]$ to be the shortest word that

- has v as a suffix
- admits u as a subword

How to compute $[u, v]$:

- Factor $u = u_1 u_2$ so that u_2 is longest suffix of u which is a subword of v.
- Then $[u, v] = u_1 v$.

E.g.: $[abcab, babc] = abcb\underline{ab}c$.

The product $[\ ,\ ]$ is neither commutative nor associative. It seems to lack good properties, but solves the universal dynamics problem for the graph Γ_n.

A linking operation

If $u, v \in F(A)$ are words in the alphabet A, we define $[u, v]$ to be the shortest word that

- has v as a suffix
- admits u as a subword

How to compute $[u, v]$:

- Factor $u = u_1 u_2$ so that u_2 is longest suffix of u which is a subword of v.
- Then $[u, v] = u_1 v$.

E.g.: $[abcab, babc] = abc\,b\underline{ab}c$.

The product $[\ ,\ ]$ is neither commutative nor associative. It seems to lack good properties, but solves the universal dynamics problem for the graph Γ_n.

A linking operation

If $u, v \in F(A)$ are words in the alphabet A, we define $[u, v]$ to be the shortest word that

- has v as a suffix
- admits u as a subword

How to compute $[u, v]$:

- Factor $u = u_1 u_2$ so that u_2 is longest suffix of u which is a subword of v.
- Then $[u, v] = u_1 v$.

E.g.: $[abcab, babc] = abcb\underline{ab}c$.

The product $[\ ,\ ]$ is neither commutative nor associative. It seems to lack good properties, but solves the universal dynamics problem for the graph Γ_n.

A universal update system on Γ_n

On the graph Γ_n set $S_i = F(a_i, \ldots, a_n)$ and define

$$f_i(s_{i+1}, \ldots, s_n) = a_i[s_n, \ldots [s_{i+3}, [s_{i+2}, s_{i+1}]] \ldots].$$

Teorema

Let $w \in F(a_1, \ldots, a_n)$. If $F_w(1, 1, \ldots, 1) = (s_1, s_2, \ldots, s_n)$, then $[s_n, \ldots [s_3, [s_2, s_1]] \ldots]$ is the (unique) reduced expression of w in $HK_{\Gamma_n} = K_n$.

The dynamical complexity of K_n is captured by symbolic-combinatorial properties of the linking operation.

Warning! One obtains a reduced expression WHEN the state $(s_1, \ldots, s_n)$ is reachable from (1, 1, ..., 1), but not in general.

joint with Collina 2015

A universal update system on Γ_n

On the graph Γ_n set $S_i = F(a_i, \ldots, a_n)$ and define

$$f_i(s_{i+1}, \ldots, s_n) = a_i[s_n, \ldots [s_{i+3}, [s_{i+2}, s_{i+1}]] \ldots].$$

Teorema

Let $w \in F(a_1, \ldots, a_n)$. If $F_w(1, 1, \ldots, 1) = (s_1, s_2, \ldots, s_n)$, then $[s_n, \ldots [s_3, [s_2, s_1]] \ldots]$ is the (unique) reduced expression of w in $HK_{\Gamma_n} = K_n$.

The dynamical complexity of K_n is captured by symbolic-combinatorial properties of the linking operation.

Warning! One obtains a reduced expression WHEN the state $(s_1, \ldots, s_n)$ is reachable from (1, 1, ..., 1), but not in general.

joint with Collina 2015

A universal update system on Γ_n

On the graph Γ_n set $S_i = F(a_i, \ldots, a_n)$ and define

$$f_i(s_{i+1}, \ldots, s_n) = a_i[s_n, \ldots [s_{i+3}, [s_{i+2}, s_{i+1}]] \ldots].$$

Teorema

Let $w \in F(a_1, \ldots, a_n)$. If $F_w(1, 1, \ldots, 1) = (s_1, s_2, \ldots, s_n)$, then $[s_n, \ldots [s_3, [s_2, s_1]] \ldots]$ is the (unique) reduced expression of w in $HK_{\Gamma_n} = K_n$.

The dynamical complexity of K_n is captured by symbolic-combinatorial properties of the linking operation.

Warning! One obtains a reduced expression WHEN the state $(s_1, \ldots, s_n)$ is reachable from (1, 1, ..., 1), but not in general.

joint with Collina 2015

A universal update system on Γ_n

On the graph Γ_n set $S_i = F(a_i, \ldots, a_n)$ and define

$$f_i(s_{i+1}, \ldots, s_n) = a_i[s_n, \ldots [s_{i+3}, [s_{i+2}, s_{i+1}]] \ldots].$$

Teorema

Let $w \in F(a_1, \ldots, a_n)$. If $F_w(1, 1, \ldots, 1) = (s_1, s_2, \ldots, s_n)$, then $[s_n, \ldots [s_3, [s_2, s_1]] \ldots]$ is the (unique) reduced expression of w in $HK_{\Gamma_n} = K_n$.

The dynamical complexity of K_n is captured by symbolic-combinatorial properties of the linking operation.

Warning! One obtains a reduced expression WHEN the state $(s_1, \ldots, s_n)$ is reachable from (1, 1, ..., 1), but not in general.

joint with Collina 2015

A universal update system on Γ_n

On the graph Γ_n set $S_i = F(a_i, \ldots, a_n)$ and define

$$f_i(s_{i+1}, \ldots, s_n) = a_i[s_n, \ldots [s_{i+3}, [s_{i+2}, s_{i+1}]] \ldots].$$

Teorema

Let $w \in F(a_1, \ldots, a_n)$. If $F_w(1, 1, \ldots, 1) = (s_1, s_2, \ldots, s_n)$, then $[s_n, \ldots [s_3, [s_2, s_1]] \ldots]$ is the (unique) reduced expression of w in $HK_{\Gamma_n} = K_n$.

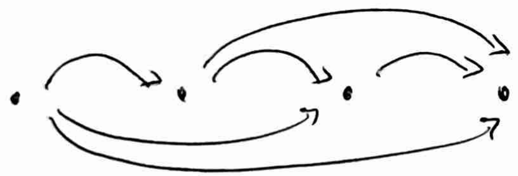
The dynamical complexity of K_n is captured by symbolic-combinatorial properties of the linking operation.

Warning! One obtains a reduced expression WHEN the state $(s_1, \ldots, s_n)$ is reachable from (1, 1, . . . , 1), but not in general.

joint with Collina 2015

EXAMPLE: Kiselman semigroup K_4

corresponds to graph



We denote both vertices and update function with a single letter

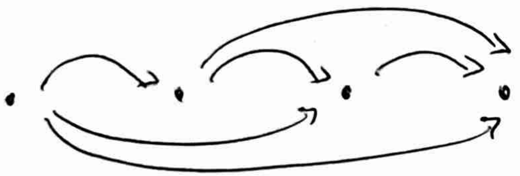
a b c d

At the beginning each (local) state is the empty word

$*$ $*$ $*$ $*$

we want to perform the word bcabdc from right to left.

EXAMPLE: Kiselman semigroup K_4 corresponds to graph



We denote both vertices and update function with a single letter

a	b	c	d
•	•	•	•

At the beginning each (local) state is the empty word

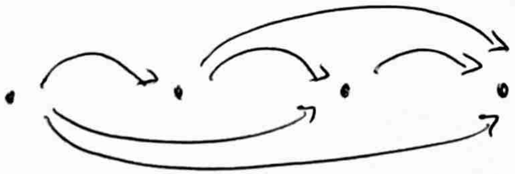
•	•	•	•
*	*	*	*

we want to perform the word bcabdc from right to left.

a	b	c	d	
*	*	*	*	
				$\downarrow$ c
*	*	c	*	

EXAMPLE: Kiselman semigroup K_4

corresponds to graph



We denote both vertices and update function with a single letter

$$\begin{array}{cccc} a & b & c & d \\ \cdot & \cdot & \cdot & \cdot \end{array}$$

At the beginning each (local) state is the empty word

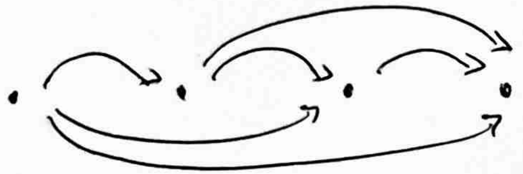
$$\begin{array}{cccc} \cdot & \cdot & \cdot & \cdot \\ * & * & * & * \end{array}$$

we want to perform the word bcabdc from right to left.

$$\begin{array}{cccc|l} a & b & c & d & \\ * & * & * & * & \\ & & & & \downarrow c \\ * & * & c & * & \\ & & & & \downarrow d \\ * & * & c & d & \end{array}$$

EXAMPLE: Kiselman semigroup K_4

corresponds to graph



We denote both vertices and update function with a single letter

a	b	c	d
•	•	•	•

At the beginning each (local) state is the empty word

•	•	•	•
*	*	*	*

we want to perform the word bc abde from right to left.

a	b	c	d	
*	*	*	*	
				$\downarrow$ c
*	*	c	*	
				$\downarrow$ d
*	*	c	d	
				$\downarrow$ b
*	bde	c	d	

EXAMPLE: Kiselman semigroup K_4

corresponds to graph

We denote both vertices and update function with a single letter

a	b	c	d
$\cdot$	$\cdot$	$\cdot$	$\cdot$

At the beginning each (local) state is the empty word

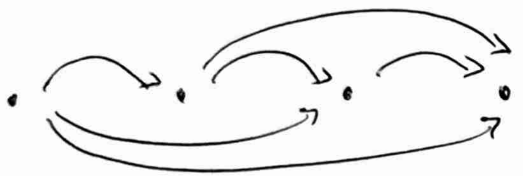
$\cdot$	$\cdot$	$\cdot$	$\cdot$
*	*	*	*

we want to perform the word

bcabdc from right to left.

a	b	c	d	
*	*	*	*	
				c
*	*	c	*	
				d
*	*	c	d	
				b
*	bdc	c	d	
				a
abdc	bdc	c	d	

EXAMPLE: Kiselman semigroup K_4 corresponds to graph



We denote both vertices and update function with a single letter

a b c d

At the beginning each (local) state is the empty word

a	b	c	d	
*	*	*	*	
				c
*	*	c	*	
				d
*	*	c	d	
				b
*	bdc	c	d	
				a
abdc	bdc	c	d	
				c
abdc	bdc	cd	d	
				b
abdc	bcd	cd	d	

we want to perform the word bcabdc from right to left.

(d, cd]
= cd

A universal update system on Γ_n

We have reached the state

$$abdc \qquad bcd \qquad cd \qquad d$$

which, according to the theorem, is induced by the word

$$[d,[cd,[bcd,abdc]]] = [d,[cd,bcab\underline{d}c]] = [d,b\underline{c}ab\underline{d}c] = bcab\underline{d}c,$$

as we indeed computed.

A universal update system on Γ_n

We have reached the state

$$abdc \qquad bcd \qquad cd \qquad d$$

which, according to the theorem, is induced by the word

$$[d, [cd, [bcd, abdc]]] = [d, [cd, bcab\underline{d}c]] = [d, b\underline{c}ab\underline{d}c] = bcab\underline{d}c,$$

as we indeed computed.

A universal update system on Γ_n

We have reached the state

$$abdc \qquad bcd \qquad cd \qquad d$$

which, according to the theorem, is induced by the word

$$[d, [cd, [bcd, abdc]]] = [d, [cd, bcab\underline{d}c]] = [d, b\underline{c}ab\underline{d}c] = bcab\underline{d}c,$$

as we indeed computed.

A universal update system on Γ_n

We have reached the state

$$abdc \qquad bcd \qquad cd \qquad d$$

which, according to the theorem, is induced by the word

$$[d, [cd, [bcd, abdc]]] = [d, [cd, bcab\underline{d}c]] = [d, b\underline{c}ab\underline{d}c] = bcab\underline{d}c,$$

as we indeed computed.

A universal update system on Γ_n

We have reached the state

$$abdc \qquad bcd \qquad cd \qquad d$$

which, according to the theorem, is induced by the word

$$[d, [cd, [bcd, abdc]]] = [d, [cd, bcab\underline{d}c]] = [d, b\underline{c}ab\underline{d}c] = bcab\underline{d}c,$$

as we indeed computed.

A universal update system on Γ_n

We have reached the state

$$abdc \qquad bcd \qquad cd \qquad d$$

which, according to the theorem, is induced by the word

$$[d,[cd,[bcd,abdc]]] = [d,[cd,bc\,ab\underline{d}c]] = [d,b\underline{c}ab\underline{d}c] = bcab\underline{d}c,$$

as we indeed computed.

What happens for other choices of Γ?

The linking operation [,] works for a few other choices of Γ (e.g.: equioriented $A_n \rightsquigarrow$ Catalan monoid) but not always. For general choices of Γ one needs to take time priority of local updates into account.

Good news: Mazorchuk's proof generalizes nicely. One may show that all simplifications from any given word to a reduced expression are monotone. However, reduced expression is not unique due to possibility to commute letters but this is the only form of non-uniqueness and can be dealt with by taking the most lexicographically convenient reduced expression.

Idea: as reduced expressions are not unique, suffix means suffix in some reduced expression. Same with subword. The linking operation needs to be redefined to account for these new features. New definition is ugly…

…but works (experimentally) in all cases (all Γ's with at most 8 vertices and a few other scattered examples).

What happens for other choices of Γ?

The linking operation [,] works for a few other choices of Γ (e.g.: equioriented $A_n \rightsquigarrow$ Catalan monoid) but not always. For general choices of Γ one needs to take time priority of local updates into account.

Good news: Mazorchuk's proof generalizes nicely. One may show that all simplifications from any given word to a reduced expression are monotone. However, reduced expression is not unique due to possibility to commute letters but this is the only form of non-uniqueness and can be dealt with by taking the most lexicographically convenient reduced expression.

Idea: as reduced expressions are not unique, suffix means suffix in some reduced expression. Same with subword. The linking operation needs to be redefined to account for these new features. New definition is ugly. . .

. . . but works (experimentally) in all cases (all Γ's with at most 8 vertices and a few other scattered examples).

What happens for other choices of Γ?

The linking operation [,] works for a few other choices of Γ (e.g.: equioriented $A_n \rightsquigarrow$ Catalan monoid) but not always. For general choices of Γ one needs to take time priority of local updates into account.

Good news: Mazorchuk's proof generalizes nicely. One may show that all simplifications from any given word to a reduced expression are monotone. However, reduced expression is not unique due to possibility to commute letters but this is the only form of non-uniqueness and can be dealt with by taking the most lexicographically convenient reduced expression.

Idea: as reduced expressions are not unique, suffix means suffix in some reduced expression. Same with subword. The linking operation needs to be redefined to account for these new features. New definition is ugly. . .

. . . but works (experimentally) in all cases (all Γ's with at most 8 vertices and a few other scattered examples).

What happens for other choices of Γ?

The linking operation [,] works for a few other choices of Γ (e.g.: equioriented $A_n \rightsquigarrow$ Catalan monoid) but not always. For general choices of Γ one needs to take time priority of local updates into account.

Good news: Mazorchuk's proof generalizes nicely. One may show that all simplifications from any given word to a reduced expression are monotone. However, reduced expression is not unique due to possibility to commute letters but this is the only form of non-uniqueness and can be dealt with by taking the most lexicographically convenient reduced expression.

Idea: as reduced expressions are not unique, suffix means suffix in some reduced expression. Same with subword. The linking operation needs to be redefined to account for these new features. New definition is ugly...

...but works (experimentally) in all cases (all Γ's with at most 8 vertices and a few other scattered examples).

What happens for other choices of Γ?

The linking operation [,] works for a few other choices of Γ (e.g.: equioriented $A_n \rightsquigarrow$ Catalan monoid) but not always. For general choices of Γ one needs to take time priority of local updates into account.

Good news: Mazorchuk's proof generalizes nicely. One may show that all simplifications from any given word to a reduced expression are monotone. However, reduced expression is not unique due to possibility to commute letters but this is the only form of non-uniqueness and can be dealt with by taking the most lexicographically convenient reduced expression.

Idea: as reduced expressions are not unique, suffix means suffix in some reduced expression. Same with subword. The linking operation needs to be redefined to account for these new features. New definition is ugly...

...but works (experimentally) in all cases (all Γ's with at most 8 vertices and a few other scattered examples).

What happens for other choices of Γ?

The linking operation [,] works for a few other choices of Γ (e.g.: equioriented $A_n \rightsquigarrow$ Catalan monoid) but not always. For general choices of Γ one needs to take time priority of local updates into account.

Good news: Mazorchuk's proof generalizes nicely. One may show that all simplifications from any given word to a reduced expression are monotone. However, reduced expression is not unique due to possibility to commute letters but this is the only form of non-uniqueness and can be dealt with by taking the most lexicographically convenient reduced expression.

Idea: as reduced expressions are not unique, suffix means suffix in some reduced expression. Same with subword. The linking operation needs to be redefined to account for these new features. New definition is ugly...

...but works (experimentally) in all cases (all Γ's with at most 8 vertices and a few other scattered examples).

What happens for other choices of Γ?

The linking operation [,] works for a few other choices of Γ (e.g.: equioriented $A_n \rightsquigarrow$ Catalan monoid) but not always. For general choices of Γ one needs to take time priority of local updates into account.

Good news: Mazorchuk's proof generalizes nicely. One may show that all simplifications from any given word to a reduced expression are monotone. However, reduced expression is not unique due to possibility to commute letters but this is the only form of non-uniqueness and can be dealt with by taking the most lexicographically convenient reduced expression.

Idea: as reduced expressions are not unique, suffix means suffix in some reduced expression. Same with subword. The linking operation needs to be redefined to account for these new features. New definition is ugly...

...but works (experimentally) in all cases (all Γ's with at most 8 vertices and a few other scattered examples).

Generalized linking operation

Set $[u, \star] = [\star, u] = u$.

Let $u = a_1 a_2 \dots a_n, v = b_1 b_2 \dots b_m$ be non empty words in the alphabet V, where V is the set of vertices of a finite acyclic oriented graph.

Choose (if there exist some) the rightmost letter b_i of v such that

- b_i commutes with all $b_j, j > i$;
- no letter in the longest suffix of u not containing b_i has an arrow pointing to b_i.

Denote by $\overline{v}$ the word obtained by removing the rightmost occurence of b_i from v (and similarly with u). Then

- if u contains b_i, and b_i commutes with all letters in the longest suffix of u not containing b_i, then set $[u, v] = [\overline{u}, \overline{v}]b_i$;
- otherwise, set $[u, v] = [u, \overline{v}]b_i$.

Generalized linking operation

Set $[u, \star] = [\star, u] = u$.

Let $u = a_1 a_2 \ldots a_n, v = b_1 b_2 \ldots b_m$ be non empty words in the alphabet V, where V is the set of vertices of a finite acyclic oriented graph.

Choose (if there exist some) the rightmost letter b_i of v such that

- b_i commutes with all $b_j, j > i$;
- no letter in the longest suffix of u not containing b_i has an arrow pointing to b_i.

Denote by $\overline{v}$ the word obtained by removing the rightmost occurence of b_i from v (and similarly with u). Then

- if u contains b_i, and b_i commutes with all letters in the longest suffix of u not containing b_i, then set $[u, v] = [\overline{u}, \overline{v}]b_i$;
- otherwise, set $[u, v] = [u, \overline{v}]b_i$.

Generalized linking operation

Set $[u, \star] = [\star, u] = u$.

Let $u = a_1 a_2 \dots a_n, v = b_1 b_2 \dots b_m$ be non empty words in the alphabet V, where V is the set of vertices of a finite acyclic oriented graph.

Choose (if there exist some) the rightmost letter b_i of v such that

- b_i commutes with all $b_j, j > i$;
- no letter in the longest suffix of u not containing b_i has an arrow pointing to b_i.

Denote by $\overline{v}$ the word obtained by removing the rightmost occurence of b_i from v (and similarly with u). Then

- if u contains b_i, and b_i commutes with all letters in the longest suffix of u not containing b_i, then set $[u, v] = [\overline{u}, \overline{v}]b_i$;
- otherwise, set $[u, v] = [u, \overline{v}]b_i$.

Generalized linking operation

Set $[u, \star] = [\star, u] = u$.

Let $u = a_1 a_2 \ldots a_n, v = b_1 b_2 \ldots b_m$ be non empty words in the alphabet V, where V is the set of vertices of a finite acyclic oriented graph.

Choose (if there exist some) the rightmost letter b_i of v such that

- b_i commutes with all $b_j, j > i$;
- no letter in the longest suffix of u not containing b_i has an arrow pointing to b_i.

Denote by $\overline{v}$ the word obtained by removing the rightmost occurence of b_i from v (and similarly with u). Then

- if u contains b_i, and b_i commutes with all letters in the longest suffix of u not containing b_i, then set $[u, v] = [\overline{u}, \overline{v}]b_i$;
- otherwise, set $[u, v] = [u, \overline{v}]b_i$.

Generalized linking operation

Set $[u, \star] = [\star, u] = u$.

Let $u = a_1 a_2 \ldots a_n, v = b_1 b_2 \ldots b_m$ be non empty words in the alphabet V, where V is the set of vertices of a finite acyclic oriented graph.

Choose (if there exist some) the rightmost letter b_i of v such that

- b_i commutes with all $b_j, j > i$;
- no letter in the longest suffix of u not containing b_i has an arrow pointing to b_i.

Denote by $\overline{v}$ the word obtained by removing the rightmost occurence of b_i from v (and similarly with u). Then

- if u contains b_i, and b_i commutes with all letters in the longest suffix of u not containing b_i, then set $[u, v] = [\overline{u}, \overline{v}] b_i$;
- otherwise, set $[u, v] = [u, \overline{v}] b_i$.

Generalized linking operation

Set $[u, \star] = [\star, u] = u$.

Let $u = a_1 a_2 \dots a_n, v = b_1 b_2 \dots b_m$ be non empty words in the alphabet V, where V is the set of vertices of a finite acyclic oriented graph.

Choose (if there exist some) the rightmost letter b_i of v such that

- b_i commutes with all $b_j, j > i$;
- no letter in the longest suffix of u not containing b_i has an arrow pointing to b_i.

Denote by $\overline{v}$ the word obtained by removing the rightmost occurence of b_i from v (and similarly with u). Then

- if u contains b_i, and b_i commutes with all letters in the longest suffix of u not containing b_i, then set $[u, v] = [\overline{u}, \overline{v}]b_i$;
- otherwise, set $[u, v] = [u, \overline{v}]b_i$.

Generalized linking operation

Set $[u, \star] = [\star, u] = u$.

Let $u = a_1 a_2 \dots a_n, v = b_1 b_2 \dots b_m$ be non empty words in the alphabet V, where V is the set of vertices of a finite acyclic oriented graph.

Choose (if there exist some) the rightmost letter b_i of v such that

- b_i commutes with all $b_j, j > i$;
- no letter in the longest suffix of u not containing b_i has an arrow pointing to b_i.

Denote by $\overline{v}$ the word obtained by removing the rightmost occurence of b_i from v (and similarly with u). Then

- if u contains b_i, and b_i commutes with all letters in the longest suffix of u not containing b_i, then set $[u, v] = [\overline{u}, \overline{v}]b_i$;
- otherwise, set $[u, v] = [u, \overline{v}]b_i$.

Generalized linking operation

Set $[u, \star] = [\star, u] = u$.

Let $u = a_1 a_2 \dots a_n, v = b_1 b_2 \dots b_m$ be non empty words in the alphabet V, where V is the set of vertices of a finite acyclic oriented graph.

Choose (if there exist some) the rightmost letter b_i of v such that

- b_i commutes with all $b_j, j > i$;
- no letter in the longest suffix of u not containing b_i has an arrow pointing to b_i.

Denote by $\overline{v}$ the word obtained by removing the rightmost occurence of b_i from v (and similarly with u). Then

- if u contains b_i, and b_i commutes with all letters in the longest suffix of u not containing b_i, then set $[u, v] = [\overline{u}, \overline{v}]b_i$;
- otherwise, set $[u, v] = [u, \overline{v}]b_i$.

Generalized linking operation

If there exists no such b_i, then choose the rightmost letter a_i of u such that

- a_i commutes with all $a_j, j > i$;
- no letter in the longest suffix of v not containing a_i has an arrow pointing to a_i.

Denote by $\overline{u}$ the word obtained by removing the rightmost occurence of a_i from v (and similarly with u). Then

- if v contains a_i, and a_i commutes with all letters in the longest suffix of v not containing a_i, then set $[u, v] = [\overline{u}, \overline{v}]a_i$;
- otherwise, set $[u, v] = [\overline{u}, v]a_i$.

If there exists no such a_i, then set $[u, v] = \star$.

Generalized linking operation

If there exists no such b_i, then choose the rightmost letter a_i of u such that

- a_i commutes with all $a_j, j > i$;
- no letter in the longest suffix of v not containing a_i has an arrow pointing to a_i.

Denote by $\overline{u}$ the word obtained by removing the rightmost occurence of a_i from v (and similarly with u). Then

- if v contains a_i, and a_i commutes with all letters in the longest suffix of v not containing a_i, then set $[u, v] = [\overline{u}, \overline{v}]a_i$;
- otherwise, set $[u, v] = [\overline{u}, v]a_i$.

If there exists no such a_i, then set $[u, v] = \star$.

Generalized linking operation

If there exists no such b_i, then choose the rightmost letter a_i of u such that

- a_i commutes with all $a_j, j > i$;
- no letter in the longest suffix of v not containing a_i has an arrow pointing to a_i.

Denote by $\overline{u}$ the word obtained by removing the rightmost occurence of a_i from v (and similarly with u). Then

- if v contains a_i, and a_i commutes with all letters in the longest suffix of v not containing a_i, then set $[u, v] = [\overline{u}, \overline{v}]a_i$;
- otherwise, set $[u, v] = [\overline{u}, v]a_i$.

If there exists no such a_i, then set $[u, v] = \star$.

Questions

- Can one find a canonical combinatorial action of Hecke-Kiselman monoids on something?
- Can one set up a universal update system also on oriented graphs with cycles?
- Does Coxeter combinatorics play a role in this setting?
- Is there a way to recursively compute the order of Hecke-Kiselman monoids as in the Coxeter setting?
- Can one prov(id)e a characterization of digraphs inducing finite Hecke-Kiselman monoids?

Questions

- Can one find a canonical combinatorial action of Hecke-Kiselman monoids on something?
- Can one set up a universal update system also on oriented graphs with cycles?
- Does Coxeter combinatorics play a role in this setting?
- Is there a way to recursively compute the order of Hecke-Kiselman monoids as in the Coxeter setting?
- Can one prov(id)e a characterization of digraphs inducing finite Hecke-Kiselman monoids?

Questions

- Can one find a canonical combinatorial action of Hecke-Kiselman monoids on something?
- Can one set up a universal update system also on oriented graphs with cycles?
- Does Coxeter combinatorics play a role in this setting?
- Is there a way to recursively compute the order of Hecke-Kiselman monoids as in the Coxeter setting?
- Can one prov(id)e a characterization of digraphs inducing finite Hecke-Kiselman monoids?

Questions

- Can one find a canonical combinatorial action of Hecke-Kiselman monoids on something?
- Can one set up a universal update system also on oriented graphs with cycles?
- Does Coxeter combinatorics play a role in this setting?
- Is there a way to recursively compute the order of Hecke-Kiselman monoids as in the Coxeter setting?
- Can one prov(id)e a characterization of digraphs inducing finite Hecke-Kiselman monoids?

Questions

- Can one find a canonical combinatorial action of Hecke-Kiselman monoids on something?
- Can one set up a universal update system also on oriented graphs with cycles?
- Does Coxeter combinatorics play a role in this setting?
- Is there a way to recursively compute the order of Hecke-Kiselman monoids as in the Coxeter setting?
- Can one prov(id)e a characterization of digraphs inducing finite Hecke-Kiselman monoids?

Bibliography

- M.H.A. Newman, "On theories with a combinatorial definition of equivalence", Ann. Math 1942
- P.N. Norton, "0-Hecke algebras", J. Austr. Math. Soc. 1979
- G.P. Huet, "Confluent reductions: abstract properties and applications to term rewriting systems", J. Assoc. Computer Machinery 1980
- R. Richardson, T. Springer, "The Bruhat order on symmetric varieties", Geom. dedicata 1990
- A. Solomon, "Catalan monoids, monoids of local endomorphisms, and their presentations", Semigroup Forum 1996
- H.S. Mortveit, C.M. Reidys, "Discrete, sequential dynamical systems", Discr. Math. 2001
- C. Kiselman, "A semigroup of operators in convexity theory", Trans. AMS 2002
- G. Kudryavtseva, V. Mazorchuk, "On Kiselman's semigroup", Yokohama Math. J. 2009
- T. Kenney, "Coxeter groups, Coxeter monoids and the Bruhat order", J. Alg. Comb. 2013
- R. Aragona, A. D'Andrea, "Hecke-Kiselman monoids of small cardinality", Semigroup Forum 2013
- E. Collina, A. D'Andrea, "A graph-dynamical interpretation of Kiselman's semigroups", J. Alg. Comb. 2015
- R. Aragona, A. D'Andrea, "Normal form in Hecke-Kiselman monoids associated with simple oriented graphs", Alg. Discr. Math. 2020
- A. D'Andrea, S. Stella, "The cardinality of Kiselman's semigroups grows double-exponentially", draft

Thanks
for your attention!!!