Lecture 7 - March 6

- start with a leftover from Lecture 5 (last of lab complements)

Garside's solution to the word & conjugacy problems

- is given 2 words \( w, w' \) in \( \sigma_i \),
  - WP: decide if \( w, w' \) = same elt of \( B_n \)
  - CP: decide if \( w, w' \) conjugates of each other.

Garside 1969

Thurston

Many improvements since
- El Rifi - Norton early 90s - vastly improved soln
- Birman - Ko - Lee 1997 - "hard germs", even later?

Motivation:
- computing with braids
- first step to solving pl. of recognising when 2 links are isotopic (Markov's thm?)

Garside case: 0) the semigroup of positive braids
- observe: the presentation of \( B_n \) doesn't involve \( \sigma_i^{-1} \)
- can use the same relations to define a semigroup (or monoid)

**Def:**
\[
B^+_n = \langle \sigma_1, \ldots, \sigma_{n-1} \mid \sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1}, \text{ relabel same as in } B_n \rangle
\]

So: two words in \( \sigma_1, \ldots, \sigma_{n-1} \) define the same elt of \( B^+_n \) iff

- can pass from one to the other by successive substitutions

- \( \sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1} \), etc.

Then:
**Def:**
\[
D(W) := \{ \text{all words obtained from } W \text{ by these operations} \}
\]

is finite because transformation preserves word length
& only finitely many possibilities for each letter.

(\( \text{easy theoretical soln to word problem}\): \( B^+_n \) \( V \equiv W \) in \( B^+_n \) iff \( D(V) = D(W) \)

- can compute \( D(V) \) by iteratively applying relations & checking if get new words ...)

Then: (Garside embedding):
**Def:**
The word map is: \( B^+_n \rightarrow B_n \) is injective.

- i.e. \( V, W \) positive braid = \( V = W \) iff \( V \equiv W \)

(so equality can be checked w/o involving inverses of generators).
A particular element (the Gaside element):

\[ \Delta = (\sigma_1 \sigma_2 \cdots \sigma_{n-1})(\sigma_1 \sigma_2 \cdots \sigma_{n-2}) \cdots (\sigma_1 \sigma_2) \sigma_1 \in B_n^+ \]

(also: 1. the half-twist rotating everything by 180° 
     2. the longest permutation braid

We'll find a normal form \( \beta = \Delta^m P \), \( m \in \mathbb{Z} \), \( P \in B_n^+ \)
for braids \( \beta \in B_n \).

This normal form is more "robust" than Artin's (depends less on labelling of strands etc...)

Proof. Garside embedding.\[\]

A lemma of Ore says that embedding follows from the following properties:

1. \( B_n^+ \) is left & right cancellative, i.e. \( AX = AY = X = Y \)
   \( xA = YA \)
2. \( B_n^+ \) is right cancellative, i.e. \( x, y \in B_n^+ \Rightarrow \exists u, v \in B_n^+ \text{ s.t. } ux = vy \).

I don't want to assume Ore's lemma, so here's a simpler proof specific to \( B_n \).

- Assume \( u, v \in B_n^+ \) equal the same elt of \( B_n \)
  
  \[ \Rightarrow \text{ can pass from } V \text{ to } W \text{ via operations: } \sigma_i \rightarrow \sigma_i \sigma_i^{-1} \]

  \[ \sigma_i \rightarrow \sigma_i \]

  \[ 
  \sigma_i \sigma_i^{-1} \rightarrow \sigma_i \sigma_i^{-1} \sigma_i \sigma_i^{-1} \rightarrow \sigma_i^{-1} \sigma_i \sigma_i^{-1} \sigma_i \sigma_i^{-1} \rightarrow \sigma_i \sigma_i^{-1} \]

  \[ \text{(That's a good fact about presentation of group } G = \langle g_1 \mid R \rangle \)

  \[ = \text{ a surgery } \langle g_1, g_1^{-1}, g_1g_1^{-1}, R \rangle \]

  Check easily: can indeed do all operations on words in \( \sigma_i^{\pm 1} \) but yield word

  \[ eg. \sigma_i^{\pm 1} \rightarrow \sigma_i \sigma_i^{-1} \rightarrow \sigma_i^{-1} \sigma_i \sigma_i^{-1} \rightarrow \sigma_i \sigma_i^{-1} \sigma_i \sigma_i^{-1} \]

  \[ \sigma_i^{-1} \sigma_i \rightarrow \sigma_i^{-1} \sigma_i \sigma_i^{-1} \sigma_i \sigma_i^{-1} \rightarrow \sigma_i \sigma_i^{-1} \sigma_i \sigma_i^{-1} \sigma_i \sigma_i^{-1} \rightarrow \sigma_i \sigma_i^{-1} \]

  \[ \sigma_i \sigma_i^{-1} \rightarrow \sigma_i \sigma_i^{-1} \]

- Observe: \( \Delta^2 \), which we know to be central in \( B_n \), is also central in \( B_n^+ \)!

  Let \( \Theta = (\sigma_1 \cdots \sigma_{n-1})^n \in B_n^+ \)

  (Don't call it \( \Delta^2 \) yet since I want show \( \Delta^2 = \Theta \) yet, although it's true...)}
\[ (1) \quad \sigma_i \cdot \Theta = \Theta \cdot \sigma_i \quad \forall i \]

Use: (\ref{eq:1}) if \( 1 \leq i \leq k \) then
\[ \sigma_i \sigma_i^\perp = \sigma_i^\perp \sigma_i \quad \forall i \leq k \quad \text{if} \quad l \leq k \text{ then } \sigma_i \sigma_l = \sigma_i^\perp \sigma_l \]
\[ \sigma_i \sigma_i^\perp \sigma_l = \sigma_i \sigma_l^\perp \sigma_i \]
\[ (\sigma_{i+1} \sigma_i \sigma_{i+2} \cdots \sigma_k)^{-1} = (\sigma_i \sigma_{i+1} \sigma_{i+2} \cdots \sigma_k)^{-1} \]
\[ = (\sigma_i \sigma_{i+1} \cdots \sigma_{i+(n-i-1)} \sigma_{i+1} \cdots \sigma_{i+(n-i-1)})^{-1} \]
\[ = (\sigma_i \sigma_{i+1} \cdots \sigma_{i+(n-i-1)})^{-1} \]
\[ = (\sigma_i \cdots \sigma_{n-i})^{-1} \sigma_{i+1} \sigma_{i+2} \cdots \sigma_{n-1} \sigma_{n-1}^{-1} \]
\[ = (\sigma_1 \cdots \sigma_{n-i})^{-1} \sigma_{i+1} \cdots \sigma_{n-1} \]

(2) \[ \forall i \exists V_i \in B^+_n \quad \Theta = \sigma_i \cdot V_i = V_i \cdot \sigma_i \]

In fact, \( \Theta = (\sigma_i \cdots \sigma_{n-i})^{-1} (\sigma_i \cdots \sigma_{n-1}) (\sigma_{i+1} \cdots \sigma_{n-i}) \cdot \sigma_i = \sigma_i \cdot V_i \)

Consider a seq of hamiltonian words in \( \sigma_i \perp \).
- Hence, let \( m = \max \) # of \( \sigma_i \perp \) opening in the given sequence
- Add each word by replacing each \( \sigma_i \perp \) by \( V_i \)
- Adding \( (\Theta)^{m-V} \), \( V = \# \) of \( V_i \) instead

Then get a sequence \( \Theta^m V \rightarrow \Theta^m W \)

where each row is \( \equiv \) obvious if row was not identity matrix
\[ A \cdot \sigma_i \perp B \Leftrightarrow A \cdot B \]
\[ \Theta \cdot \sigma_i \cdot V \cdot \tilde{B} \equiv \Theta \cdot \tilde{A} \cdot \sigma_i \cdot B \equiv \Theta^{-1} \tilde{A} \cdot B \]

\( \Theta^{-1} \)

So \( V = W \in B_n \Rightarrow \Theta^m V = \Theta^m W \) in \( B^+_n \) for some \( m \).

Just need left-cancellation \( AX = AY \Rightarrow X \equiv Y \).

PF. of left-cancellation: enough to prove it when \( A = \) a single letter (induct)

Lemma: \[ \sigma_i X \equiv \sigma_k Y \Rightarrow \text{if } i = k \text{ then } X \equiv Y \text{ (left-cancellation).} \]
\[ \text{if } i = k \text{ then } X \equiv Y \text{ (left-cancellation).} \]
\[ \text{if } i = k \text{ then } Y \equiv \sigma_k Z \]
\[ \text{if } i = k \text{ then } X \equiv \sigma_k \sigma_i Z \]
\[ \text{if } i = k \text{ then } Y \equiv \sigma_k \sigma_i Z \]
\[ \text{if } i = k \text{ then } Y \equiv \sigma_k \sigma_i Z \]
\[ \text{if } i = k \text{ then } Y \equiv \sigma_k \sigma_i Z \]
PF: simulating induction on \( \text{word length} \) of words needed to pass from \( \sigma_i x \) to \( \sigma_k y \).

- If \( x, y \) have word length 0 or 1,
- If pass \( \sigma_i x \Rightarrow \sigma_k y \) in a single step, this is obvious.
- Assume true wherever \( \text{#ops} \leq n-1 \) & for all shorter words:

\[
\begin{align*}
\text{base case, 1st step:} & \quad \sigma_i x \equiv \sigma_j w \equiv \sigma_k y, \quad \text{by induction on chain length}. \\
\text{look at 1st op:} & \quad i=j=1, \quad \text{by induction on chain length}. \\
\text{case 1:} & \quad w \equiv \ldots \equiv z, \quad y \equiv \ldots \equiv z \\
\text{case 2:} & \quad j=k \equiv \ldots \equiv z, \quad x \equiv \ldots \equiv z
\end{align*}
\]

- Can assume \( j \neq \{i, k\} \).
- Induction on chain length,
- if \( i=k \):

\[
\begin{align*}
\text{case 1:} & \quad 1-j-i \geq 2, \quad w \equiv \sigma_i \tilde{z}, \quad \tilde{x} \equiv \sigma_j \tilde{z}, \quad y \equiv \sigma_j \tilde{z}' \\
\text{induction on word length} & \quad \sigma_i \tilde{z} \equiv \sigma_j \tilde{z}', \quad \sigma_i \tilde{x} \equiv \sigma_j \tilde{x}' & \sigma_i \tilde{y} \equiv \sigma_j \tilde{y}' \\
\text{similarly} & \quad 1-j-i \equiv 1, \quad \tilde{x} \equiv \sigma_j \sigma_i \tilde{z}, \quad y \equiv \sigma_j \sigma_i \tilde{z}'
\end{align*}
\]

- Last case scenario:

\[
\begin{align*}
\text{if } 1-i \equiv 1-j \geq 2 \text{ and } 1-j-i \equiv (j-k) = 1: & \quad \tilde{x} \equiv \sigma_j \sigma_i \tilde{z}, \quad \tilde{y} \equiv \sigma_j \sigma_i \tilde{z}' \\
\text{(chain len. induction)} & \quad \tilde{x} \equiv \sigma_j \tilde{z}, \quad \tilde{y} \equiv \sigma_j \tilde{z}' \\
\text{induction again } & \quad \exists \tilde{u}, \tilde{u}' \text{ s.t. } \tilde{z} \equiv \sigma_i \tilde{u}, \quad \tilde{x} \equiv \sigma_j \tilde{u}' \\
\text{again on } \tilde{y} & \quad \tilde{y} \equiv \sigma_j \tilde{u}' \equiv \sigma_j \sigma_i \tilde{u}' \\
\text{then } & \quad \tilde{x} \equiv \sigma_j \sigma_i \tilde{u} \equiv \sigma_j \sigma_i \tilde{u}' \equiv \sigma_j \tilde{u}' \equiv \tilde{u}' \equiv \tilde{u} \equiv \tilde{u}' \\
\text{where } & \quad \tilde{y} \equiv \sigma_j \tilde{u}' \equiv \sigma_j \sigma_i \tilde{u}' \equiv \sigma_j \tilde{u}' \equiv \tilde{u}' \equiv \tilde{u} \equiv \tilde{u}'
\end{align*}
\]

All cases work like this — use word length induction to eventually find the right prefix in \( x \) & \( y \).

NB: this lemma actually says very specific things about cancellation!