Multiline queues with spectral parameters

Darij Grinberg
joint work with Erik Aas and Travis Scrimshaw

28 June 2018
Leibniz Universität Hannover

slides: http://www.cip.ifi.lmu.de/~grinberg/algebra/hannover2018.pdf
paper:
http://www.cip.ifi.lmu.de/~grinberg/algebra/mlqs.pdf

Sites and words

- We study a combinatorial algorithm: the action of *queues* on *words*.
- Fix a positive integer n.

Sites and words

- We study a combinatorial algorithm: the action of *queues* on *words*.
- Fix a positive integer n.
- For a nonnegative integer k, let $[k]$ be the set $\{1, 2, \ldots, k\}$.

Sites and words

- We study a combinatorial algorithm: the action of *queues* on *words*.
- Fix a positive integer n.
- For a nonnegative integer k, let $[k]$ be the set $\{1, 2, \ldots, k\}$.
- We shall refer to the elements $1, 2, \ldots, n \in \mathbb{Z}/n\mathbb{Z}$ as *sites*. Regard them as points on a line that "wraps around" cyclically:

$$\cdots \quad n-1 \quad n \quad 1 \quad 2 \quad \cdots \quad n-1 \quad n \quad 1 \quad 2 \quad \cdots$$

Sites and words

- We study a combinatorial algorithm: the action of *queues* on *words*.
- Fix a positive integer n.
- For a nonnegative integer k, let $[k]$ be the set $\{1, 2, \ldots, k\}$.
- We shall refer to the elements $1, 2, \ldots, n \in \mathbb{Z}/n\mathbb{Z}$ as *sites*. Regard them as points on a line that "wraps around" cyclically:

$$\cdots \quad n-1 \quad n \quad 1 \quad 2 \quad \cdots \quad n-1 \quad n \quad 1 \quad 2 \quad \cdots$$

- A *word* means a map $\{\text{sites}\} \to \{\text{positive integers}\}$. If u is a word and i is a site, then $u_i := u(i)$.

Sites and words

- We study a combinatorial algorithm: the action of *queues* on *words*.
- Fix a positive integer n.
- For a nonnegative integer k, let $[k]$ be the set $\{1, 2, \ldots, k\}$.
- We shall refer to the elements $1, 2, \ldots, n \in \mathbb{Z}/n\mathbb{Z}$ as *sites*. Regard them as points on a line that "wraps around" cyclically:

$$\cdots \quad n-1 \quad n \quad 1 \quad 2 \quad \cdots \quad n-1 \quad n \quad 1 \quad 2 \quad \cdots$$

- A *word* means a map $\{\text{sites}\} \to \{\text{positive integers}\}$. If u is a word and i is a site, then $u_i := u(i)$.
- Write $u_1 u_2 \cdots u_n$ for a word u ("one-line notation").

Sites and words

- We study a combinatorial algorithm: the action of *queues* on *words*.
- Fix a positive integer n.
- For a nonnegative integer k, let $[k]$ be the set $\{1, 2, \ldots, k\}$.
- We shall refer to the elements $1, 2, \ldots, n \in \mathbb{Z}/n\mathbb{Z}$ as *sites*. Regard them as points on a line that "wraps around" cyclically:

$$\cdots \quad n-1 \quad n \quad 1 \quad 2 \quad \cdots \quad n-1 \quad n \quad 1 \quad 2 \quad \cdots$$

- A *word* means a map $\{\text{sites}\} \to \{\text{positive integers}\}$. If u is a word and i is a site, then $u_i := u(i)$.
- Write $u_1 u_2 \cdots u_n$ for a word u ("one-line notation"). Example: The word 33122 (for $n = 5$) is the map

$$\begin{array}{cccccccccccc} & i & = & \cdots & 4 & 5 & 1 & 2 & 3 & 4 & 5 & 1 & 2 & \cdots \\ \mapsto & u_i & = & \cdots & 2 & 2 & 3 & 3 & 1 & 2 & 2 & 3 & 3 & \cdots \end{array}$$

Queues

- A *queue* means a set of sites.

Queues

- A *queue* means a set of sites.
- Draw a queue q by putting circles on all the sites $i \in q$.

Queues

- A *queue* means a set of sites.
- Draw a queue q by putting circles on all the sites $i \in q$.
  Example: The queue $\{2, 5\}$ (for $n = 7$) is represented by

$$\cdots \quad 6 \quad 7 \quad 1 \quad \textcircled{2} \quad 3 \quad 4 \quad \textcircled{5} \quad 6 \quad 7 \quad 1 \quad \textcircled{2} \quad \cdots$$

Queues

- A *queue* means a set of sites.
- Draw a queue q by putting circles on all the sites $i \in q$.
  Example: The queue $\{2, 5\}$ (for $n = 7$) is represented by

$$\cdots \quad 6 \quad 7 \quad 1 \quad \textcircled{2} \quad 3 \quad 4 \quad \textcircled{5} \quad 6 \quad 7 \quad 1 \quad \textcircled{2} \quad \cdots$$

- We shall omit all the grey parts in the future (i.e., we will draw only one copy of each site).

Queues

- A *queue* means a set of sites.
- Draw a queue q by putting circles on all the sites $i \in q$.
  Example: The queue $\{2, 5\}$ (for $n = 7$) is represented by

$$1 \quad \textcircled{2} \quad 3 \quad 4 \quad \textcircled{5} \quad 6 \quad 7$$

- We shall omit all the grey parts in the future (i.e., we will draw only one copy of each site).

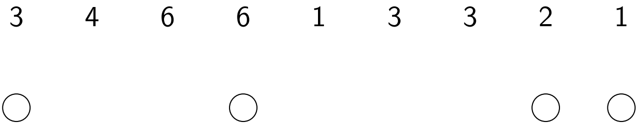
Action of queues on words, 1: example

- Let q be a queue, and u a word. We shall define a word $q(u)$.

Action of queues on words, 1: example

- Let q be a queue, and u a word. We shall define a word $q(u)$.
- **Idea:**
  - Draw u on top.
  - Draw q as circles in the middle.
  - Build $q(u)$ letter by letter, as follows...

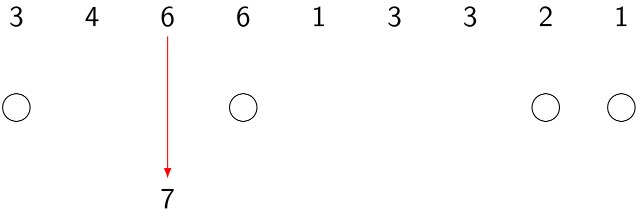
Example: $n = 9$ and $u = 346613321$ and $q = \{1, 4, 8, 9\}$:



Action of queues on words, 1: example

- Let q be a queue, and u a word. We shall define a word $q(u)$.
- **Idea:**
  - Draw u on top.
  - Draw q as circles in the middle.
  - Build $q(u)$ letter by letter, as follows...

Example: $n = 9$ and $u = 346613321$ and $q = \{1, 4, 8, 9\}$:



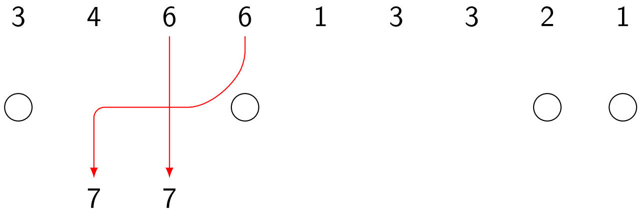
Phase I: For each of the largest $n - |q|$ letters of u (in decreasing order),

- **drop** this letter **down** and **add** 1 to it;
- **move** it left until hitting some unoccupied site $i \notin q$;
- **place** it there.

Action of queues on words, 1: example

- Let q be a queue, and u a word. We shall define a word $q(u)$.
- **Idea:**
  - Draw u on top.
  - Draw q as circles in the middle.
  - Build $q(u)$ letter by letter, as follows...

Example: $n = 9$ and $u = 346613321$ and $q = \{1, 4, 8, 9\}$:



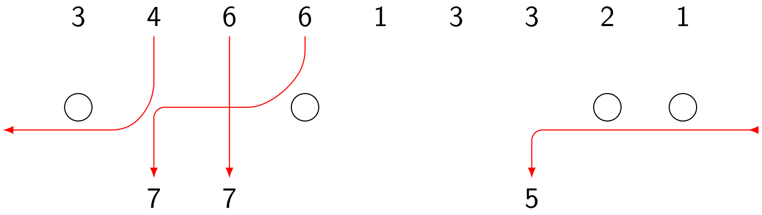
Phase I: For each of the largest $n - |q|$ letters of u (in decreasing order),

- **drop** this letter **down** and **add** 1 to it;
- **move** it left until hitting some unoccupied site $i \notin q$;
- **place** it there.

Action of queues on words, 1: example

- Let q be a queue, and u a word. We shall define a word $q(u)$.
- **Idea:**
  - Draw u on top.
  - Draw q as circles in the middle.
  - Build $q(u)$ letter by letter, as follows...

Example: $n = 9$ and $u = 346613321$ and $q = \{1, 4, 8, 9\}$:



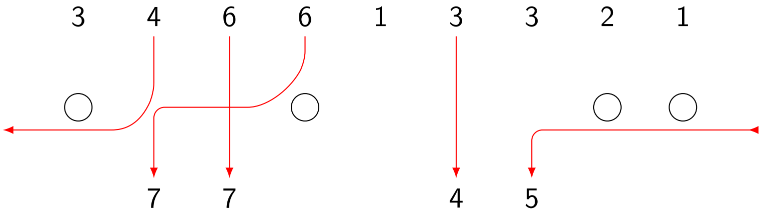
Phase I: For each of the largest $n - |q|$ letters of u (in decreasing order),

- **drop** this letter **down** and **add** 1 to it;
- **move** it left until hitting some unoccupied site $i \notin q$;
- **place** it there.

Action of queues on words, 1: example

- Let q be a queue, and u a word. We shall define a word $q(u)$.
- **Idea:**
  - Draw u on top.
  - Draw q as circles in the middle.
  - Build $q(u)$ letter by letter, as follows...

Example: $n = 9$ and $u = 346613321$ and $q = \{1, 4, 8, 9\}$:



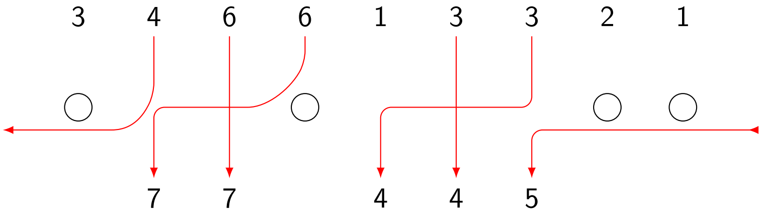
Phase I: For each of the largest $n - |q|$ letters of u (in decreasing order),

- **drop** this letter **down** and **add** 1 to it;
- **move** it left until hitting some unoccupied site $i \notin q$;
- **place** it there.

Action of queues on words, 1: example

- Let q be a queue, and u a word. We shall define a word $q(u)$.
- **Idea:**
  - Draw u on top.
  - Draw q as circles in the middle.
  - Build $q(u)$ letter by letter, as follows...

Example: $n = 9$ and $u = 346613321$ and $q = \{1, 4, 8, 9\}$:



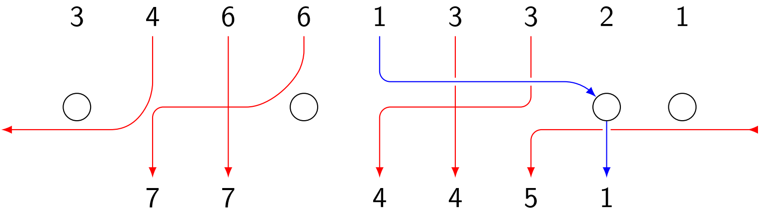
Phase I: For each of the largest $n - |q|$ letters of u (in decreasing order),

- **drop** this letter **down** and **add** 1 to it;
- **move** it left until hitting some unoccupied site $i \notin q$;
- **place** it there.

Action of queues on words, 1: example

- Let q be a queue, and u a word. We shall define a word $q(u)$.
- **Idea:**
  - Draw u on top.
  - Draw q as circles in the middle.
  - Build $q(u)$ letter by letter, as follows...

Example: $n = 9$ and $u = 346613321$ and $q = \{1, 4, 8, 9\}$:

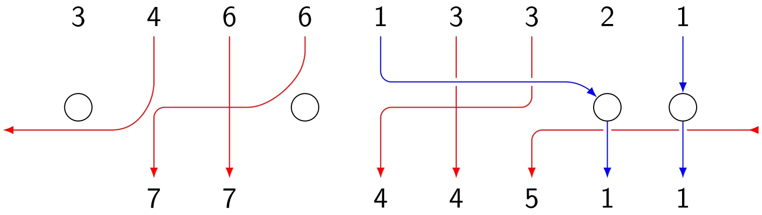


Phase II: For each of the smallest $|q|$ letters of u (in increasing order),

- **drop** this letter down;
- **move** it **right** until hitting some unoccupied site $i \in q$;
- **place** it there.

- Let q be a queue, and u a word. We shall define a word $q(u)$.
- **Idea:**
  - Draw u on top.
  - Draw q as circles in the middle.
  - Build $q(u)$ letter by letter, as follows...

Example: $n = 9$ and $u = 346613321$ and $q = \{1, 4, 8, 9\}$:



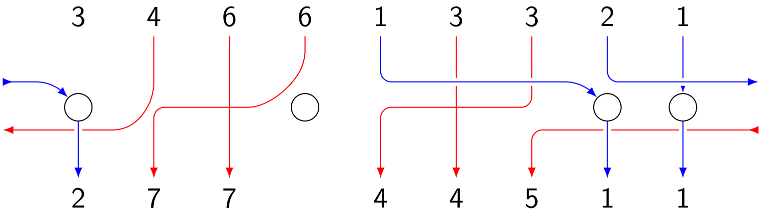
Phase II: For each of the smallest $|q|$ letters of u (in increasing order),

- **drop** this letter down;
- **move** it **right** until hitting some unoccupied site $i \in q$;
- **place** it there.

Action of queues on words, 1: example

- Let q be a queue, and u a word. We shall define a word $q(u)$.
- **Idea:**
  - Draw u on top.
  - Draw q as circles in the middle.
  - Build $q(u)$ letter by letter, as follows...

Example: $n = 9$ and $u = 346613321$ and $q = \{1, 4, 8, 9\}$:



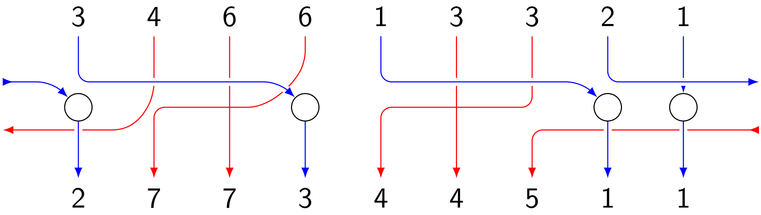
Phase II: For each of the smallest $|q|$ letters of u (in increasing order),

- **drop** this letter down;
- **move** it **right** until hitting some unoccupied site $i \in q$;
- **place** it there.

Action of queues on words, 1: example

- Let q be a queue, and u a word. We shall define a word $q(u)$.
- **Idea:**
  - Draw u on top.
  - Draw q as circles in the middle.
  - Build $q(u)$ letter by letter, as follows...

Example: $n = 9$ and $u = 346613321$ and $q = \{1, 4, 8, 9\}$:



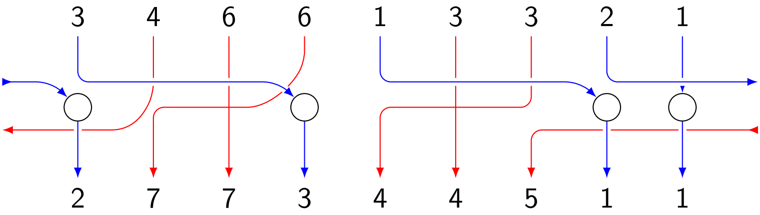
Phase II: For each of the smallest $|q|$ letters of u (in increasing order),

- **drop** this letter down;
- **move** it **right** until hitting some unoccupied site $i \in q$;
- **place** it there.

Action of queues on words, 1: example

- Let q be a queue, and u a word. We shall define a word $q(u)$.
- **Idea:**
  - Draw u on top.
  - Draw q as circles in the middle.
  - Build $q(u)$ letter by letter, as follows...

Example: $n = 9$ and $u = 346613321$ and $q = \{1, 4, 8, 9\}$:

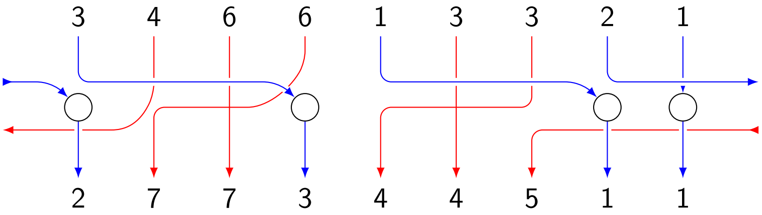


The letters on the bottom now form $q(u)$.

Action of queues on words, 1: example

- Let q be a queue, and u a word. We shall define a word $q(u)$.
- **Idea:**
  - Draw u on top.
  - Draw q as circles in the middle.
  - Build $q(u)$ letter by letter, as follows...

Example: $n = 9$ and $u = 346613321$ and $q = \{1, 4, 8, 9\}$:



The letters on the bottom now form $q(u)$.

Proposition. Equal letters can be processed in any order.

Action of queues on words, 2: formal definition

- Let q be a queue, and u a word. Define a word $q(u)$ as follows:
  In the beginning, $v = q(u)$ is a word whose letters are unset.
  Choose a permutation $(i_1, i_2, \ldots, i_n)$ of $(1, 2, \ldots, n)$ such that $u_{i_1} \leq u_{i_2} \leq \cdots \leq u_{i_n}$.

  Phase I. For $i = i_n, i_{n-1}, \ldots, i_{|q|+1}$, do the following: Find the first site j weakly to the left (cyclically) of i such that $j \notin q$ and v_j is not set. Then set $v_j = u_i + 1$.

  Phase II. For $i = i_1, i_2, \ldots, i_{|q|}$, do the following: Find the first site j weakly to the right (cyclically) of i such that $j \in q$ and v_j is not set. Then set $v_j = u_i$.

Action of queues on words, 2: formal definition

- Let q be a queue, and u a word. Define a word $q(u)$ as follows:
  In the beginning, $v = q(u)$ is a word whose letters are unset.
  Choose a permutation $(i_1, i_2, \ldots, i_n)$ of $(1, 2, \ldots, n)$ such that $u_{i_1} \leq u_{i_2} \leq \cdots \leq u_{i_n}$.

  Phase I. For $i = i_n, i_{n-1}, \ldots, i_{|q|+1}$, do the following: Find the first site j weakly to the left (cyclically) of i such that $j \notin q$ and v_j is not set. Then set $v_j = u_i + 1$.

  Phase II. For $i = i_1, i_2, \ldots, i_{|q|}$, do the following: Find the first site j weakly to the right (cyclically) of i such that $j \in q$ and v_j is not set. Then set $v_j = u_i$.

- **Proposition.**
  - The resulting word $v = q(u)$ does not depend on the choice of permutation $(i_1, i_2, \ldots, i_n)$.
  - Phase I and Phase II can be done in parallel.

Remark on TASEP connection

- This action of queues on words is inspired by the "discrete MLQs" of Aas and Linusson (arXiv:1501.04417).
  Main difference: They have no Phase I, but their words have empty positions.

Remark on TASEP connection

- This action of queues on words is inspired by the "discrete MLQs" of Aas and Linusson (arXiv:1501.04417).
  Main difference: They have no Phase I, but their words have empty positions.
  (Our picture subsumes theirs – fill the empty positions with high letters.)

Remark on TASEP connection

- This action of queues on words is inspired by the "discrete MLQs" of Aas and Linusson (arXiv:1501.04417).
  Main difference: They have no Phase I, but their words have empty positions.
  (Our picture subsumes theirs – fill the empty positions with high letters.)
- Their motivation: compute stationary distribution of TASEP (totally asymmetric exclusion process) on a circle.
  Our work proves two of their conjectures.

Types of words

- The *type* of a word u is the sequence $\mathbf{m} = (m_1, m_2, \ldots)$, where $m_k = (\#$ of all sites i such that $u_i = k)$.

Types of words

- The *type* of a word u is the sequence $\mathbf{m} = (m_1, m_2, \ldots)$, where $m_k = (\#$ of all sites i such that $u_i = k)$.
  Example: The word 1255135 has type $(2, 1, 1, 0, 3, 0, 0, 0, \ldots)$.

Types of words

- The *type* of a word u is the sequence $\mathbf{m} = (m_1, m_2, \ldots)$, where $m_k = (\#$ of all sites i such that $u_i = k)$.
  Example: The word 1255135 has type $(2, 1, 1, 0, 3, 0, 0, 0, \ldots)$.
- We omit trailing zeroes from infinite sequences.
  That is, we abbreviate $(m_1, m_2, \ldots, m_k, 0, 0, 0, \ldots)$ as $(m_1, m_2, \ldots, m_k)$.

Types of words

- The *type* of a word u is the sequence $\mathbf{m} = (m_1, m_2, \ldots)$, where $m_k = (\#$ of all sites i such that $u_i = k)$.
  Example: The word 1255135 has type $(2, 1, 1, 0, 3)$.
- We omit trailing zeroes from infinite sequences.
  That is, we abbreviate $(m_1, m_2, \ldots, m_k, 0, 0, 0, \ldots)$ as $(m_1, m_2, \ldots, m_k)$.

Types of words

- The *type* of a word u is the sequence $\mathbf{m} = (m_1, m_2, \ldots)$, where $m_k = (\#$ of all sites i such that $u_i = k)$.
  Example: The word 1255135 has type $(2, 1, 1, 0, 3)$.
- We omit trailing zeroes from infinite sequences.
  That is, we abbreviate $(m_1, m_2, \ldots, m_k, 0, 0, 0, \ldots)$ as $(m_1, m_2, \ldots, m_k)$.
- A word u is *packed with ℓ classes* if its type $\mathbf{m}$ has $m_1, m_2, \ldots, m_\ell > 0$ and $m_{\ell+1} = m_{\ell+2} = \cdots = 0$.

Types of words

- The *type* of a word u is the sequence $\mathbf{m} = (m_1, m_2, \ldots)$, where $m_k = (\#\text{ of all sites } i \text{ such that } u_i = k)$.
  Example: The word 1255135 has type $(2, 1, 1, 0, 3)$.
- We omit trailing zeroes from infinite sequences.
  That is, we abbreviate $(m_1, m_2, \ldots, m_k, 0, 0, 0, \ldots)$ as $(m_1, m_2, \ldots, m_k)$.
- A word u is *packed with ℓ classes* if its type $\mathbf{m}$ has $m_1, m_2, \ldots, m_\ell > 0$ and $m_{\ell+1} = m_{\ell+2} = \cdots = 0$.
  Example: The word 1255135 is not packed.
  The word 1244134 is packed with 4 classes.

MLQs

- A *MLQ* (short for "multiline queue") is a tuple of queues.

MLQs

- A *MLQ* (short for "multiline queue") is a tuple of queues.
- If $\mathbf{q} = (q_1, q_2, \ldots, q_k)$ is an MLQ, and u is a word, then

$$\mathbf{q}(u) := q_k\left(q_{k-1}\left(\cdots\left(q_1(u)\right)\right)\right).$$

MLQs

- A *MLQ* (short for "multiline queue") is a tuple of queues.
- If $\mathbf{q} = (q_1, q_2, \ldots, q_k)$ is an MLQ, and u is a word, then

$$\mathbf{q}(u) := q_k\left(q_{k-1}\left(\cdots\left(q_1(u)\right)\right)\right).$$

- Let $\ell > 0$, and let σ be a permutation of $[\ell - 1]$.
  Let $\mathbf{m} = (m_1, m_2, \ldots, m_\ell)$ be a sequence of positive integers.
  A *σ-twisted MLQ of type $\mathbf{m}$* means an MLQ
  $\mathbf{q} = (q_1, q_2, \ldots, q_{\ell-1})$ such that

$$|q_i| = m_1 + m_2 + \cdots + m_{\sigma(i)} \text{ for all } i, \qquad \text{and}$$

$$n = m_1 + m_2 + \cdots .$$

MLQs

- A *MLQ* (short for "multiline queue") is a tuple of queues.
- If $\mathbf{q} = (q_1, q_2, \ldots, q_k)$ is an MLQ, and u is a word, then
$$\mathbf{q}(u) := q_k\left(q_{k-1}\left(\cdots\left(q_1(u)\right)\right)\right).$$
- Let $\ell > 0$, and let σ be a permutation of $[\ell - 1]$.
Let $\mathbf{m} = (m_1, m_2, \ldots, m_\ell)$ be a sequence of positive integers.
A *σ-twisted MLQ of type* $\mathbf{m}$ means an MLQ
$\mathbf{q} = (q_1, q_2, \ldots, q_{\ell-1})$ such that
$$|q_i| = m_1 + m_2 + \cdots + m_{\sigma(i)} \text{ for all } i, \qquad \text{and}$$
$$n = m_1 + m_2 + \cdots.$$
Example: $n = 6$ and $\mathbf{m} = (2, 3, 1)$ and $\ell = 3$ and $\sigma = (2, 1)$ (one-line notation). Then, a σ-twisted MLQ of type $\mathbf{m}$ is an MLQ $\mathbf{q} = (q_1, q_2)$ with $|q_1| = m_1 + m_2 = 2 + 3 = 5$ and $|q_2| = m_1 = 2$.

MLQs

- A *MLQ* (short for "multiline queue") is a tuple of queues.
- If $\mathbf{q} = (q_1, q_2, \ldots, q_k)$ is an MLQ, and u is a word, then

$$\mathbf{q}(u) := q_k\left(q_{k-1}\left(\cdots\left(q_1(u)\right)\right)\right).$$

- Let $\ell > 0$, and let σ be a permutation of $[\ell - 1]$.
  Let $\mathbf{m} = (m_1, m_2, \ldots, m_\ell)$ be a sequence of positive integers.
  A *σ-twisted MLQ of type $\mathbf{m}$* means an MLQ
  $\mathbf{q} = (q_1, q_2, \ldots, q_{\ell-1})$ such that

$$|q_i| = m_1 + m_2 + \cdots + m_{\sigma(i)} \text{ for all } i, \qquad \text{and}$$

$$n = m_1 + m_2 + \cdots.$$

  Example: $n = 6$ and $\mathbf{m} = (2, 3, 1)$ and $\ell = 3$ and $\sigma = (2, 1)$ (one-line notation). Then, a σ-twisted MLQ of type $\mathbf{m}$ is an MLQ $\mathbf{q} = (q_1, q_2)$ with $|q_1| = m_1 + m_2 = 2 + 3 = 5$ and $|q_2| = m_1 = 2$. For example, $\mathbf{q} = (\{1, 3, 4, 5, 6\}, \{2, 3\})$

MLQs

- A *MLQ* (short for "multiline queue") is a tuple of queues.
- If $\mathbf{q} = (q_1, q_2, \ldots, q_k)$ is an MLQ, and u is a word, then

$$\mathbf{q}(u) := q_k\left(q_{k-1}\left(\cdots\left(q_1(u)\right)\right)\right).$$

- Let $\ell > 0$, and let σ be a permutation of $[\ell - 1]$.
  Let $\mathbf{m} = (m_1, m_2, \ldots, m_\ell)$ be a sequence of positive integers.
  A *σ-twisted MLQ of type $\mathbf{m}$* means an MLQ
  $\mathbf{q} = (q_1, q_2, \ldots, q_{\ell-1})$ such that

$$|q_i| = m_1 + m_2 + \cdots + m_{\sigma(i)} \text{ for all } i, \qquad \text{and}$$

$$n = m_1 + m_2 + \cdots.$$

  Example: $n = 6$ and $\mathbf{m} = (2, 3, 1)$ and $\ell = 3$ and $\sigma = (2, 1)$ (one-line notation). Then, a σ-twisted MLQ of type $\mathbf{m}$ is an MLQ $\mathbf{q} = (q_1, q_2)$ with $|q_1| = m_1 + m_2 = 2 + 3 = 5$ and $|q_2| = m_1 = 2$. For example, $\mathbf{q} = (\{1, 3, 4, 5, 6\}, \{2, 3\})$ and $\mathbf{q}(111111) = 311222$.

MLQs

- A *MLQ* (short for "multiline queue") is a tuple of queues.
- If $\mathbf{q} = (q_1, q_2, \ldots, q_k)$ is an MLQ, and u is a word, then

$$\mathbf{q}(u) := q_k\left(q_{k-1}\left(\cdots\left(q_1(u)\right)\right)\right).$$

- Let $\ell > 0$, and let σ be a permutation of $[\ell - 1]$.
  Let $\mathbf{m} = (m_1, m_2, \ldots, m_\ell)$ be a sequence of positive integers.
  A *σ-twisted MLQ of type* $\mathbf{m}$ means an MLQ
  $\mathbf{q} = (q_1, q_2, \ldots, q_{\ell-1})$ such that

$$|q_i| = m_1 + m_2 + \cdots + m_{\sigma(i)} \text{ for all } i, \qquad \text{and}$$

$$n = m_1 + m_2 + \cdots.$$

  Example: $n = 6$ and $\mathbf{m} = (2, 3, 1)$ and $\ell = 3$ and $\sigma = (2, 1)$ (one-line notation). Then, a σ-twisted MLQ of type $\mathbf{m}$ is an MLQ $\mathbf{q} = (q_1, q_2)$ with $|q_1| = m_1 + m_2 = 2 + 3 = 5$ and $|q_2| = m_1 = 2$. For example, $\mathbf{q} = (\{1, 3, 4, 5, 6\}, \{4, 5\})$

MLQs

- A *MLQ* (short for "multiline queue") is a tuple of queues.
- If $\mathbf{q} = (q_1, q_2, \ldots, q_k)$ is an MLQ, and u is a word, then

$$\mathbf{q}(u) := q_k\left(q_{k-1}\left(\cdots\left(q_1(u)\right)\right)\right).$$

- Let $\ell > 0$, and let σ be a permutation of $[\ell - 1]$.
  Let $\mathbf{m} = (m_1, m_2, \ldots, m_\ell)$ be a sequence of positive integers.
  A *σ-twisted MLQ of type $\mathbf{m}$* means an MLQ
  $\mathbf{q} = (q_1, q_2, \ldots, q_{\ell-1})$ such that

$$|q_i| = m_1 + m_2 + \cdots + m_{\sigma(i)} \text{ for all } i, \qquad \text{and}$$

$$n = m_1 + m_2 + \cdots.$$

  Example: $n = 6$ and $\mathbf{m} = (2, 3, 1)$ and $\ell = 3$ and $\sigma = (2, 1)$ (one-line notation). Then, a σ-twisted MLQ of type $\mathbf{m}$ is an MLQ $\mathbf{q} = (q_1, q_2)$ with $|q_1| = m_1 + m_2 = 2 + 3 = 5$ and $|q_2| = m_1 = 2$. For example, $\mathbf{q} = (\{1, 3, 4, 5, 6\}, \{4, 5\})$ and $\mathbf{q}(111111) = 232112$.

MLQs

- A *MLQ* (short for "multiline queue") is a tuple of queues.
- If $\mathbf{q} = (q_1, q_2, \ldots, q_k)$ is an MLQ, and u is a word, then

$$\mathbf{q}(u) := q_k\left(q_{k-1}\left(\cdots\left(q_1(u)\right)\right)\right).$$

- Let $\ell > 0$, and let σ be a permutation of $[\ell - 1]$.
  Let $\mathbf{m} = (m_1, m_2, \ldots, m_\ell)$ be a sequence of positive integers.
  A *σ-twisted MLQ of type* $\mathbf{m}$ means an MLQ
  $\mathbf{q} = (q_1, q_2, \ldots, q_{\ell-1})$ such that

$$|q_i| = m_1 + m_2 + \cdots + m_{\sigma(i)} \text{ for all } i, \qquad \text{and}$$

$$n = m_1 + m_2 + \cdots.$$

- Equivalently: A *σ-twisted MLQ of type* $\mathbf{m}$ can be defined as an MLQ $\mathbf{q} = (q_1, q_2, \ldots, q_{\ell-1})$ such that
  - the word $\mathbf{q}(1 \cdots 1)$ has type $\mathbf{m}$ (where $1 \cdots 1$ is the word whose values all equal 1);
  - we have $0 < \left|q_{\sigma^{-1}(1)}\right| < \left|q_{\sigma^{-1}(2)}\right| < \cdots < \left|q_{\sigma^{-1}(\ell-1)}\right|$.

Generating functions, 1: definition

- Now, let $x_1, x_2, \ldots, x_n$ be commuting variables.

Generating functions, 1: definition

- Now, let $x_1, x_2, \ldots, x_n$ be commuting variables.
- For any $\ell \geq 1$, any permutation σ of $[\ell - 1]$, and any packed word u of type $\mathbf{m}$ with ℓ classes, we define the *σ-spectral weight $\langle u \rangle_\sigma$* by

$$\langle u \rangle_\sigma := \sum_{\substack{\mathbf{q} \text{ is a } \sigma\text{-twisted} \\ \text{MLQ of type } \mathbf{m} \\ \text{satisfying } u = \mathbf{q}(1 \cdots 1)}} \operatorname{wt} \mathbf{q}.$$

Here:

- $1 \cdots 1$ denotes the word whose all values are 1.
- $\operatorname{wt} \mathbf{q} := \prod_{p=1}^{k} \prod_{i \in q_p} x_i$ for any MLQ $\mathbf{q} = (q_1, q_2, \ldots, q_k)$.

- Now, let $x_1, x_2, \ldots, x_n$ be commuting variables.
- For any $\ell \geq 1$, any permutation σ of $[\ell - 1]$, and any packed word u of type $\mathbf{m}$ with ℓ classes, we define the *σ-spectral weight $\langle u \rangle_\sigma$* by

$$\langle u \rangle_\sigma := \sum_{\substack{\mathbf{q} \text{ is a } \sigma\text{-twisted} \\ \text{MLQ of type } \mathbf{m} \\ \text{satisfying } u = \mathbf{q}(1 \cdots 1)}} \operatorname{wt} \mathbf{q}.$$

Here:

- $1 \cdots 1$ denotes the word whose all values are 1.
- $\operatorname{wt} \mathbf{q} := \prod_{p=1}^{k} \prod_{i \in q_p} x_i$ for any MLQ $\mathbf{q} = (q_1, q_2, \ldots, q_k)$.

Example: Recall that $(\{1,3,4,5,6\}, \{4,5\})$ is a σ-twisted MLQ of type $\mathbf{m}$ for $n = 6$ and $\mathbf{m} = (2,3,1)$ and $\ell = 3$ and $\sigma = (2,1)$ (one-line notation) satisfying $\mathbf{q}(111111) = 232112$. It contributes a monomial

$$(x_1 x_3 x_4 x_5 x_6)(x_4 x_5) = x_1 x_3 x_4^2 x_5^2 x_6 \quad \text{to } \langle 232112 \rangle_\sigma.$$

Generating functions, 1: definition

- Now, let $x_1, x_2, \ldots, x_n$ be commuting variables.
- For any $\ell \geq 1$, any permutation σ of $[\ell - 1]$, and any packed word u of type $\mathbf{m}$ with ℓ classes, we define the *σ-spectral weight $\langle u \rangle_\sigma$* by

$$\langle u \rangle_\sigma := \sum_{\substack{\mathbf{q} \text{ is a } \sigma\text{-twisted} \\ \text{MLQ of type } \mathbf{m} \\ \text{satisfying } u = \mathbf{q}(1 \cdots 1)}} \operatorname{wt} \mathbf{q}.$$

  Here:
  - $1 \cdots 1$ denotes the word whose all values are 1.
  - $\operatorname{wt} \mathbf{q} := \prod_{p=1}^{k} \prod_{i \in q_p} x_i$ for any MLQ $\mathbf{q} = (q_1, q_2, \ldots, q_k)$.
- Set $\langle u \rangle := \langle u \rangle_{\mathrm{id}}$ for the permutation id of $[\ell - 1]$.

Generating functions, 2: more examples

- Example: For $n = 5$, $\ell = 5$ and $\mathbf{m} = (1, 1, 2, 1)$, we have

$$\langle 13234 \rangle = x_1 x_2 x_3^2 x_4 (x_1^2 + x_1 x_4 + x_1 x_5 + x_4 x_5 + x_5^2).$$

Generating functions, 2: more examples

- Example: For $n = 5$, $\ell = 5$ and $\mathbf{m} = (1,1,2,1)$, we have

$$\langle 13234\rangle = x_1x_2x_3^2x_4(x_1^2 + x_1x_4 + x_1x_5 + x_4x_5 + x_5^2).$$

- Examples: For $n = 5$, $\ell = 5$ and $\mathbf{m} = (1,1,1,1,1)$, we have

$$\begin{aligned}\langle 13245\rangle &= x_1x_2x_3^2x_4(x_1^2 + x_1x_4 + x_1x_5 + x_4^2 + x_4x_5 + x_5^2)\\ &\quad\cdot(x_1x_2x_3 + x_1x_2x_5 + x_1x_3x_5 + x_2x_3x_5),\\ \langle 14235\rangle &= x_1x_2x_3^2x_4^2(x_1^3x_2 + x_1^3x_3 + x_1^3x_5 + x_1^2x_2x_3 + x_1^2x_2x_4\\ &\quad + 2x_1^2x_2x_5 + x_1^2x_3x_4 + 2x_1^2x_3x_5 + x_1^2x_4x_5\\ &\quad + x_1^2x_5^2 + x_1x_2x_3x_5 + x_1x_2x_4x_5 + 2x_1x_2x_5^2\\ &\quad + x_1x_3x_4x_5 + 2x_1x_3x_5^2 + x_1x_4x_5^2 + x_1x_5^3\\ &\quad + x_2x_3x_5^2 + x_2x_4x_5^2 + x_2x_5^3 + x_3x_4x_5^2 + x_3x_5^3).\end{aligned}$$

The symmetry theorem, 1: statement

- **Theorem.** For any $\ell \geq 1$, any permutation σ of $[\ell - 1]$, and any packed word u of type $\mathbf{m}$ with ℓ classes, we have

$$\langle u \rangle_\sigma = \langle u \rangle .$$

The symmetry theorem, 1: statement

- **Theorem.** For any $\ell \geq 1$, any permutation σ of $[\ell - 1]$, and any packed word u of type $\mathbf{m}$ with ℓ classes, we have
$$\langle u \rangle_\sigma = \langle u \rangle \, .$$
- This yields a recent conjecture by Arita, Ayyer, Mallick and Prolhac on the TASEP.

The symmetry theorem, 1: statement

- **Theorem.** For any $\ell \geq 1$, any permutation σ of $[\ell - 1]$, and any packed word u of type $\mathbf{m}$ with ℓ classes, we have
$$\langle u \rangle_\sigma = \langle u \rangle .$$
- This yields a recent conjecture by Arita, Ayyer, Mallick and Prolhac on the TASEP.
- This is proven bijectively, using a "duality transformation" on MLQs that leaves their action on words unchanged.
- **Main lemma.** If q_1 and q_2 are two queues, then there are two queues q_1' and q_2' satisfying
$$\left|q_1'\right| = \left|q_2\right| \qquad \text{and} \qquad \left|q_2'\right| = \left|q_1\right| \qquad \text{and}$$
$$\left(\prod_{i \in q_1'} x_i\right)\left(\prod_{i \in q_2'} x_i\right) = \left(\prod_{i \in q_1} x_i\right)\left(\prod_{i \in q_2} x_i\right)$$
such that every word u satisfies
$$q_1'\left(q_2'\left(u\right)\right) = q_1\left(q_2\left(u\right)\right).$$

The symmetry theorem, 2: idea of proof

- The construction of q'_1 and q'_2 is combinatorial:
  - Encode the pair (q_1, q_2) as a $2n$-letter word $b = (b_1, b_2, \ldots, b_{2n})$ over the 3-letter alphabet $\{), (, \circ\}$. Namely, for each i,
    - let b_{2i-1} be an opening parenthesis "$($" if $i \in q_1$, otherwise a neutral symbol "$\circ$";
    - let b_{2i} be a closing parenthesis "$)$" if $i \in q_2$, otherwise a neutral symbol "$\circ$".

The symmetry theorem, 2: idea of proof

- The construction of q_1' and q_2' is combinatorial:
  - Encode the pair (q_1, q_2) as a $2n$-letter word $b = (b_1, b_2, \ldots, b_{2n})$ over the 3-letter alphabet $\{), (, \circ\}$. Namely, for each i,
    - let b_{2i-1} be an opening parenthesis "(" if $i \in q_1$, otherwise a neutral symbol "$\circ$";
    - let b_{2i} be a closing parenthesis ")" if $i \in q_2$, otherwise a neutral symbol "$\circ$".

  Convenient example:

$$n = 10;$$
$$q_1 = \{2, 6, 7, 9\};$$
$$q_2 = \{1, 3, 5, 7, 8\}.$$

The symmetry theorem, 2: idea of proof

- The construction of q_1' and q_2' is combinatorial:
  - Encode the pair (q_1, q_2) as a $2n$-letter word $b = (b_1, b_2, \ldots, b_{2n})$ over the 3-letter alphabet $\{), (, \circ\}$. Namely, for each i,
    - let b_{2i-1} be an opening parenthesis "(" if $i \in q_1$, otherwise a neutral symbol "$\circ$";
    - let b_{2i} be a closing parenthesis ")" if $i \in q_2$, otherwise a neutral symbol "$\circ$".

  Convenient example:

$$n = 10;$$
$$q_1 = \{2, 6, 7, 9\};$$
$$q_2 = \{1, 3, 5, 7, 8\}.$$

  Then,

b	=	$\circ)$	$(\circ$	$\circ)$	$\circ\circ$	$\circ)$	$(\circ$	$()$	$\circ)$	$(\circ$	$\circ\circ$
i	=	1	2	3	4	5	6	7	8	9	10

The symmetry theorem, 2: idea of proof

- The construction of q_1' and q_2' is combinatorial:
  - Encode the pair (q_1, q_2) as a $2n$-letter word $b = (b_1, b_2, \ldots, b_{2n})$ over the 3-letter alphabet $\{), (, \circ\}$. Namely, for each i,
    - let b_{2i-1} be an opening parenthesis "(" if $i \in q_1$, otherwise a neutral symbol "$\circ$";
    - let b_{2i} be a closing parenthesis ")" if $i \in q_2$, otherwise a neutral symbol "$\circ$".

    Convenient example:

$$n = 10;$$
$$q_1 = \{2, 6, 7, 9\};$$
$$q_2 = \{1, 3, 5, 7, 8\}.$$

Then,

$$b = \circ)\quad (\circ\quad \circ)\quad \circ\circ\quad \circ)\quad (\circ\quad ()\quad \circ)\quad (\circ\quad \circ\circ$$

The symmetry theorem, 2: idea of proof

- The construction of q'_1 and q'_2 is combinatorial:
  - Encode the pair (q_1, q_2) as a $2n$-letter word $b = (b_1, b_2, \ldots, b_{2n})$ over the 3-letter alphabet $\{), (, \circ\}$. Namely, for each i,
    - let b_{2i-1} be an opening parenthesis "$($" if $i \in q_1$, otherwise a neutral symbol "$\circ$";
    - let b_{2i} be a closing parenthesis "$)$" if $i \in q_2$, otherwise a neutral symbol "$\circ$".

    Convenient example:

$$n = 10;$$
$$q_1 = \{2, 6, 7, 9\};$$
$$q_2 = \{1, 3, 5, 7, 8\}.$$

    Then,

$$b = \circ)(\circ\circ)\circ\circ\circ)(\circ()\circ)(\circ\circ\circ$$

The symmetry theorem, 2: idea of proof

- The construction of q'_1 and q'_2 is combinatorial:
  - Encode the pair (q_1, q_2) as a $2n$-letter word $b = (b_1, b_2, \ldots, b_{2n})$ over the 3-letter alphabet $\{), (, \circ\}$. Namely, for each i,
    - let b_{2i-1} be an opening parenthesis "(" if $i \in q_1$, otherwise a neutral symbol "$\circ$";
    - let b_{2i} be a closing parenthesis ")" if $i \in q_2$, otherwise a neutral symbol "$\circ$".
  - Match parentheses in b "the usual way" but keeping in mind that the word wraps around cyclically. In our above example:

$$b = \circ)(\circ\circ)\circ\circ\circ)(\circ()\circ)(\circ\circ\circ$$

The symmetry theorem, 2: idea of proof

- The construction of q'_1 and q'_2 is combinatorial:
  - Encode the pair (q_1, q_2) as a $2n$-letter word $b = (b_1, b_2, \ldots, b_{2n})$ over the 3-letter alphabet $\{), (, \circ\}$. Namely, for each i,
    - let b_{2i-1} be an opening parenthesis "(" if $i \in q_1$, otherwise a neutral symbol "$\circ$";
    - let b_{2i} be a closing parenthesis ")" if $i \in q_2$, otherwise a neutral symbol "$\circ$".
  - Match parentheses in b "the usual way" but keeping in mind that the word wraps around cyclically. In our above example:

$$b = \circ)(_1\circ\circ)_1\circ\circ\circ)(\circ()\circ)(\circ\circ\circ$$

The symmetry theorem, 2: idea of proof

- The construction of q_1' and q_2' is combinatorial:
  - Encode the pair (q_1, q_2) as a $2n$-letter word $b = (b_1, b_2, \ldots, b_{2n})$ over the 3-letter alphabet $\{), (, \circ\}$. Namely, for each i,
    - let b_{2i-1} be an opening parenthesis "(" if $i \in q_1$, otherwise a neutral symbol "$\circ$";
    - let b_{2i} be a closing parenthesis ")" if $i \in q_2$, otherwise a neutral symbol "$\circ$".
  - Match parentheses in b "the usual way" but keeping in mind that the word wraps around cyclically. In our above example:

$$b = \circ)(_1\circ\circ)_1\circ\circ\circ)(\circ(_2)_2\circ)(\circ\circ\circ$$

The symmetry theorem, 2: idea of proof

- The construction of q_1' and q_2' is combinatorial:
  - Encode the pair (q_1, q_2) as a $2n$-letter word $b = (b_1, b_2, \ldots, b_{2n})$ over the 3-letter alphabet $\{), (, \circ\}$. Namely, for each i,
    - let b_{2i-1} be an opening parenthesis "(" if $i \in q_1$, otherwise a neutral symbol "$\circ$";
    - let b_{2i} be a closing parenthesis ")" if $i \in q_2$, otherwise a neutral symbol "$\circ$".
  - Match parentheses in b "the usual way" but keeping in mind that the word wraps around cyclically. In our above example:

$$b = \circ)(_1\circ\circ)_1\circ\circ\circ)(_3\circ(_2)_2\circ)_3(\circ\circ\circ$$

The symmetry theorem, 2: idea of proof

- The construction of q_1' and q_2' is combinatorial:
  - Encode the pair (q_1, q_2) as a $2n$-letter word $b = (b_1, b_2, \ldots, b_{2n})$ over the 3-letter alphabet $\{), (, \circ\}$. Namely, for each i,
    - let b_{2i-1} be an opening parenthesis "(" if $i \in q_1$, otherwise a neutral symbol "$\circ$";
    - let b_{2i} be a closing parenthesis ")" if $i \in q_2$, otherwise a neutral symbol "$\circ$".
  - Match parentheses in b "the usual way" but keeping in mind that the word wraps around cyclically. In our above example:

$$b = \circ)_4(_1\circ\circ)_1\circ\circ\circ)(_3\circ(_2)_2\circ)_3(_4\circ\circ\circ$$

The symmetry theorem, 2: idea of proof

- The construction of q_1' and q_2' is combinatorial:
  - Encode the pair (q_1, q_2) as a $2n$-letter word $b = (b_1, b_2, \ldots, b_{2n})$ over the 3-letter alphabet $\{), (, \circ\}$. Namely, for each i,
    - let b_{2i-1} be an opening parenthesis "$($" if $i \in q_1$, otherwise a neutral symbol "$\circ$";
    - let b_{2i} be a closing parenthesis "$)$" if $i \in q_2$, otherwise a neutral symbol "$\circ$".
  - Match parentheses in b "the usual way" but keeping in mind that the word wraps around cyclically.
  - Replace the unmatched parentheses by their duals – e.g., if they were $)$'s, make them $($'s.

The symmetry theorem, 2: idea of proof

- The construction of q'_1 and q'_2 is combinatorial:
  - Encode the pair (q_1, q_2) as a $2n$-letter word $b = (b_1, b_2, \ldots, b_{2n})$ over the 3-letter alphabet $\{), (, \circ\}$. Namely, for each i,
    - let b_{2i-1} be an opening parenthesis "(" if $i \in q_1$, otherwise a neutral symbol "$\circ$";
    - let b_{2i} be a closing parenthesis ")" if $i \in q_2$, otherwise a neutral symbol "$\circ$".
  - Match parentheses in b "the usual way" but keeping in mind that the word wraps around cyclically.
  - Replace the unmatched parentheses by their duals – e.g., if they were)'s, make them ('s.

    In our above example:

$$b = \circ)_4(_1\circ\circ)_1\circ\circ\circ)(_3\circ(_2)_2\circ)_3(_4\circ\circ\circ$$

The symmetry theorem, 2: idea of proof

- The construction of q'_1 and q'_2 is combinatorial:
  - Encode the pair (q_1, q_2) as a $2n$-letter word $b = (b_1, b_2, \ldots, b_{2n})$ over the 3-letter alphabet $\{), (, \circ\}$. Namely, for each i,
    - let b_{2i-1} be an opening parenthesis "$($" if $i \in q_1$, otherwise a neutral symbol "$\circ$";
    - let b_{2i} be a closing parenthesis "$)$" if $i \in q_2$, otherwise a neutral symbol "$\circ$".
  - Match parentheses in b "the usual way" but keeping in mind that the word wraps around cyclically.
  - Replace the unmatched parentheses by their duals – e.g., if they were $)$'s, make them $($'s.

    In our above example:

$$b = \circ)_4 (_1 \circ\circ)_1 \circ\circ\circ) (_3 \circ (_2)_2 \circ)_3 (_4 \circ\circ\circ$$

The symmetry theorem, 2: idea of proof

- The construction of q'_1 and q'_2 is combinatorial:
  - Encode the pair (q_1, q_2) as a $2n$-letter word $b = (b_1, b_2, \ldots, b_{2n})$ over the 3-letter alphabet $\{), (, \circ\}$. Namely, for each i,
    - let b_{2i-1} be an opening parenthesis "(" if $i \in q_1$, otherwise a neutral symbol "$\circ$";
    - let b_{2i} be a closing parenthesis ")" if $i \in q_2$, otherwise a neutral symbol "$\circ$".
  - Match parentheses in b "the usual way" but keeping in mind that the word wraps around cyclically.
  - Replace the unmatched parentheses by their duals – e.g., if they were)'s, make them ('s.

    In our above example:

$$b' = \circ)_4(_1\circ\circ)_1\circ\circ\circ((_3\circ(_2)_2\circ)_3(_4\circ\circ\circ$$

The symmetry theorem, 2: idea of proof

- The construction of q_1' and q_2' is combinatorial:
  - Encode the pair (q_1, q_2) as a $2n$-letter word $b = (b_1, b_2, \ldots, b_{2n})$ over the 3-letter alphabet $\{), (, \circ\}$. Namely, for each i,
    - let b_{2i-1} be an opening parenthesis "(" if $i \in q_1$, otherwise a neutral symbol "$\circ$";
    - let b_{2i} be a closing parenthesis ")" if $i \in q_2$, otherwise a neutral symbol "$\circ$".
  - Match parentheses in b "the usual way" but keeping in mind that the word wraps around cyclically.
  - Replace the unmatched parentheses by their duals – e.g., if they were)'s, make them ('s.
  - Turn the resulting word b' into two sets q_1' and q_2' as follows:
    - $q_1' = \{ i \in [n] \mid \text{either } b_{2i-1}' \text{ or } b_{2i}' \text{ is a "("} \}$;
    - $q_2' = \{ i \in [n] \mid \text{either } b_{2i-1}' \text{ or } b_{2i}' \text{ is a ")"} \}$.

The symmetry theorem, 3: comments

- Note that
  - if $|q_1| < |q_2|$, then q_1' is obtained from q_1 by adding some elements from q_2, whereas q_2' is obtained from q_2 by removing these elements;
  - if $|q_1| = |q_2|$, then $q_1' = q_1$ and $q_2' = q_2$;
  - if $|q_1| > |q_2|$, then q_1' is obtained from q_1 by removing some elements, whereas q_2' is obtained from q_2 by adding these elements.

The symmetry theorem, 3: comments

- Note that
  - if $|q_1| < |q_2|$, then q_1' is obtained from q_1 by adding some elements from q_2, whereas q_2' is obtained from q_2 by removing these elements;
  - if $|q_1| = |q_2|$, then $q_1' = q_1$ and $q_2' = q_2$;
  - if $|q_1| > |q_2|$, then q_1' is obtained from q_1 by removing some elements, whereas q_2' is obtained from q_2 by adding these elements.
- This is closely connected to the Lascoux-Schützenberger action of the symmetric group on words (a.k.a. the Weyl group action on the word crystal of type A).
- Note: the map $(q_1, q_2) \mapsto (q_1', q_2')$ is an involution.

A Jacobi-Trudi-like formula

- But can we compute $\langle u \rangle$ without enumerating all MLQs?

A Jacobi-Trudi-like formula

- But can we compute $\langle u \rangle$ without enumerating all MLQs?
- We have a partial answer (which subsumes two conjectures by Aas and Linusson).

- **Theorem.** Let $B = \{b_1 < b_2 < \cdots < b_r\} \subseteq [n]$.
  Let $v_1 v_2 \cdots v_r$ be a weakly decreasing (non-cyclic) packed word of length r with $\ell - 1$ classes.
  Define a word u of length n by $u_i = v_j$ if $i = b_j$ for some j, otherwise $u_i = \ell$.
  Then
  $$\langle u \rangle = \left(\prod_{i \in B} x_i \right) \det\big(h_{i-j-1+\ell-v_j}(x_1, x_2, \ldots, x_{b_j}) \big)_{1 \le i,j \le r}.$$

- **Theorem.** Let $B = \{b_1 < b_2 < \cdots < b_r\} \subseteq [n]$.
  Let $v_1 v_2 \cdots v_r$ be a weakly decreasing (non-cyclic) packed word of length r with $\ell - 1$ classes.
  Define a word u of length n by $u_i = v_j$ if $i = b_j$ for some j, otherwise $u_i = \ell$.
  Then
  $$\langle u \rangle = \left(\prod_{i \in B} x_i \right) \det\left(h_{i-j-1+\ell-v_j}(x_1, x_2, \ldots, x_{b_j}) \right)_{1 \le i,j \le r}.$$
  Example: $n = 8$ and $r = 4$ and $B = \{1 < 3 < 4 < 7\}$ and $\ell = 4$ and $v_1 v_2 \cdots v_r = 3321$. Then,
  $$\langle 3432441 \rangle = (x_1 x_3 x_4 x_7) \begin{vmatrix} h_0(x_1) & h_{-1}(x_1,x_2,x_3) & h_{-1}(x_1,x_2,x_3,x_4) & h_{-1}(x_1,x_2,\ldots,x_7) \\ h_1(x_1) & h_0(x_1,x_2,x_3) & h_0(x_1,x_2,x_3,x_4) & h_0(x_1,x_2,\ldots,x_7) \\ h_2(x_1) & h_1(x_1,x_2,x_3) & h_1(x_1,x_2,x_3,x_4) & h_1(x_1,x_2,\ldots,x_7) \\ h_3(x_1) & h_2(x_1,x_2,x_3) & h_2(x_1,x_2,x_3,x_4) & h_2(x_1,x_2,\ldots,x_7) \end{vmatrix}$$

Bonus problem

Dual stable Grothendieck polynomials

Reminder on Schur functions

- The following is not related to MLQs (or is it?), but a conjecture I'm very curious to hear ideas about.
  (And it's a Jacobi-Trudi type formula, too.)

Reminder on Schur functions

- The following is not related to MLQs (or is it?), but a conjecture I'm very curious to hear ideas about.
  (And it's a Jacobi-Trudi type formula, too.)
- Fix a commutative ring $\mathbf{k}$.
  Recall that for any skew partition λ/μ, the *(skew) Schur function $s_{\lambda/\mu}$* is defined as the power series

$$\sum_{T \text{ is an SST of shape } \lambda/\mu} \mathbf{x}^{\operatorname{cont} T} \in \mathbf{k}[[x_1, x_2, x_3, \ldots]],$$

  where "SST" is short for "semistandard Young tableau", and where

$$\mathbf{x}^{\operatorname{cont} T} = \prod_{k \geq 1} x_k^{\text{number of times } T \text{ contains entry } k}.$$

Reminder on Schur functions

- The following is not related to MLQs (or is it?), but a conjecture I'm very curious to hear ideas about. (And it's a Jacobi-Trudi type formula, too.)
- Fix a commutative ring $\mathbf{k}$. Recall that for any skew partition λ/μ, the *(skew) Schur function $s_{\lambda/\mu}$* is defined as the power series

$$\sum_{T \text{ is an SST of shape } \lambda/\mu} \mathbf{x}^{\operatorname{cont} T} \in \mathbf{k}\left[\left[x_1, x_2, x_3, \ldots\right]\right],$$

where "SST" is short for "semistandard Young tableau", and where

$$\mathbf{x}^{\operatorname{cont} T} = \prod_{k \geq 1} x_k^{\text{number of times } T \text{ contains entry } k}.$$

- Let us generalize this by extending the sum and introducing extra parameters.

Dual stable Grothendieck polynomials, 1: RPPs

- A *reverse plane partition (RPP)* is defined like an SST (semistandard Young tableau), but entries increase **weakly** both along rows and down columns. For example,

	1	2	2
	2	2	
2	4		

is an RPP.

Dual stable Grothendieck polynomials, 1: RPPs

- A *reverse plane partition (RPP)* is defined like an SST (semistandard Young tableau), but entries increase **weakly** both along rows and down columns. For example,

	1	2	2
	2	2	
2	4		

is an RPP.

(In detail: An RPP is a map T from a skew Young diagram to $\{$positive integers$\}$ such that $T(i,j) \leq T(i,j+1)$ and $T(i,j) \leq T(i+1,j)$ whenever these are defined.)

- A *reverse plane partition (RPP)* is defined like an SST (semistandard Young tableau), but entries increase **weakly** both along rows and down columns. For example,

	1	2	2
	2	2	
2	4		

is an RPP.

(In detail: An RPP is a map T from a skew Young diagram to $\{\text{positive integers}\}$ such that $T(i,j) \leq T(i,j+1)$ and $T(i,j) \leq T(i+1,j)$ whenever these are defined.)

- Let $\mathbf{k}$ be a commutative ring, and fix any elements $t_1, t_2, t_3, \ldots \in \mathbf{k}$.

Dual stable Grothendieck polynomials, 2: definition

- Given a skew partition λ/μ, we define the *refined dual stable Grothendieck polynomial* $\widetilde{g}_{\lambda/\mu}$ to be the formal power series

$$\sum_{T \text{ is an RPP of shape } \lambda/\mu} \mathbf{x}^{\operatorname{ircont} T} \mathbf{t}^{\operatorname{ceq} T} \in \mathbf{k}[[x_1, x_2, x_3, \ldots]],$$

where

$$\mathbf{x}^{\operatorname{ircont} T} = \prod_{k \geq 1} x_k^{\text{number of columns of } T \text{ containing entry } k}$$

and

$$\mathbf{t}^{\operatorname{ceq} T} = \prod_{i \geq 1} t_i^{\text{number of } j \text{ such that } T(i,j) = T(i+1,j)}$$

(where $T(i,j) = T(i+1,j)$ implies, in particular, that both (i,j) and $(i+1,j)$ are cells of T).
This is a formal power series in $x_1, x_2, x_3, \ldots$ (despite the name "polynomial").

Dual stable Grothendieck polynomials, 3: examples on $\mathbf{x}^{\text{ircont } T}$

- Recall:

$$\mathbf{x}^{\text{ircont } T} = \prod_{k \geq 1} x_k^{\text{number of columns of } T \text{ containing entry } k}.$$

- If $T = \begin{array}{|c|c|c|} \hline & 1 & 2 & 2 \\ \hline & 2 & 2 & \\ \hline 2 & 3 & & \\ \hline \end{array}$, then $\mathbf{x}^{\text{ircont } T} = x_1 x_2^4 x_3$. The x_2 has exponent 4, not 5, because the two 2's in column 3 count only once.

Dual stable Grothendieck polynomials, 3: examples on $\mathbf{x}^{\operatorname{ircont} T}$

- Recall:

$$\mathbf{x}^{\operatorname{ircont} T} = \prod_{k \geq 1} x_k^{\text{number of columns of } T \text{ containing entry } k}.$$

- If $T = \begin{array}{|c|c|c|c|} \hline & 1 & 2 & 2 \\ \hline & 2 & 2 & \\ \hline 2 & 3 & & \\ \hline \end{array}$, then $\mathbf{x}^{\operatorname{ircont} T} = x_1 x_2^4 x_3$. The x_2 has exponent 4, not 5, because the two 2's in column 3 count only once.
- If T is an SST, then $\mathbf{x}^{\operatorname{ircont} T} = \mathbf{x}^{\operatorname{cont} T}$.

Dual stable Grothendieck polynomials, 3: examples on $\mathbf{t}^{\operatorname{ceq} T}$

- Recall that

$$\mathbf{t}^{\operatorname{ceq} T} = \prod_{i \geq 1} t_i^{\text{number of } j \text{ such that } T(i,j) = T(i+1,j)}$$

- If $T = \begin{array}{|c|c|c|c|} \cline{2-4} \multicolumn{1}{c|}{} & 1 & 2 & 2 \\ \cline{2-4} \multicolumn{1}{c|}{} & 2 & 2 & \multicolumn{1}{c}{} \\ \hline 2 & 3 & \multicolumn{2}{c}{} \\ \cline{1-2} \end{array}$, then $\mathbf{t}^{\operatorname{ceq} T} = t_1$, due to $T(1,3) = T(2,3)$.

Dual stable Grothendieck polynomials, 3: examples on $\mathbf{t}^{\operatorname{ceq} T}$

- Recall that

$$\mathbf{t}^{\operatorname{ceq} T} = \prod_{i \geq 1} t_i^{\text{number of } j \text{ such that } T(i,j) = T(i+1,j)}$$

- If $T = \begin{array}{|c|c|c|c|} \hline & 1 & 2 & 2 \\ \hline & 2 & 2 & \\ \hline 2 & 3 & & \\ \hline \end{array}$, then $\mathbf{t}^{\operatorname{ceq} T} = t_1$, due to $T(1,3) = T(2,3)$.
- If T is an SST, then $\mathbf{t}^{\operatorname{ceq} T} = 1$.
- In general, $\mathbf{t}^{\operatorname{ceq} T}$ measures "how often" T breaks the SST condition.

- If we set $t_1 = t_2 = t_3 = \cdots = 0$, then $\widetilde{g}_{\lambda/\mu} = s_{\lambda/\mu}$.

Dual stable Grothendieck polynomials, 5

- If we set $t_1 = t_2 = t_3 = \cdots = 0$, then $\widetilde{g}_{\lambda/\mu} = s_{\lambda/\mu}$.
- If we set $t_1 = t_2 = t_3 = \cdots = 1$, then $\widetilde{g}_{\lambda/\mu} = g_{\lambda/\mu}$, the *dual stable Grothendieck polynomial* of Lam and Pylyavskyy (arXiv:0705.2189).
- The general case, to our knowledge, is new.

Dual stable Grothendieck polynomials, 5

- If we set $t_1 = t_2 = t_3 = \cdots = 0$, then $\widetilde{g}_{\lambda/\mu} = s_{\lambda/\mu}$.
- If we set $t_1 = t_2 = t_3 = \cdots = 1$, then $\widetilde{g}_{\lambda/\mu} = g_{\lambda/\mu}$, the *dual stable Grothendieck polynomial* of Lam and Pylyavskyy (arXiv:0705.2189).
- The general case, to our knowledge, is new.
- **Theorem (Galashin, G., Liu, arXiv:1509.03803):** The power series $\widetilde{g}_{\lambda/\mu}$ is symmetric in the x_i (not in the t_i).

- If we set $t_1 = t_2 = t_3 = \cdots = 0$, then $\widetilde{g}_{\lambda/\mu} = s_{\lambda/\mu}$.
- If we set $t_1 = t_2 = t_3 = \cdots = 1$, then $\widetilde{g}_{\lambda/\mu} = g_{\lambda/\mu}$, the *dual stable Grothendieck polynomial* of Lam and Pylyavskyy (arXiv:0705.2189).
- The general case, to our knowledge, is new.
- **Theorem (Galashin, G., Liu, arXiv:1509.03803):** The power series $\widetilde{g}_{\lambda/\mu}$ is symmetric in the x_i (not in the t_i).
- **Example 1:** If $\lambda = (n)$ and $\mu = ()$, then $\widetilde{g}_{\lambda/\mu} = h_n$, the n-th complete homogeneous symmetric function.

- If we set $t_1 = t_2 = t_3 = \cdots = 0$, then $\widetilde{g}_{\lambda/\mu} = s_{\lambda/\mu}$.
- If we set $t_1 = t_2 = t_3 = \cdots = 1$, then $\widetilde{g}_{\lambda/\mu} = g_{\lambda/\mu}$, the *dual stable Grothendieck polynomial* of Lam and Pylyavskyy (arXiv:0705.2189).
- The general case, to our knowledge, is new.
- **Theorem (Galashin, G., Liu, arXiv:1509.03803):** The power series $\widetilde{g}_{\lambda/\mu}$ is symmetric in the x_i (not in the t_i).
- **Example 1:** If $\lambda = (n)$ and $\mu = ()$, then $\widetilde{g}_{\lambda/\mu} = h_n$, the n-th complete homogeneous symmetric function.
- **Example 2:** If $\lambda = \left(\underbrace{1, 1, \ldots, 1}_{n \text{ ones}}\right)$ and $\mu = ()$, then $\widetilde{g}_{\lambda/\mu} = e_n(t_1, t_2, \ldots, t_{n-1}, x_1, x_2, x_3, \ldots)$, where e_n is the n-th elementary symmetric function.

- If we set $t_1 = t_2 = t_3 = \cdots = 0$, then $\widetilde{g}_{\lambda/\mu} = s_{\lambda/\mu}$.
- If we set $t_1 = t_2 = t_3 = \cdots = 1$, then $\widetilde{g}_{\lambda/\mu} = g_{\lambda/\mu}$, the *dual stable Grothendieck polynomial* of Lam and Pylyavskyy (arXiv:0705.2189).
- The general case, to our knowledge, is new.
- **Theorem (Galashin, G., Liu, arXiv:1509.03803):** The power series $\widetilde{g}_{\lambda/\mu}$ is symmetric in the x_i (not in the t_i).
- **Example 1:** If $\lambda = (n)$ and $\mu = ()$, then $\widetilde{g}_{\lambda/\mu} = h_n$, the n-th complete homogeneous symmetric function.
- **Example 2:** If $\lambda = \left(\underbrace{1, 1, \ldots, 1}_{n \text{ ones}}\right)$ and $\mu = ()$, then $\widetilde{g}_{\lambda/\mu} = e_n(t_1, t_2, \ldots, t_{n-1}, x_1, x_2, x_3, \ldots)$, where e_n is the n-th elementary symmetric function.
- **Example 3:** If $\lambda = (2, 1)$ and $\mu = ()$, then $\widetilde{g}_{\lambda/\mu} = \sum_{a \leq b;\ a < c} x_a x_b x_c + t_1 \sum_{a \leq b} x_a x_b = s_{(2,1)} + t_1 s_{(2)}$.

Jacobi-Trudi identity?

- **Conjecture:** Let the conjugate partitions of λ and μ be $\lambda^t = ((\lambda^t)_1, (\lambda^t)_2, \ldots, (\lambda^t)_N)$ and $\mu^t = ((\mu^t)_1, (\mu^t)_2, \ldots, (\mu^t)_N)$. Then,

$$\widetilde{g}_{\lambda/\mu}$$

$$= \det\left(\left(e_{(\lambda^t)_i - i - (\mu^t)_j + j}\left(\mathbf{x}, \mathbf{t}\left[\left(\mu^t\right)_j + 1 : \left(\lambda^t\right)_i\right]\right)\right)_{1\leq i\leq N,\ 1\leq j\leq N}\right).$$

Here, $(\mathbf{x}, \mathbf{t}[k : \ell])$ denotes the alphabet $(x_1, x_2, x_3, \ldots, t_k, t_{k+1}, \ldots, t_{\ell-1})$.
Warning: If $\ell \leq k$, then $t_k, t_{k+1}, \ldots, t_{\ell-1}$ means nothing. No "antimatter" variables!

Jacobi-Trudi identity?

- **Conjecture:** Let the conjugate partitions of λ and μ be $\lambda^t = \left((\lambda^t)_1, (\lambda^t)_2, \ldots, (\lambda^t)_N\right)$ and $\mu^t = \left((\mu^t)_1, (\mu^t)_2, \ldots, (\mu^t)_N\right)$. Then,

  $$\widetilde{g}_{\lambda/\mu} = \det\left(\left(e_{(\lambda^t)_i - i - (\mu^t)_j + j}\left(\mathbf{x}, \mathbf{t}\left[\left(\mu^t\right)_j + 1 : \left(\lambda^t\right)_i\right]\right)\right)_{1\leq i\leq N,\ 1\leq j\leq N}\right).$$

  Here, $(\mathbf{x}, \mathbf{t}[k : \ell])$ denotes the alphabet $(x_1, x_2, x_3, \ldots, t_k, t_{k+1}, \ldots, t_{\ell-1})$.
  Warning: If $\ell \leq k$, then $t_k, t_{k+1}, \ldots, t_{\ell-1}$ means nothing. No "antimatter" variables!
- This would generalize the Jacobi-Trudi identity for Schur functions in terms of e_i's.

Jacobi-Trudi identity?

- **Conjecture:** Let the conjugate partitions of λ and μ be $\lambda^t = ((\lambda^t)_1, (\lambda^t)_2, \ldots, (\lambda^t)_N)$ and $\mu^t = ((\mu^t)_1, (\mu^t)_2, \ldots, (\mu^t)_N)$. Then,

  $$\widetilde{g}_{\lambda/\mu} = \det\left(\left(e_{(\lambda^t)_i - i - (\mu^t)_j + j}\left(\mathbf{x}, \mathbf{t}\left[\left(\mu^t\right)_j + 1 : \left(\lambda^t\right)_i\right]\right)\right)_{1\leq i\leq N,\ 1\leq j\leq N}\right).$$

  Here, $(\mathbf{x}, \mathbf{t}[k : \ell])$ denotes the alphabet $(x_1, x_2, x_3, \ldots, t_k, t_{k+1}, \ldots, t_{\ell-1})$.
  Warning: If $\ell \leq k$, then $t_k, t_{k+1}, \ldots, t_{\ell-1}$ means nothing. No "antimatter" variables!
- This would generalize the Jacobi-Trudi identity for Schur functions in terms of e_i's.
- I have some even stronger conjectures, with less evidence...

Jacobi-Trudi identity?

- **Conjecture:** Let the conjugate partitions of λ and μ be $\lambda^t = ((\lambda^t)_1, (\lambda^t)_2, \ldots, (\lambda^t)_N)$ and $\mu^t = ((\mu^t)_1, (\mu^t)_2, \ldots, (\mu^t)_N)$. Then,

$$\widetilde{g}_{\lambda/\mu} = \det\left(\left(e_{(\lambda^t)_i - i - (\mu^t)_j + j}\left(\mathbf{x}, \mathbf{t}\left[\left(\mu^t\right)_j + 1 : \left(\lambda^t\right)_i\right]\right)\right)_{1\leq i\leq N,\ 1\leq j\leq N}\right).$$

  Here, $(\mathbf{x}, \mathbf{t}[k : \ell])$ denotes the alphabet $(x_1, x_2, x_3, \ldots, t_k, t_{k+1}, \ldots, t_{\ell-1})$.
  Warning: If $\ell \leq k$, then $t_k, t_{k+1}, \ldots, t_{\ell-1}$ means nothing. No "antimatter" variables!
- This would generalize the Jacobi-Trudi identity for Schur functions in terms of e_i's.
- I have some even stronger conjectures, with less evidence...
- The case $\mu = \varnothing$ has been proven by Damir Yeliussizov in arXiv:1601.01581.

Thank you

- Christine Bessenrodt for the invitation.
- Erik Aas and Travis Scrimshaw for collaboration.
- you for attending.