
DES ENSEMBLES AUX TRESSES

DES ENSEMBLES AUX TRESSES

Patrick Dehornoy

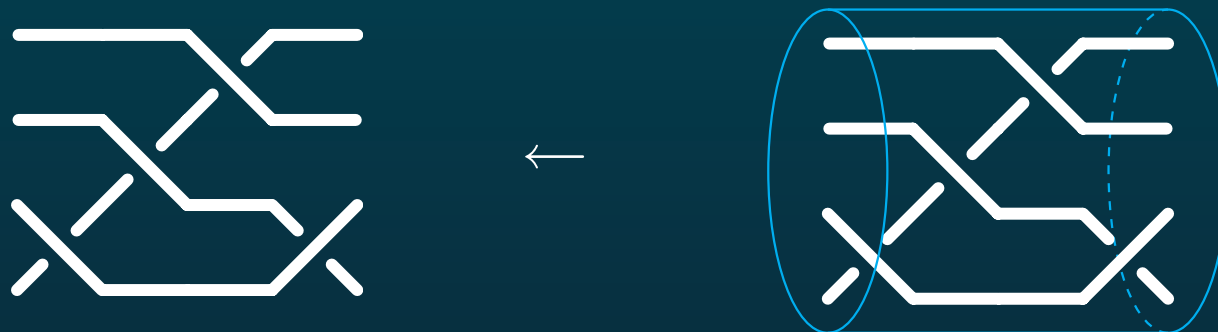


Laboratoire de Mathématiques
Nicolas Oresme, Caen

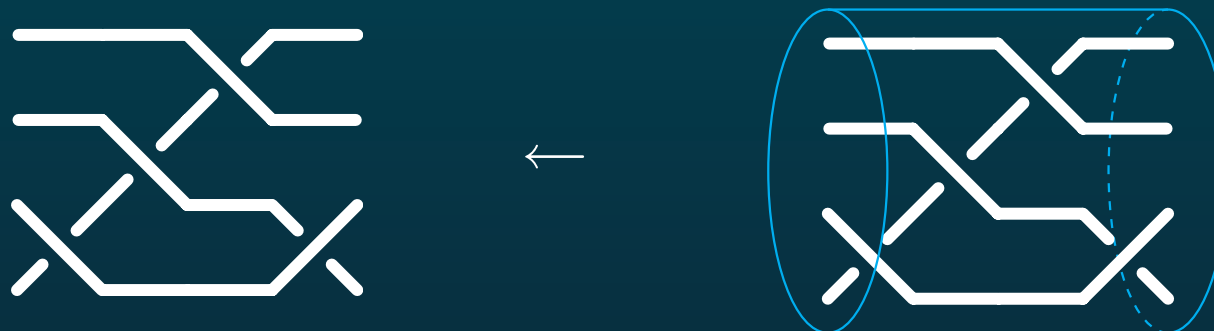
- Un **diagramme de tresse** à 4 brins



- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D

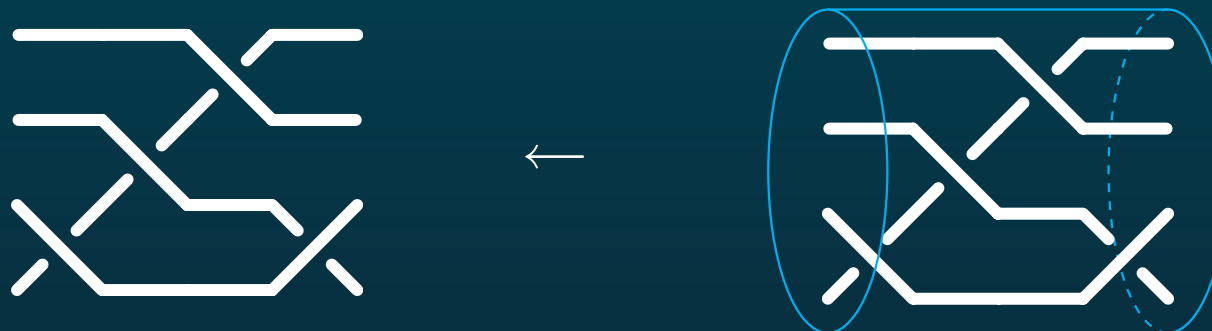


- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D



- isotopie = bouger les brins de la figure 3D en laissant les extrémités fixes

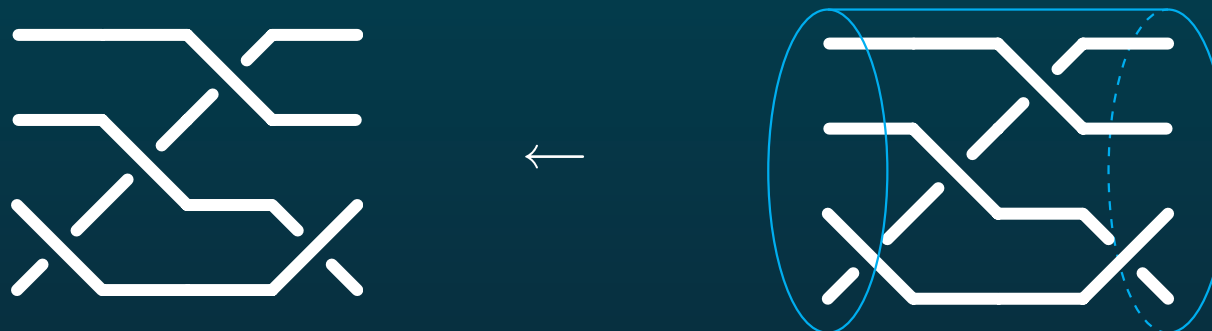
- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D



- isotopie = bouger les brins de la figure 3D en laissant les extrémités fixes



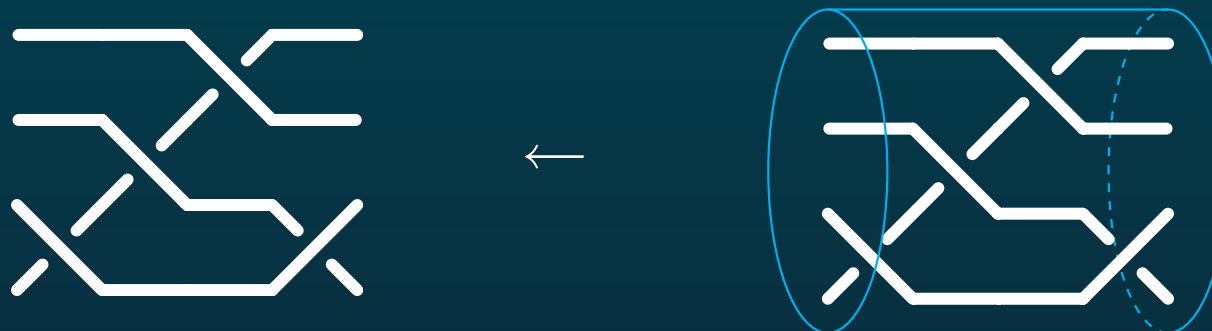
- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D



- isotopie = bouger les brins de la figure 3D en laissant les extrémités fixes



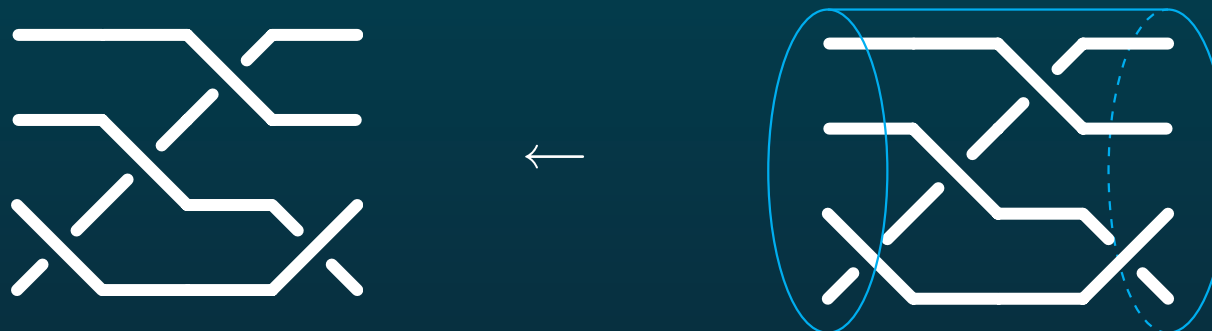
- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D



- isotopie = bouger les brins de la figure 3D en laissant les extrémités fixes



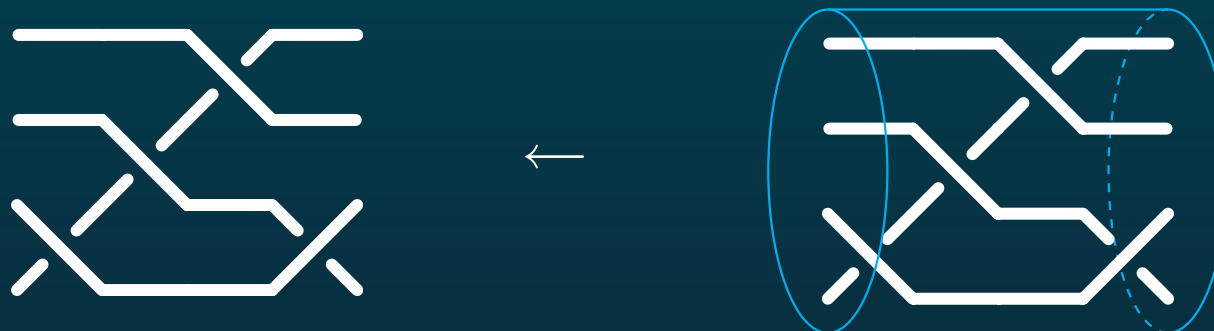
- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D



- isotopie = bouger les brins de la figure 3D en laissant les extrémités fixes



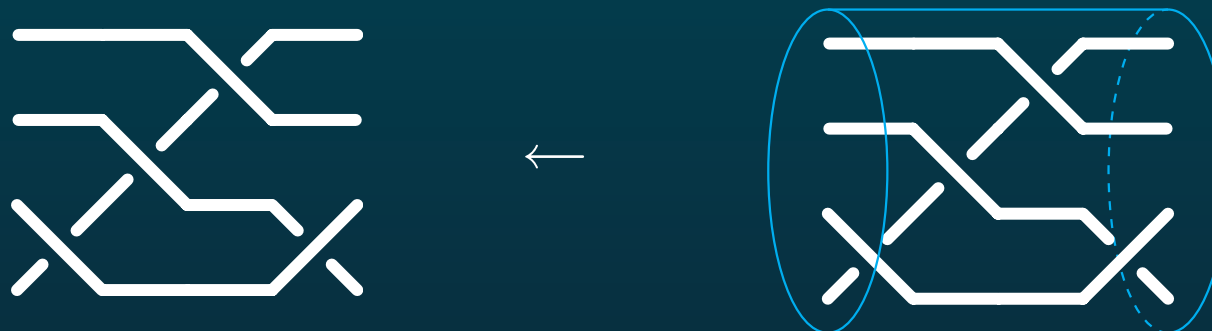
- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D



- isotopie = bouger les brins de la figure 3D en laissant les extrémités fixes



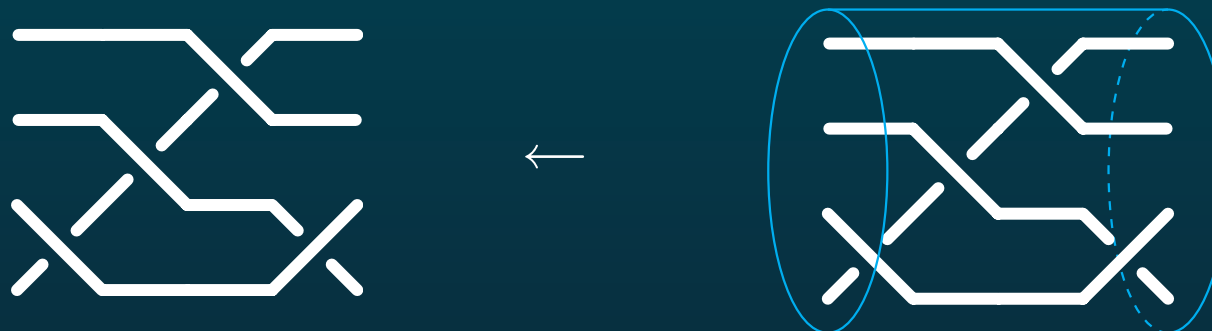
- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D



- isotopie = bouger les brins de la figure 3D en laissant les extrémités fixes



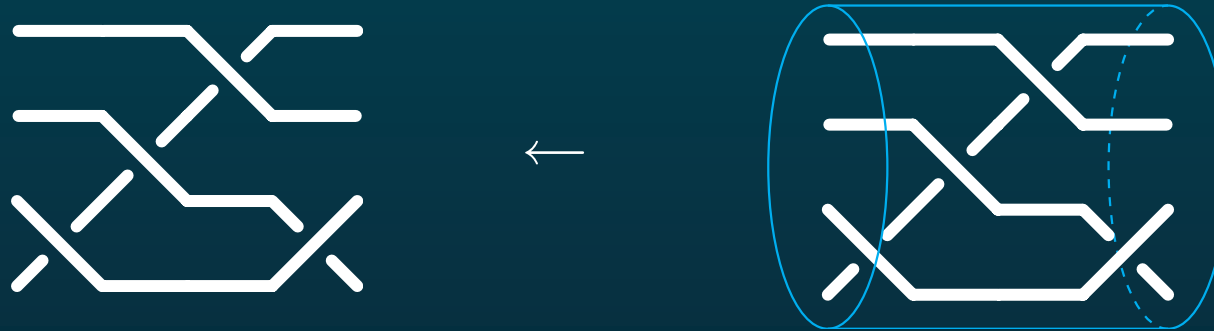
- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D



- isotopie = bouger les brins de la figure 3D en laissant les extrémités fixes



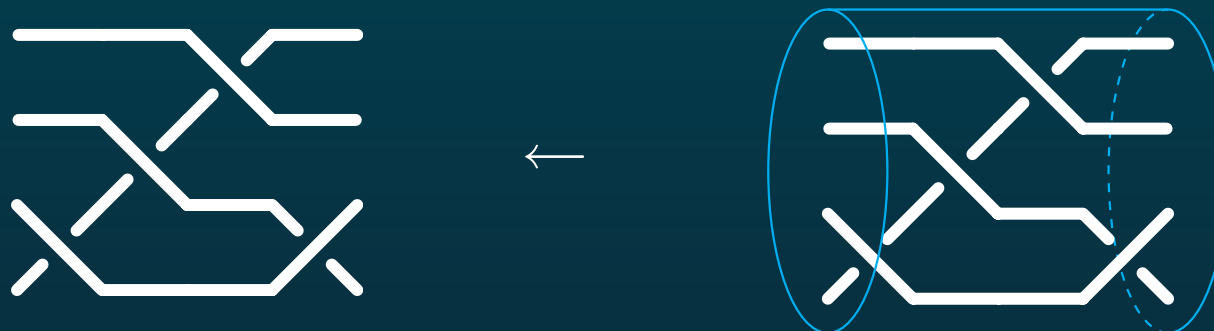
- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D



- isotopie = bouger les brins de la figure 3D en laissant les extrémités fixes



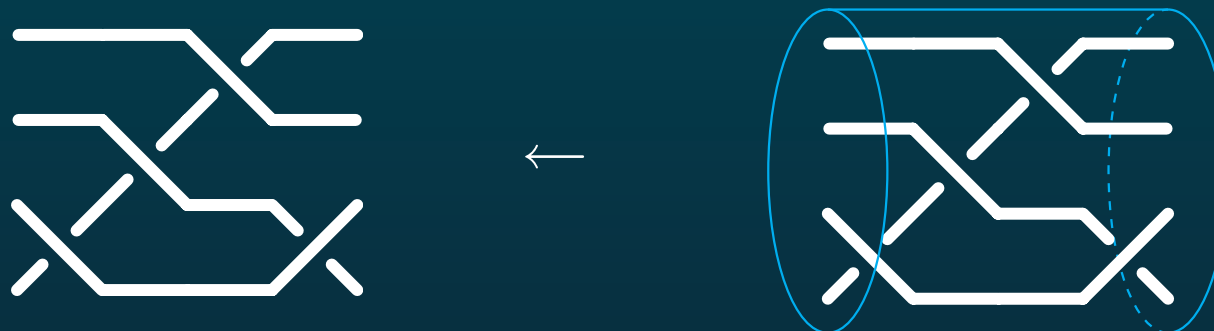
- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D



- isotopie = bouger les brins de la figure 3D en laissant les extrémités fixes



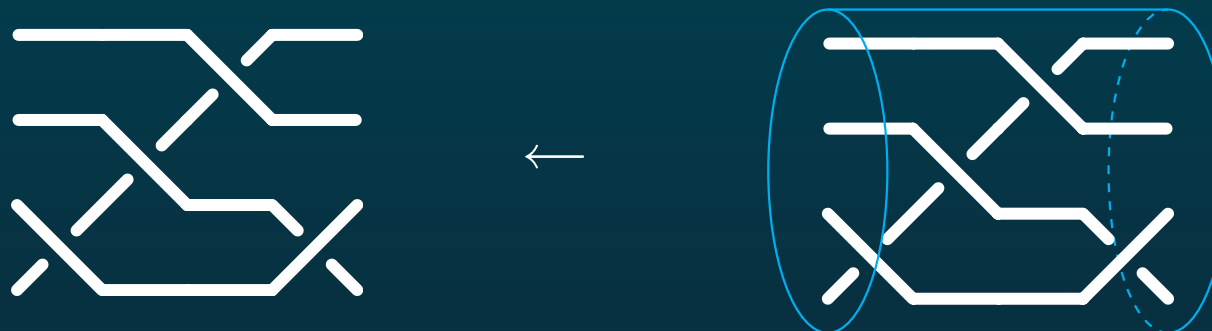
- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D



- isotopie = bouger les brins de la figure 3D en laissant les extrémités fixes



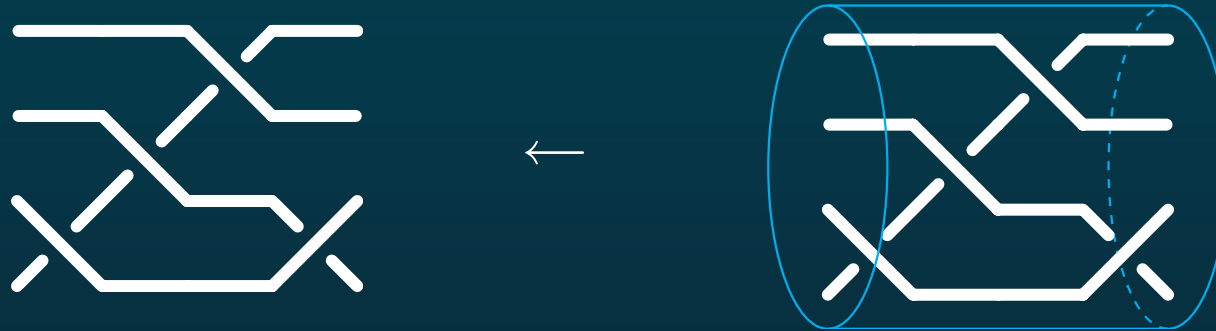
- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D



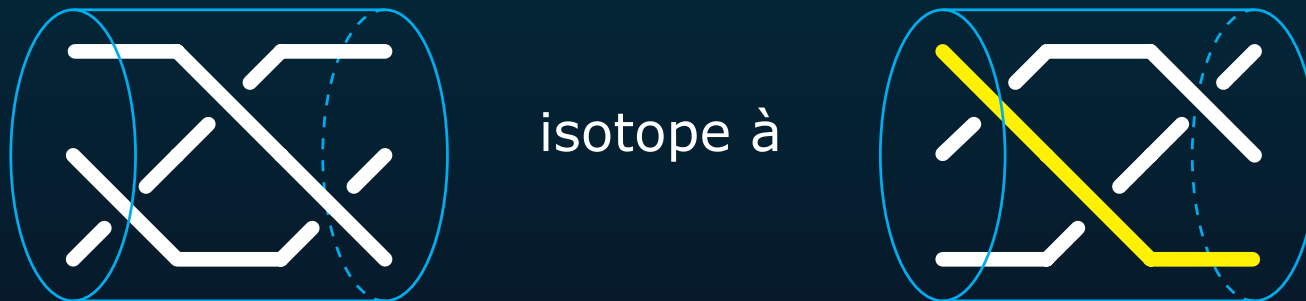
- isotopie = bouger les brins de la figure 3D en laissant les extrémités fixes



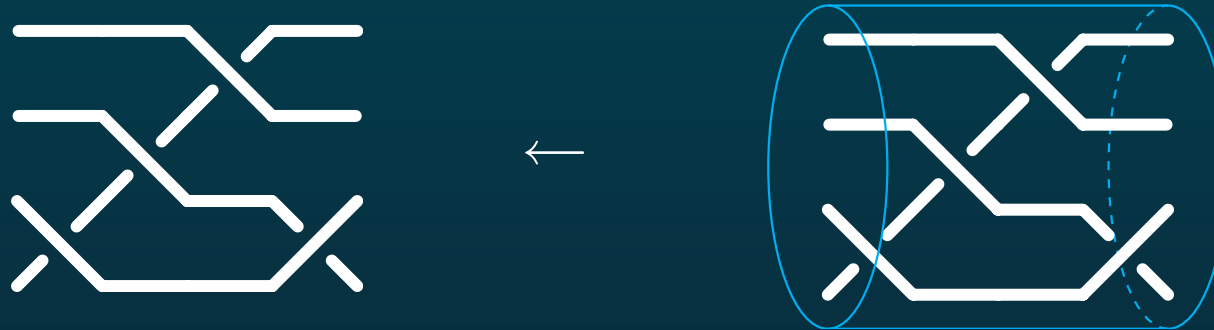
- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D



- isotopie = bouger les brins de la figure 3D en laissant les extrémités fixes



- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D

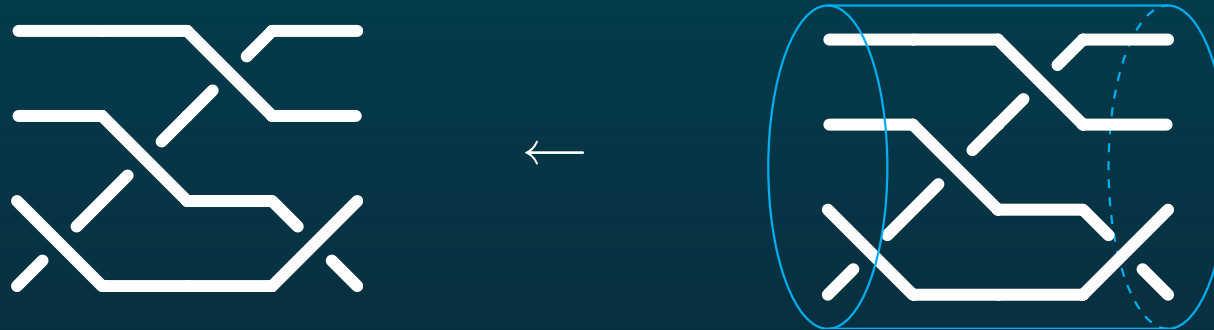


- isotopie = bouger les brins de la figure 3D en laissant les extrémités fixes



- une **tresse** = une classe d'isotopie \rightsquigarrow représentée par un diagramme 2D,

- Un **diagramme de tresse** à 4 brins = projection 2D d'une figure 3D

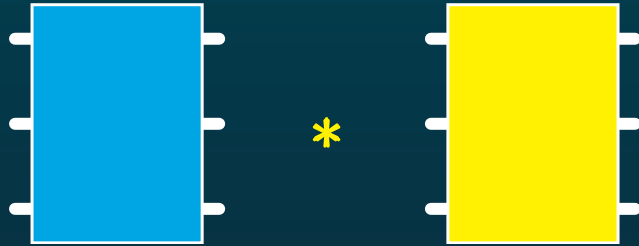


- isotopie = bouger les brins de la figure 3D en laissant les extrémités fixes

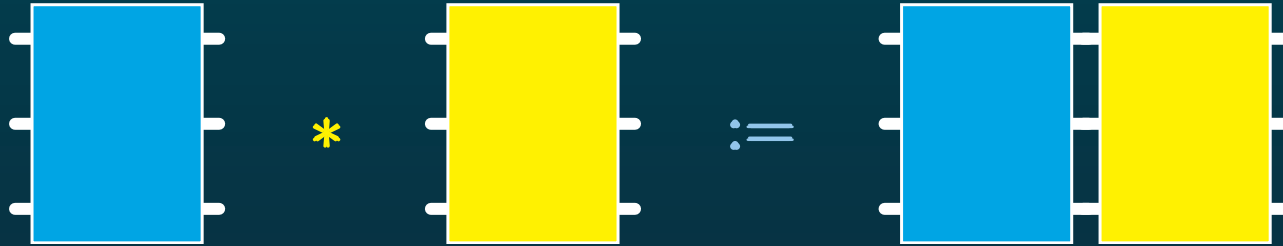


- une **tresse** = une classe d'isotopie \rightsquigarrow représentée par un diagramme 2D, **mais** différents diagrammes peuvent correspondre à la même tresse.

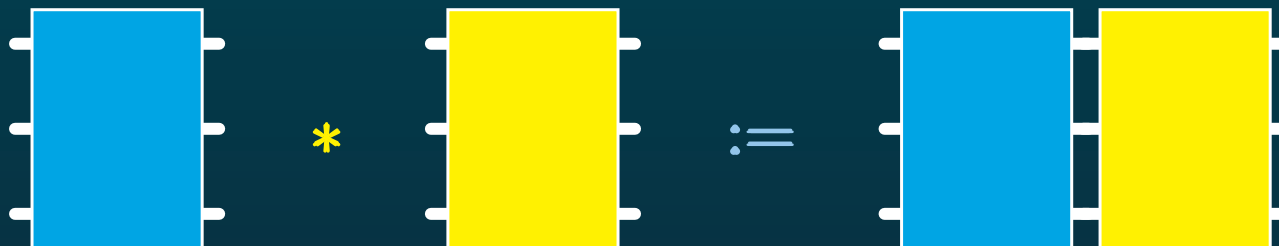
- Produit de deux tresses:



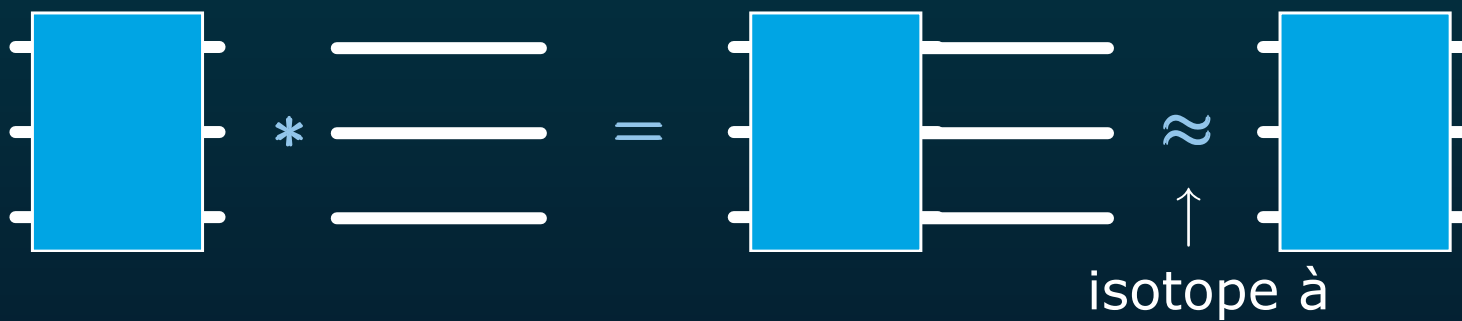
- Produit de deux tresses:



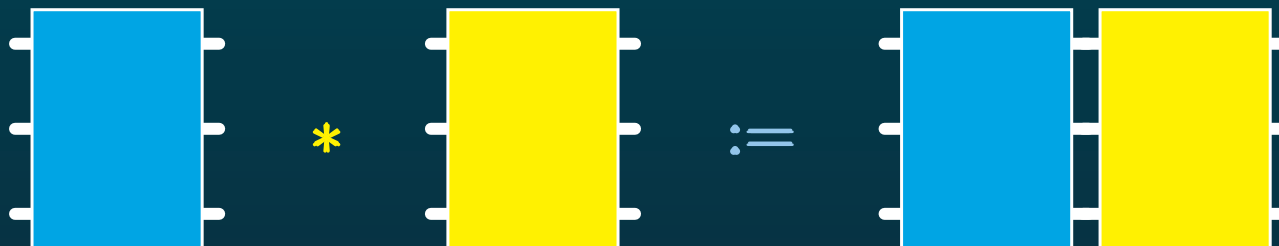
- Produit de deux tresses:



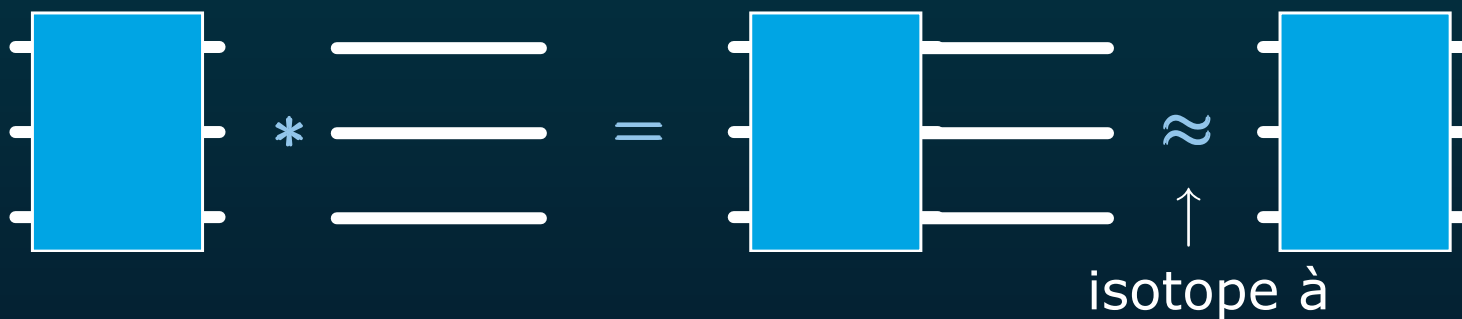
- Alors



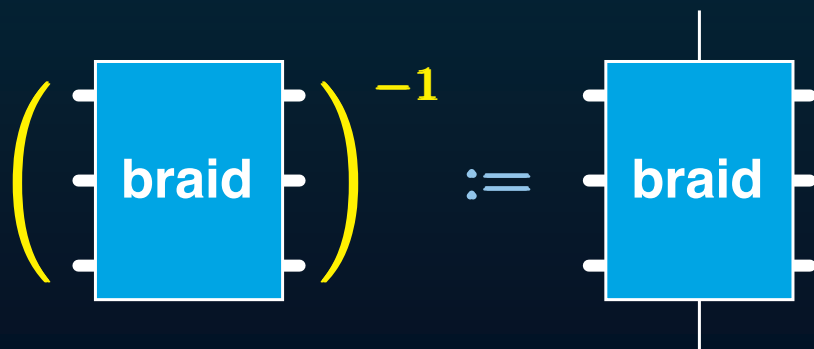
- Produit de deux tresses:



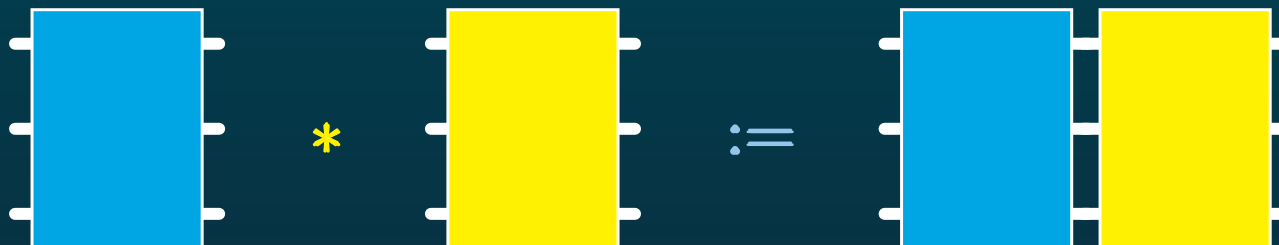
- Alors



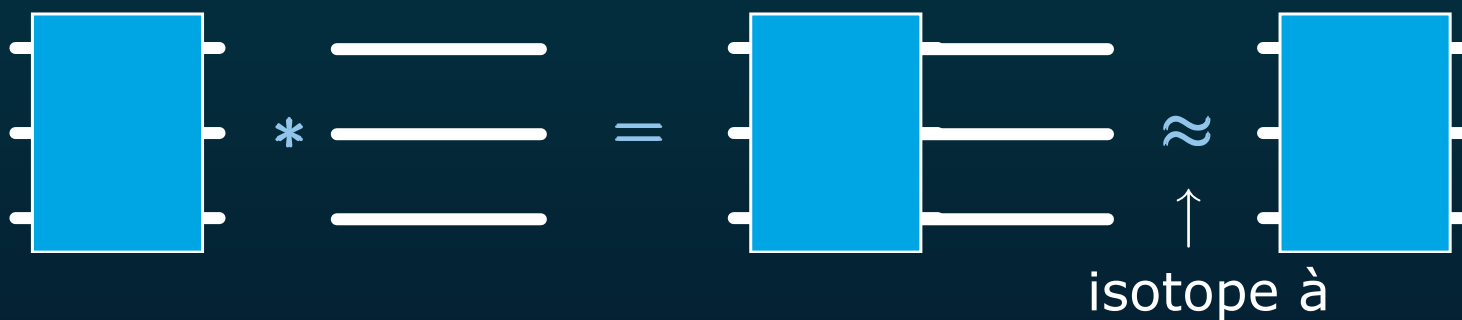
- et



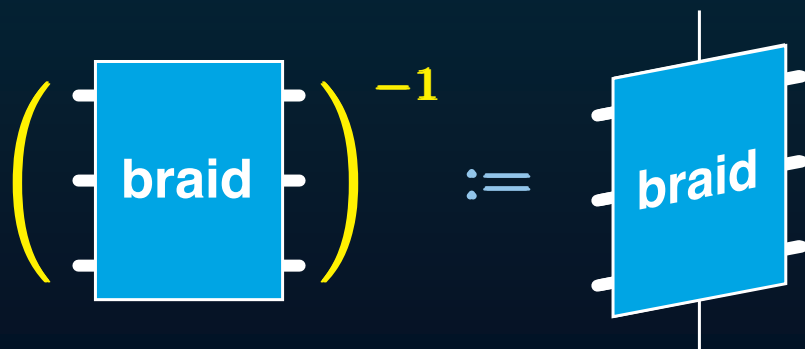
- Produit de deux tresses:



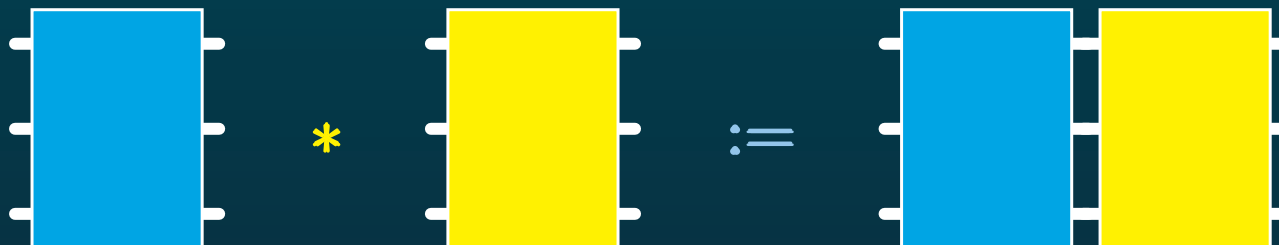
- Alors



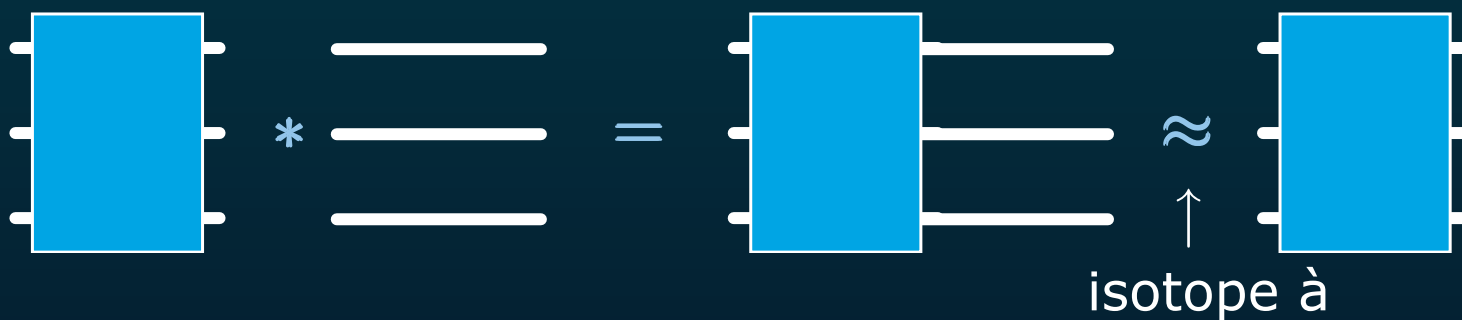
- et



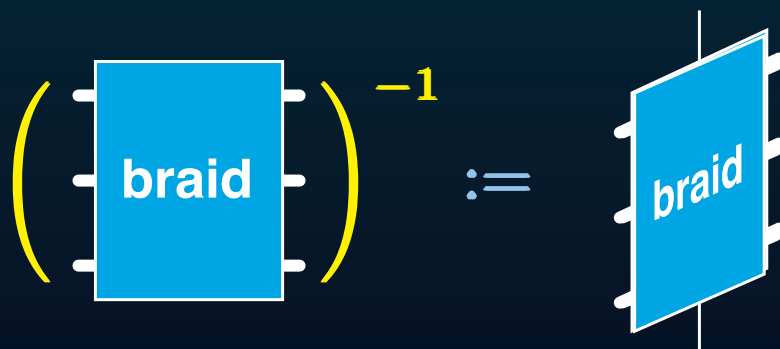
- Produit de deux tresses:



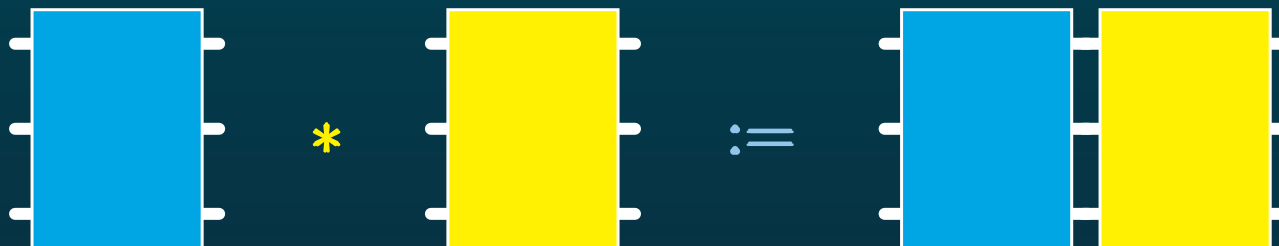
- Alors



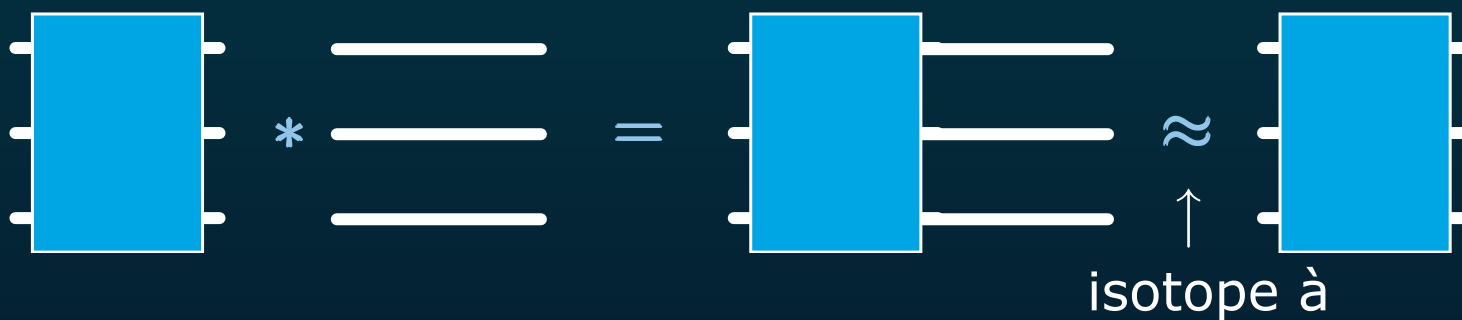
- et



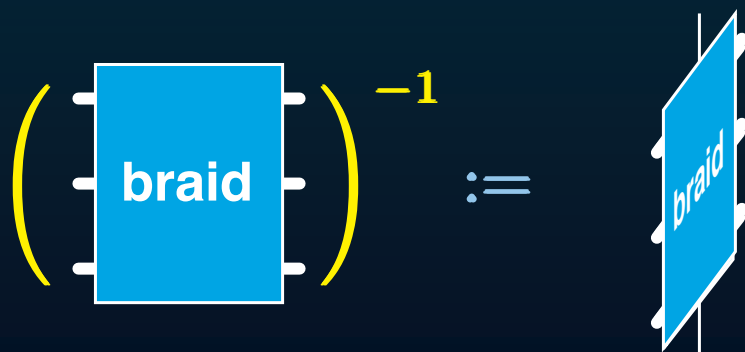
- Produit de deux tresses:



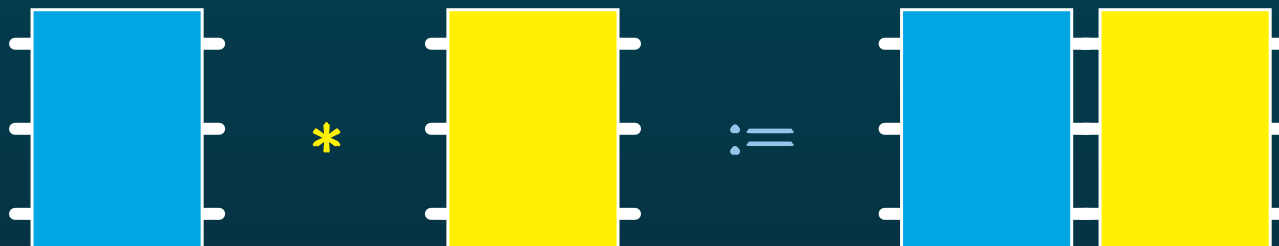
- Alors



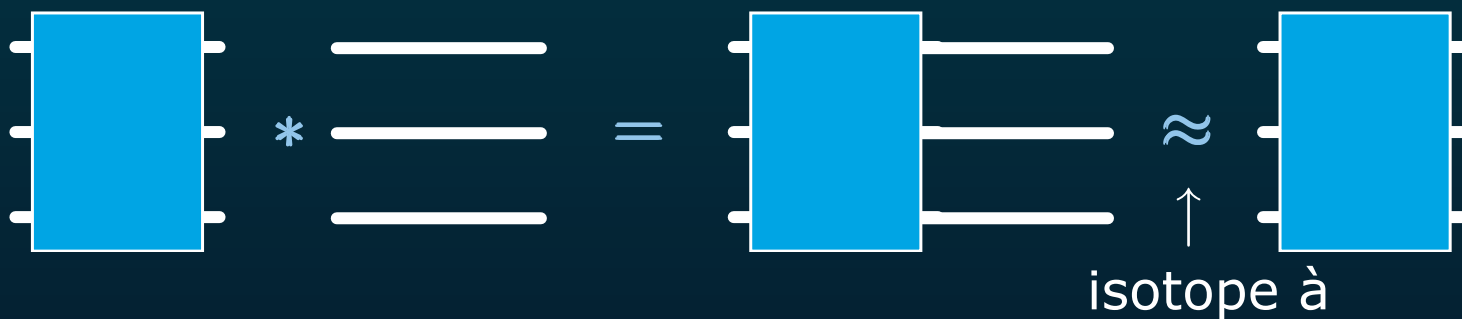
- et



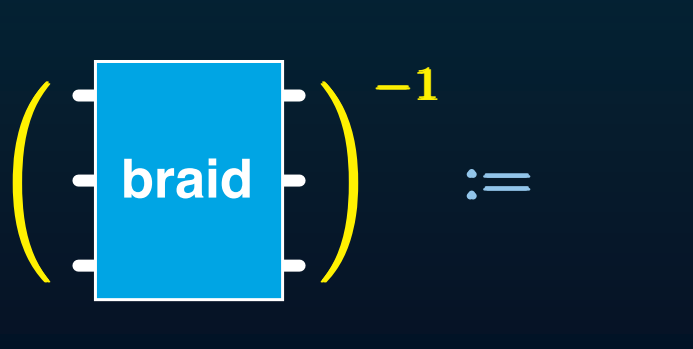
- Produit de deux tresses:



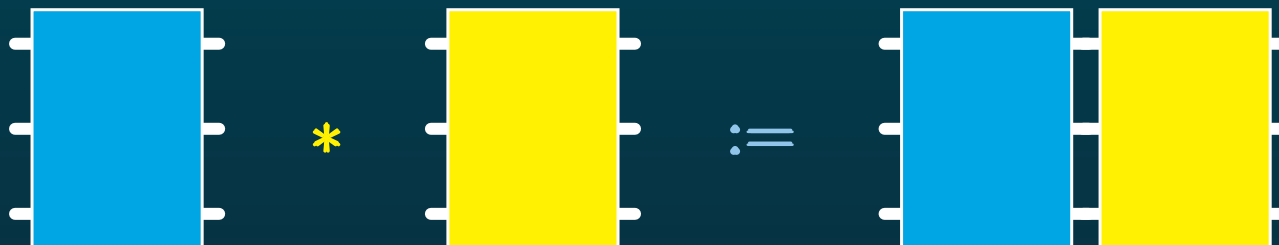
- Alors



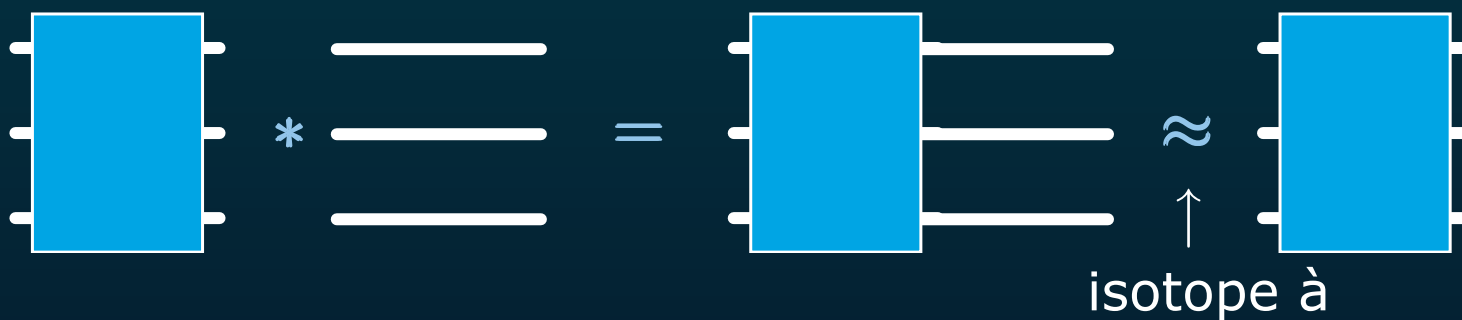
- et



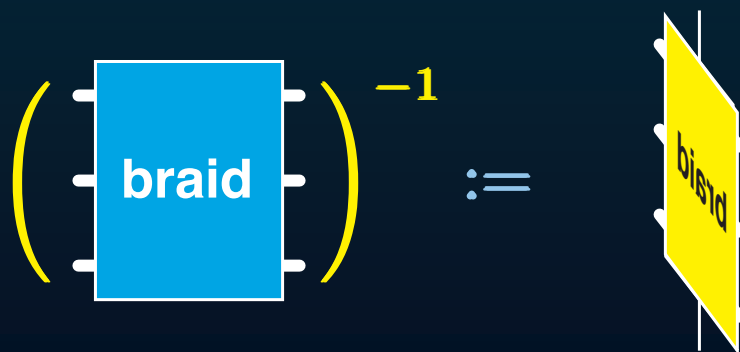
- Produit de deux tresses:



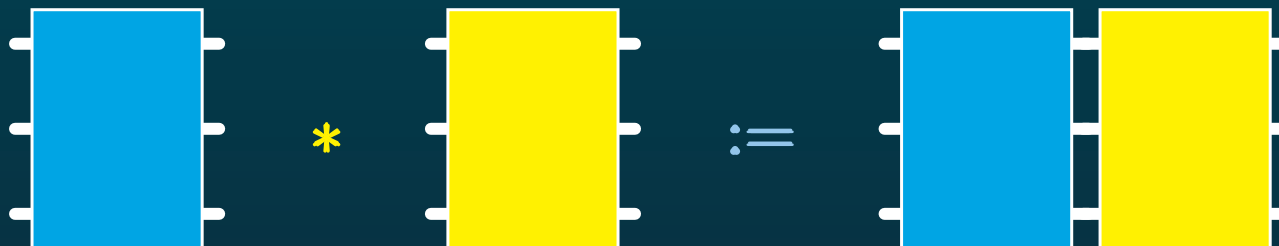
- Alors



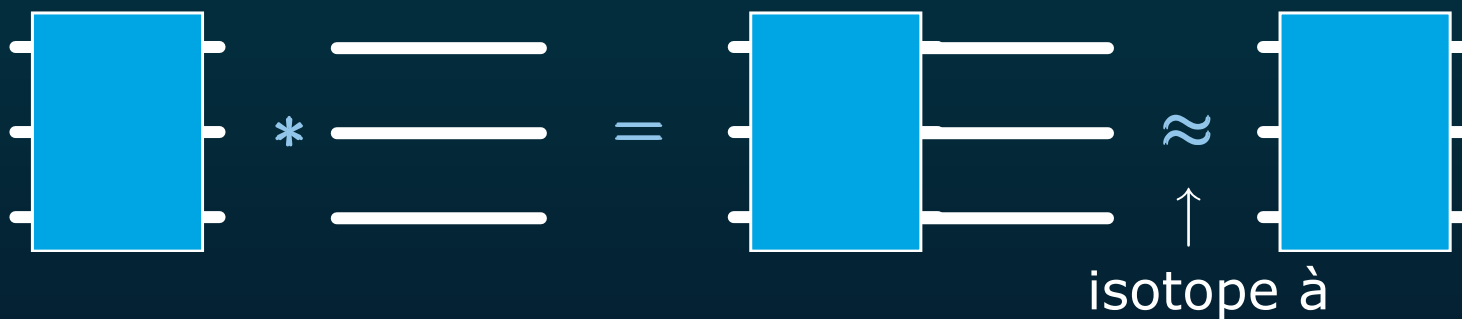
- et



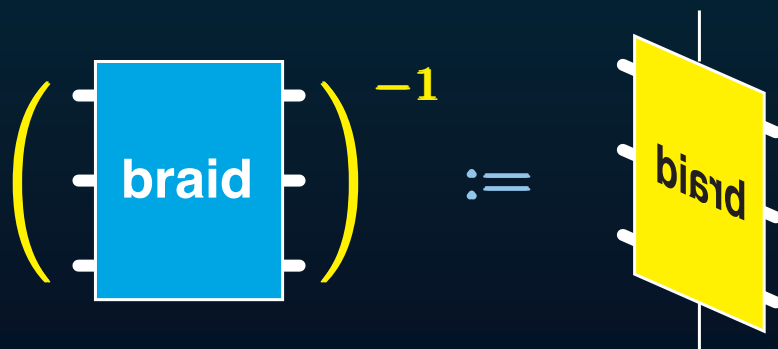
- Produit de deux tresses:



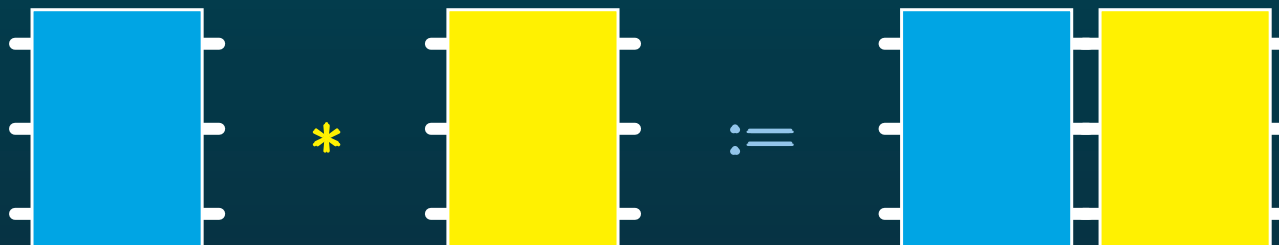
- Alors



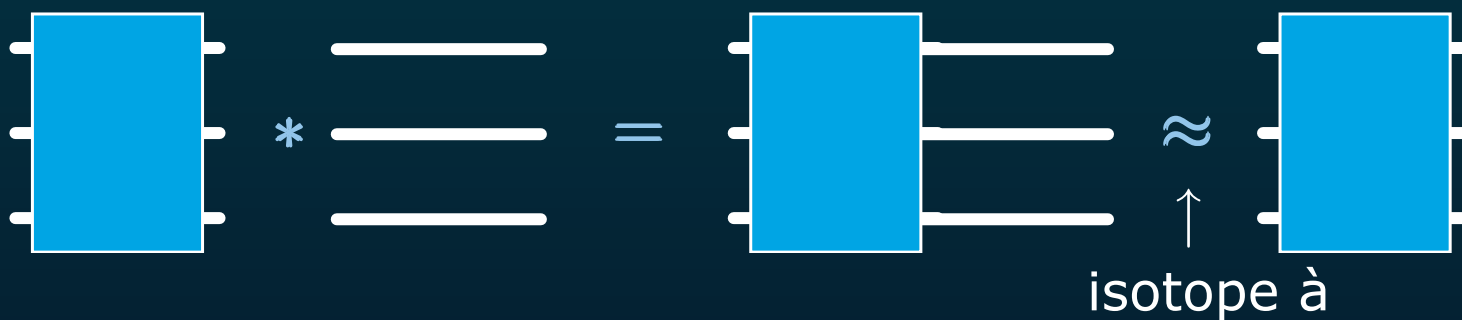
- et



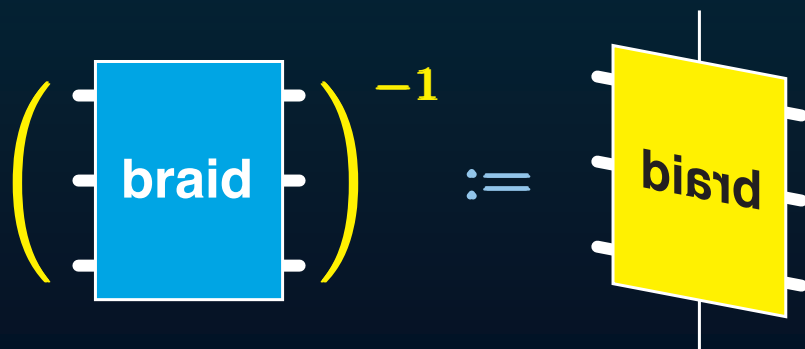
- Produit de deux tresses:



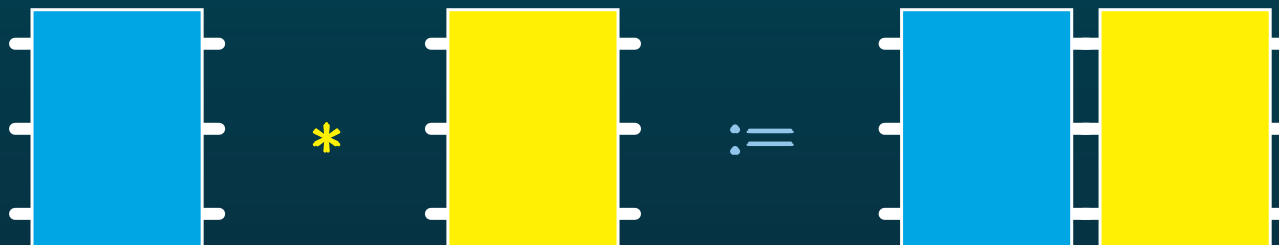
- Alors



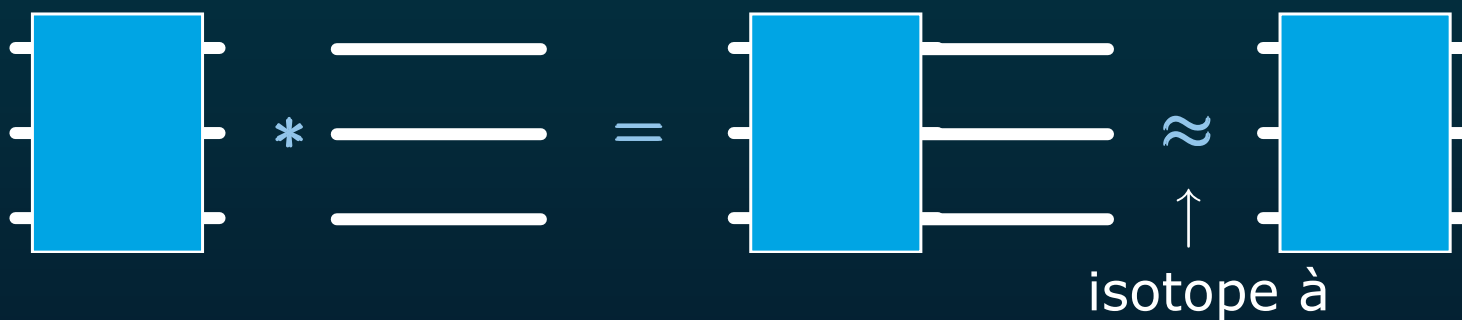
- et



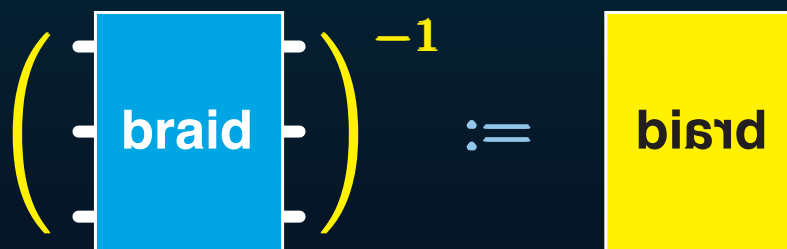
- Produit de deux tresses:



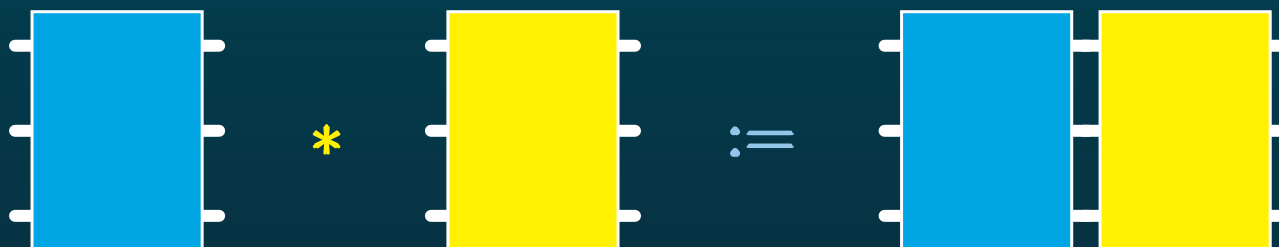
- Alors



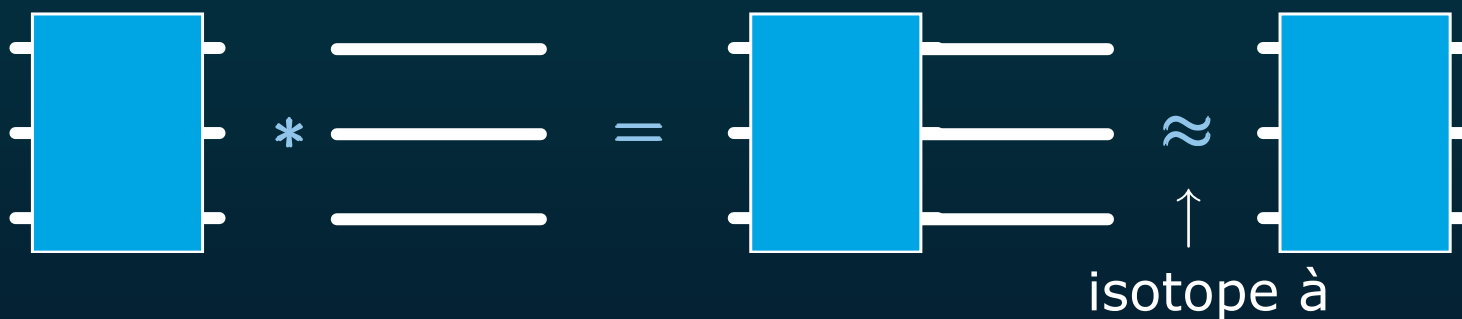
- et



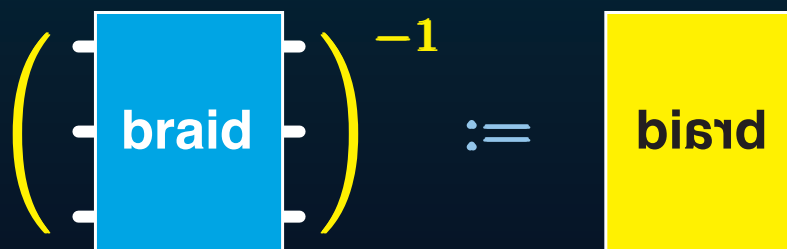
- Produit de deux tresses:



- Alors



- et

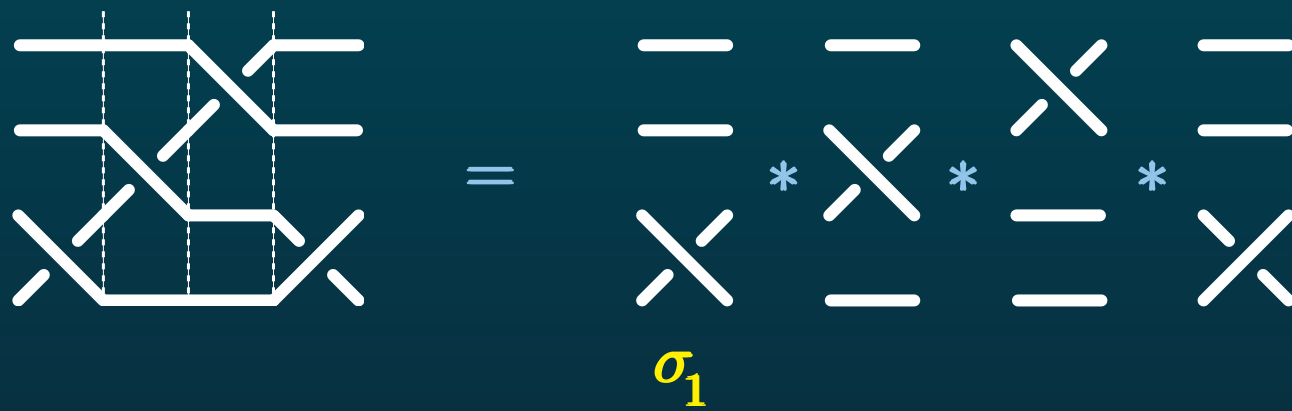


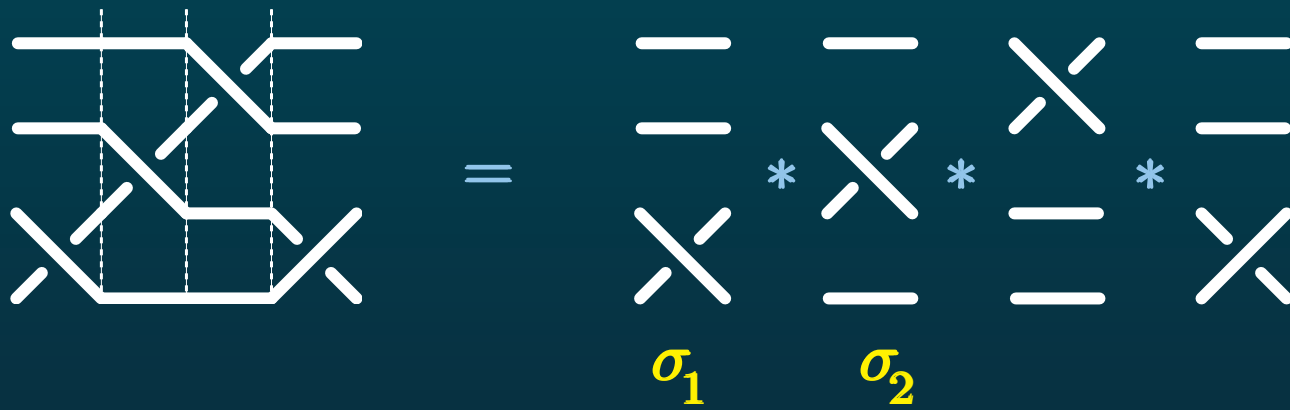
↪ Pour chaque n , le groupe B_n des tresses à n brins (E. Artin, ~1925).

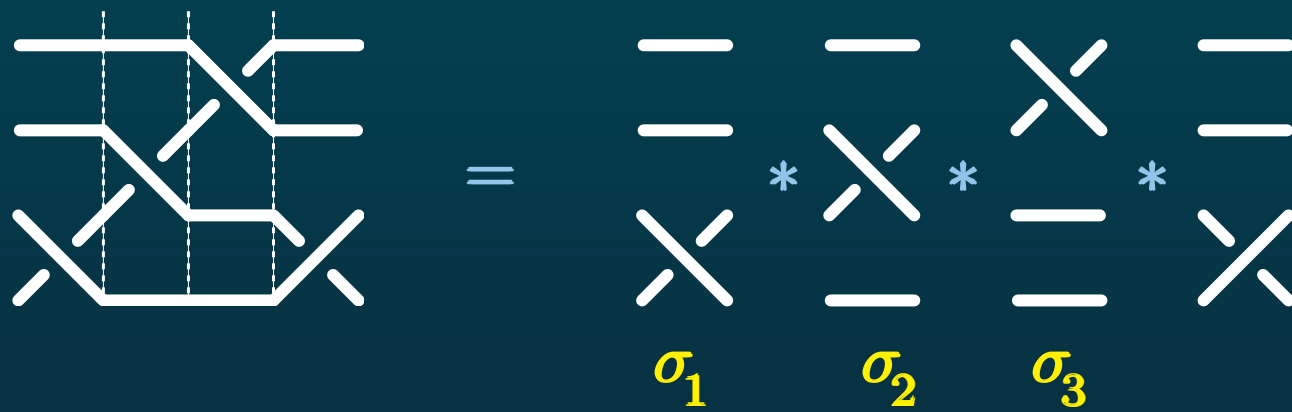








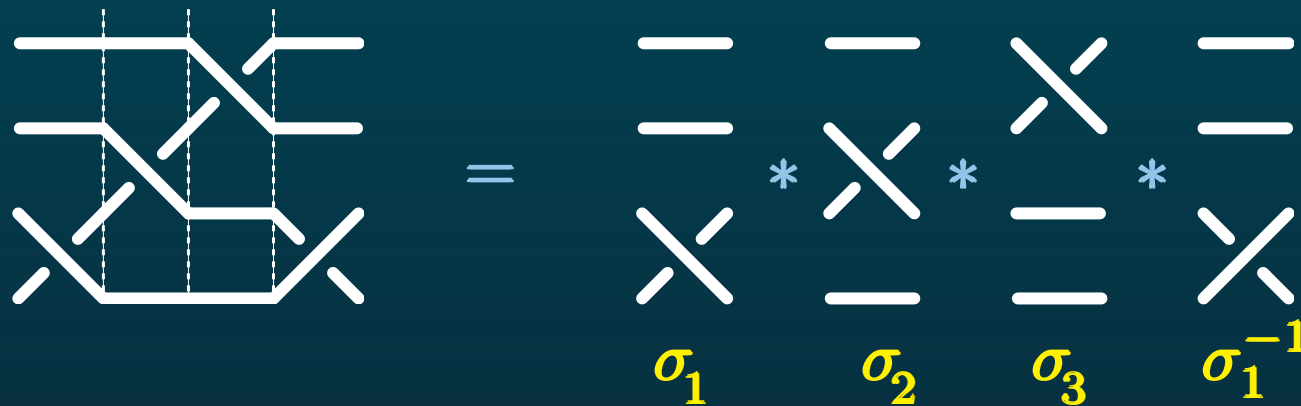




The diagram illustrates the decomposition of a braid into a product of generators. On the left, a braid with four strands is shown, with three vertical dashed lines indicating the positions of the generators. This braid is equal to the product of four generators, each represented by a specific braid diagram:

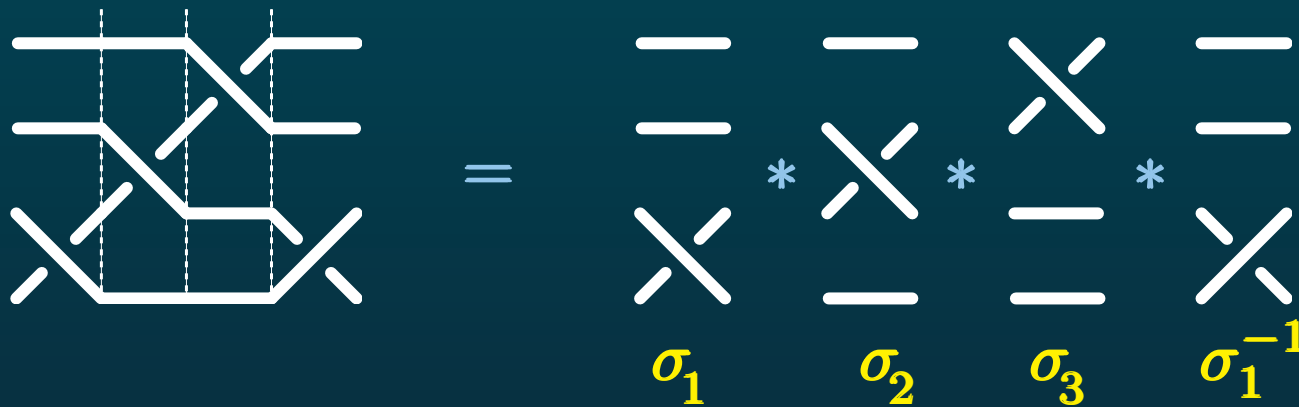
- σ_1 : A braid with two strands crossing, where the strand from the top-left goes to the bottom-right and the strand from the top-right goes to the bottom-left.
- σ_2 : A braid with two strands crossing, where the strand from the top-left goes to the bottom-left and the strand from the top-right goes to the bottom-right.
- σ_3 : A braid with two strands crossing, where the strand from the top-left goes to the bottom-right and the strand from the top-right goes to the bottom-left.
- σ_1^{-1} : A braid with two strands crossing, where the strand from the top-left goes to the bottom-left and the strand from the top-right goes to the bottom-right.

The generators are separated by multiplication symbols ($*$).



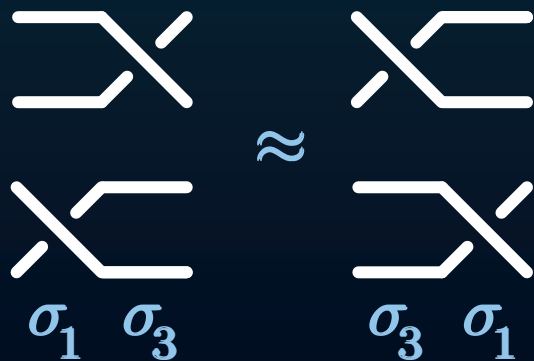
• **Théorème** (Artin): Le groupe de tresses B_n est engendré par $\sigma_1, \dots, \sigma_{n-1}$, soumis aux relations

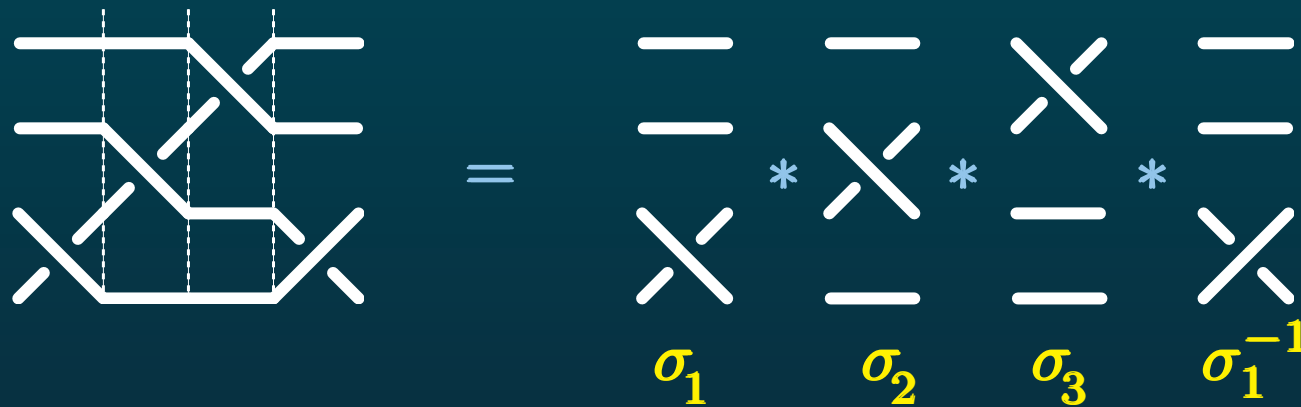
$$\sigma_i \sigma_j = \sigma_j \sigma_i \text{ pour } |i - j| \geq 2, \text{ et } \sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j \text{ pour } |i - j| = 1.$$



• **Théorème** (Artin): Le groupe de tresses B_n est engendré par $\sigma_1, \dots, \sigma_{n-1}$, soumis aux relations

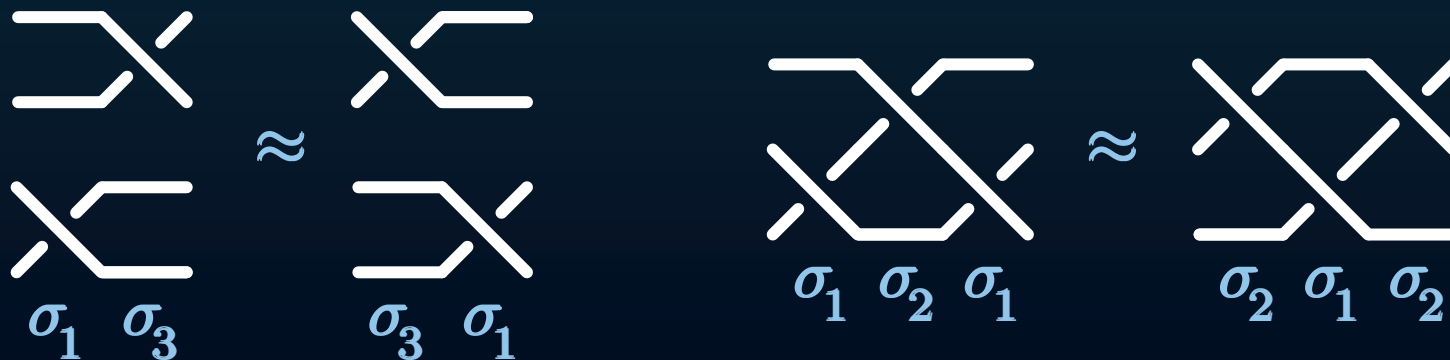
$$\sigma_i \sigma_j = \sigma_j \sigma_i \text{ pour } |i - j| \geq 2, \text{ et } \sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j \text{ pour } |i - j| = 1.$$





• **Théorème** (Artin): Le groupe de tresses B_n est engendré par $\sigma_1, \dots, \sigma_{n-1}$, soumis aux relations

$$\sigma_i \sigma_j = \sigma_j \sigma_i \text{ pour } |i - j| \geq 2, \text{ et } \sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j \text{ pour } |i - j| = 1.$$



- Le **problème d'isotopie** des tresses:
Reconnaître si un diagramme de tresse est isotope au diagramme trivial

- Le **problème d'isotopie** des tresses:

Reconnaître si un diagramme de tresse est isotope au diagramme trivial

⇔ Reconnaître si un mot de tresse w représente **1** dans le groupe des tresses.

- Le **problème d'isotopie** des tresses:

Reconnaître si un diagramme de tresse est isotope au diagramme trivial

⇔ Reconnaître si un mot de tresse w représente **1** dans le groupe des tresses.

↪ Problème #0 pour des applications, par ex. cryptographiques

- Le **problème d'isotopie** des tresses:

Reconnaître si un diagramme de tresse est isotope au diagramme trivial

⇔ Reconnaître si un mot de tresse w représente **1** dans le groupe des tresses.

↪ Problème #0 pour des applications, par ex. cryptographiques

↪ Nombreuses solutions;

- Le **problème d'isotopie** des tresses:

Reconnaître si un diagramme de tresse est isotope au diagramme trivial

⇔ Reconnaître si un mot de tresse w représente **1** dans le groupe des tresses.

- ↪ Problème #0 pour des applications, par ex. cryptographiques
- ↪ Nombreuses solutions;
- ↪ **Une** solution (en fait: la plus efficace à ce jour)

- Le **problème d'isotopie** des tresses:

Reconnaître si un diagramme de tresse est isotope au diagramme trivial

⇔ Reconnaître si un mot de tresse w représente **1** dans le groupe des tresses.

- ↪ Problème #0 pour des applications, par ex. cryptographiques
- ↪ Nombreuses solutions;
- ↪ **Une** solution (en fait: la plus efficace à ce jour)

- Dans un groupe libre, un mot représente **1** ssi il **se réduit** au mot vide en détruisant (itérativement) les xx^{-1} et $x^{-1}x$.

- Le **problème d'isotopie** des tresses:

Reconnaître si un diagramme de tresse est isotope au diagramme trivial

⇔ Reconnaître si un mot de tresse w représente **1** dans le groupe des tresses.

- ↪ Problème #0 pour des applications, par ex. cryptographiques
- ↪ Nombreuses solutions;
- ↪ **Une** solution (en fait: la plus efficace à ce jour)

- Dans un groupe libre, un mot représente **1** ssi il **se réduit** au mot vide en détruisant (itérativement) les xx^{-1} et $x^{-1}x$.

- Faux pour groupe non libre: $\sigma_1 \sigma_2 \sigma_1 \sigma_2^{-1} \sigma_1^{-1} \sigma_2^{-1}$ représente **1** dans B_n .

- Le **problème d'isotopie** des tresses:

Reconnaître si un diagramme de tresse est isotope au diagramme trivial

⇔ Reconnaître si un mot de tresse w représente 1 dans le groupe des tresses.

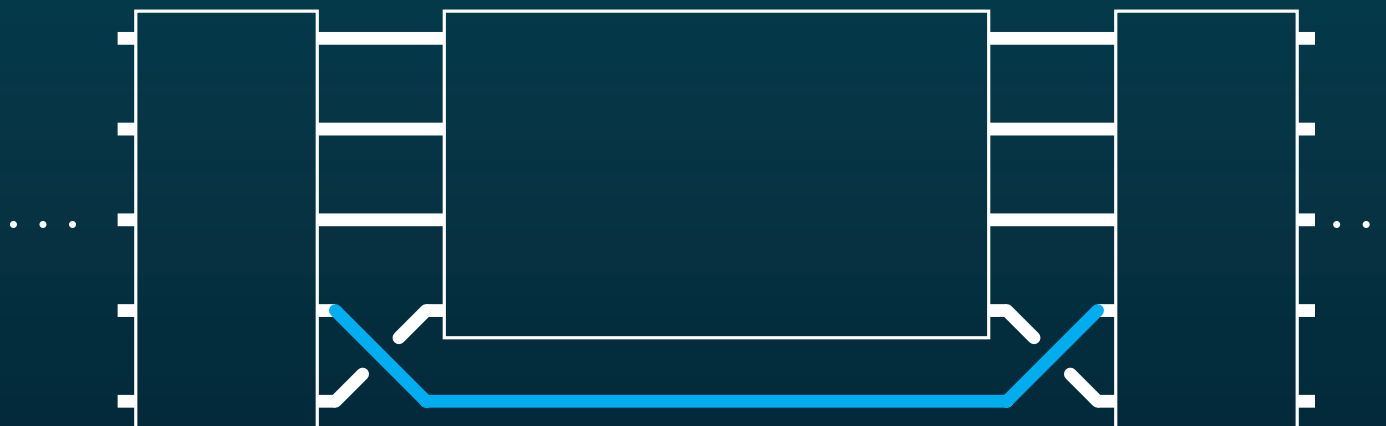
- ↪ Problème #0 pour des applications, par ex. cryptographiques
- ↪ Nombreuses solutions;
- ↪ **Une** solution (en fait: la plus efficace à ce jour)

- Dans un groupe libre, un mot représente 1 ssi il **se réduit** au mot vide en détruisant (itérativement) les xx^{-1} et $x^{-1}x$.

- Faux pour groupe non libre: $\sigma_1 \sigma_2 \sigma_1 \sigma_2^{-1} \sigma_1^{-1} \sigma_2^{-1}$ représente 1 dans B_n .

- Mais $\sigma_1 \sigma_2 \sigma_1 \sigma_2^{-1} \sigma_1^{-1} \sigma_2^{-1}$ contient un motif $\sigma_1 \dots \sigma_1^{-1}$ sans $\sigma_1^{\pm 1}$ au milieu:
une **σ_1 -poignée**.

- Une σ_1 -poignée:



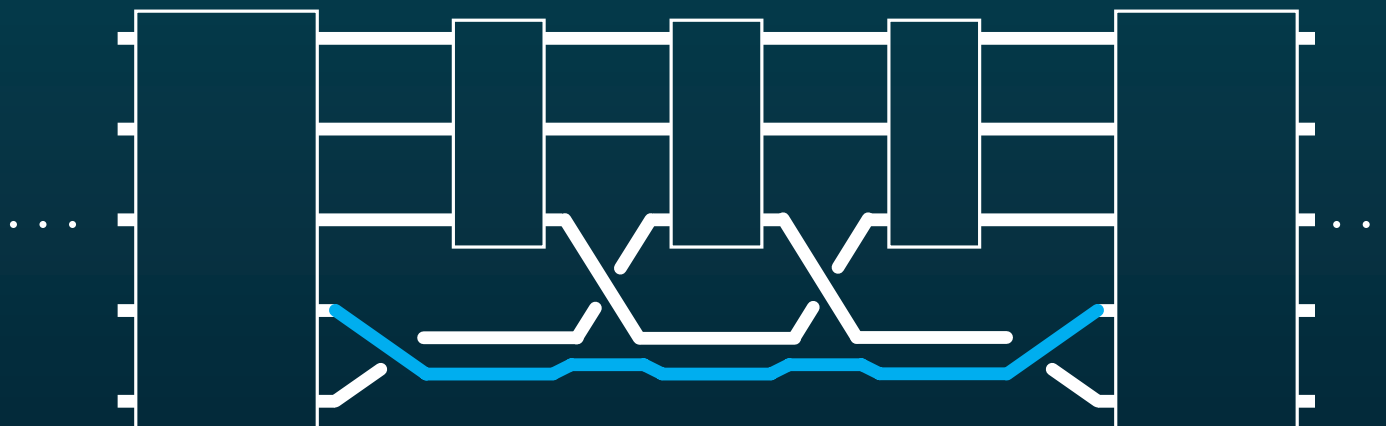
- Une σ_1 -poignée:



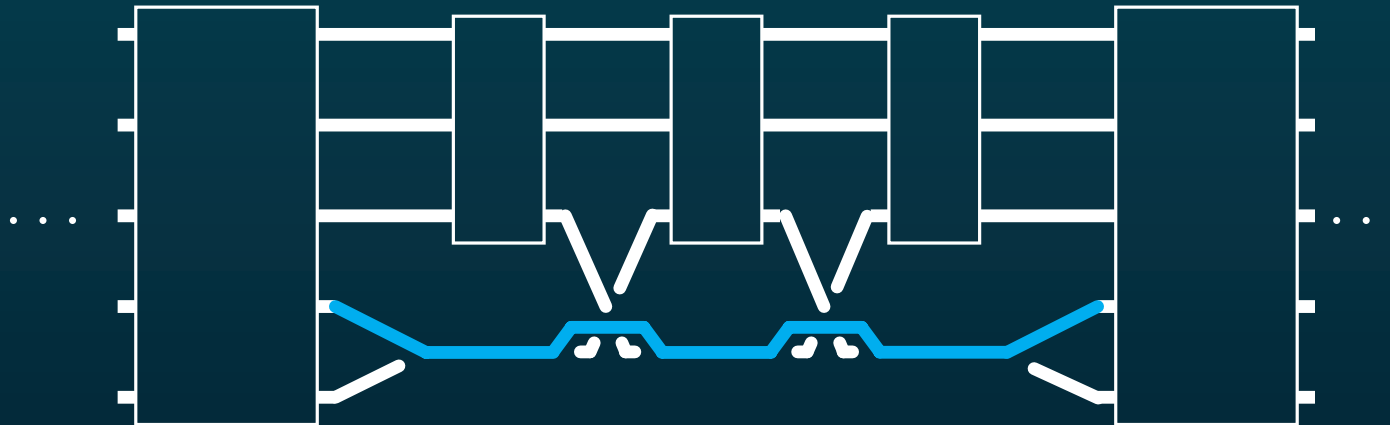
- Une σ_1 -poignée:



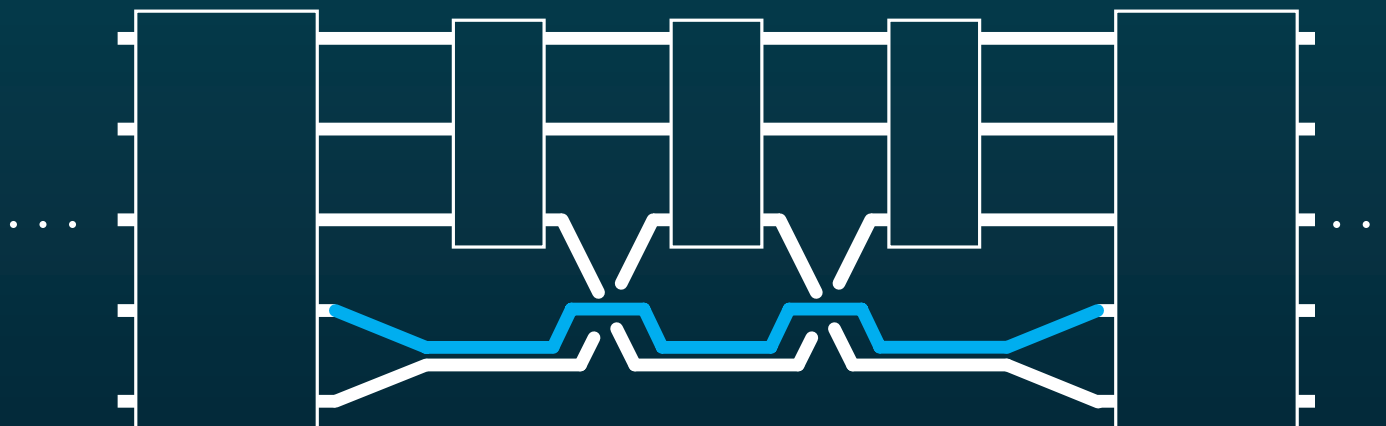
- Une σ_1 -poignée:



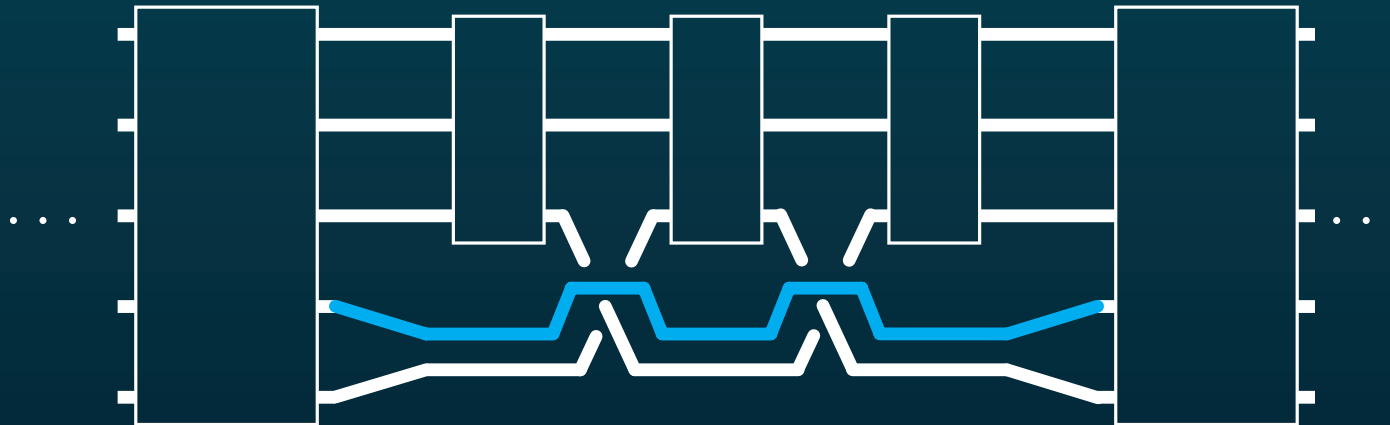
- Une σ_1 -poignée:



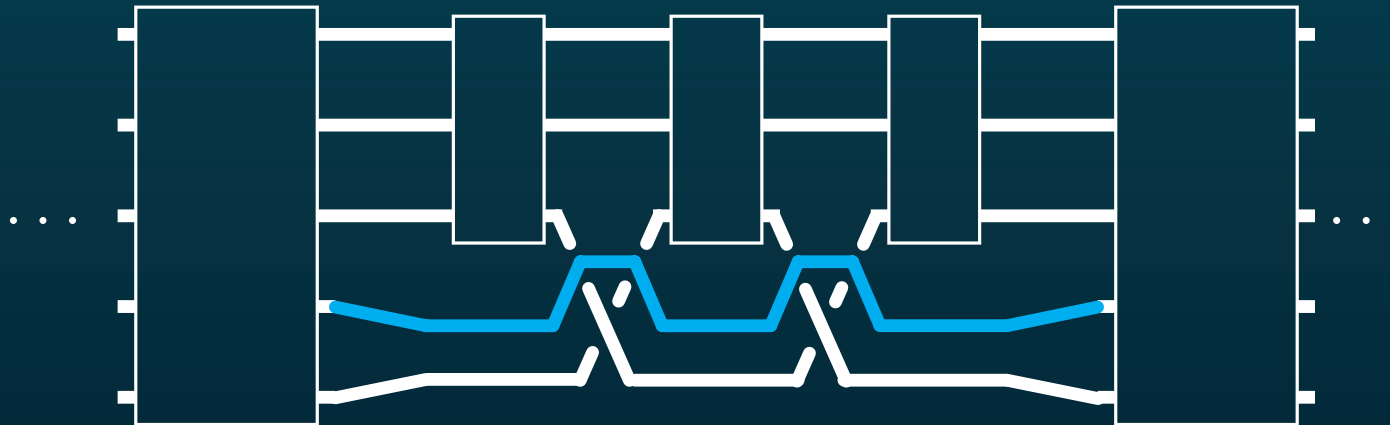
- Une σ_1 -poignée:



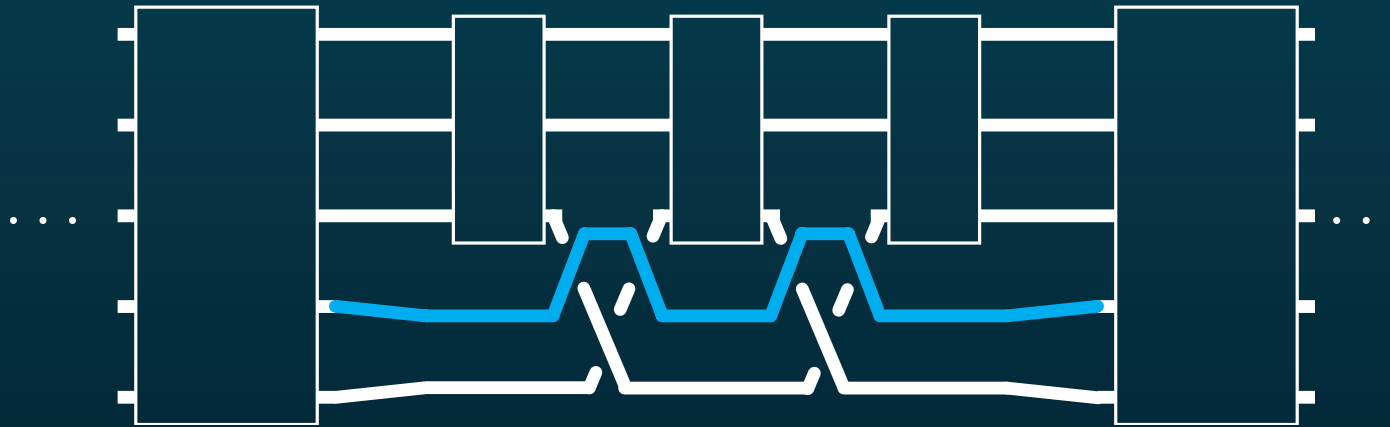
- Une σ_1 -poignée:



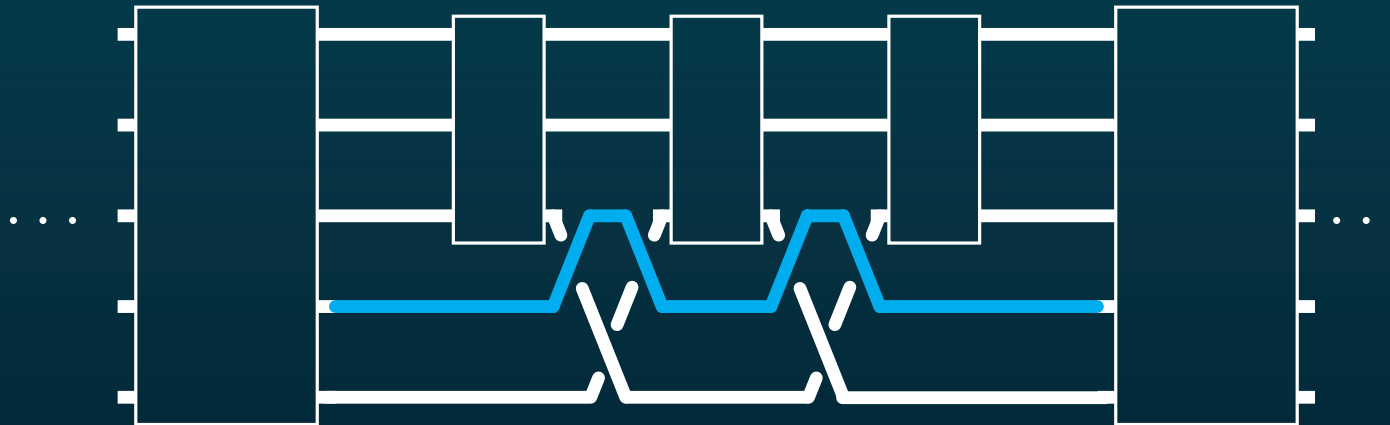
- Une σ_1 -poignée:



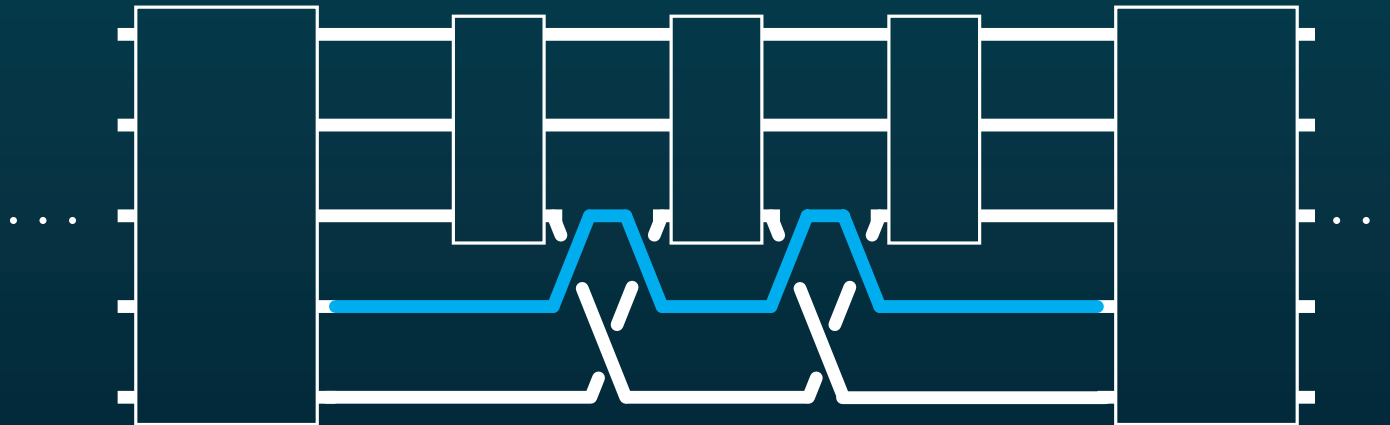
- Une σ_1 -poignée:



- Une σ_1 -poignée:

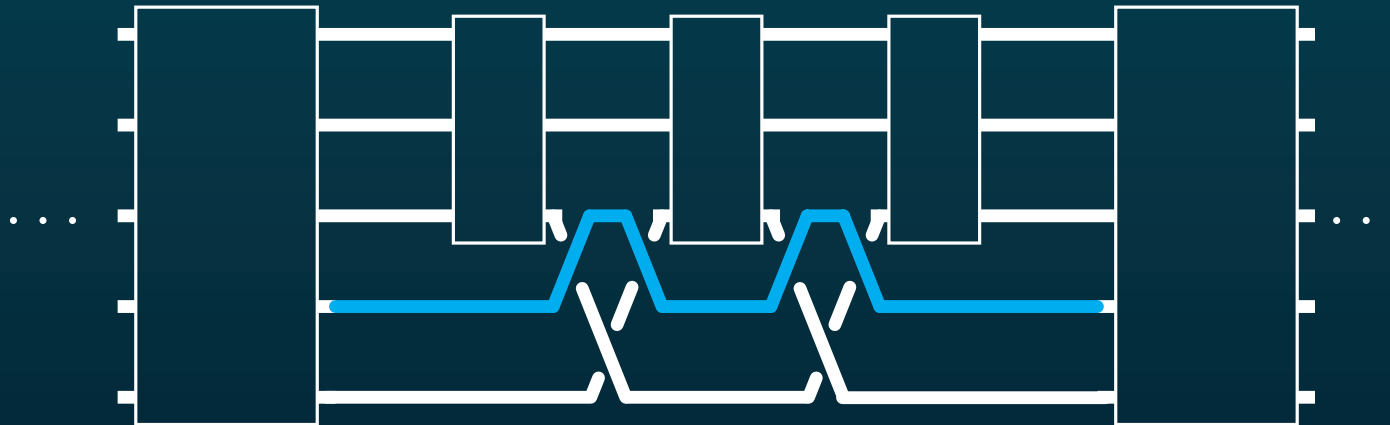


- Une σ_1 -poignée:



- Définition: **Réduire** une σ_1 -poignée = opération ci-dessus
 (= détruire les $\sigma_1^{\pm 1}$, remplacer les $\sigma_2^{\pm 1}$ par $\sigma_2^{-e} \sigma_1^{\pm 1} \sigma_2^e$).

- Une σ_1 -poignée:



- Définition: **Réduire** une σ_1 -poignée = opération ci-dessus
 (= détruire les $\sigma_1^{\pm 1}$, remplacer les $\sigma_2^{\pm 1}$ par $\sigma_2^{-e} \sigma_1^{\pm 1} \sigma_2^e$).

- **Théorème 1:** (D. 1995) La réduction des poignées se termine en un nombre fini d'étapes; un mot de tresse représente **1** ssi il se réduit au mot vide.

- OK, mais d'où vient cette réduction, pourquoi fonctionne-t-elle?

- OK, mais d'où vient cette réduction, pourquoi fonctionne-t-elle?

• **Théorème 2:** (D. 1992) Pour a, b dans B_n , déclarons $a < b$ si $a^{-1}b$ peut être représenté par un mot dans lequel le générateur σ_i d'indice minimal apparaît seulement positivement. Alors $<$ est un ordre total sur B_n .

- OK, mais d'où vient cette réduction, pourquoi fonctionne-t-elle?

• **Théorème 2:** (D. 1992) Pour a, b dans B_n , déclarons $a < b$ si $a^{-1}b$ peut être représenté par un mot dans lequel le générateur σ_i d'indice minimal apparaît seulement positivement. Alors $<$ est un ordre total sur B_n .

σ_i apparaît, mais σ_i^{-1} n'apparaît pas

- OK, mais d'où vient cette réduction, pourquoi fonctionne-t-elle?

• **Théorème 2:** (D. 1992) Pour a, b dans B_n , déclarons $a < b$ si $a^{-1}b$ peut être représenté par un mot dans lequel le générateur σ_i d'indice minimal apparaît seulement positivement. Alors $<$ est un ordre total sur B_n .

σ_i apparaît, mais σ_i^{-1} n'apparaît pas

- Deux ingrédients:

- (A): Un mot de tresse contenant σ_1 et pas σ_1^{-1} ne représente pas 1 ;

- OK, mais d'où vient cette réduction, pourquoi fonctionne-t-elle?

• **Théorème 2:** (D. 1992) Pour a, b dans B_n , déclarons $a < b$ si $a^{-1}b$ peut être représenté par un mot dans lequel le générateur σ_i d'indice minimal apparaît seulement positivement. Alors $<$ est un ordre total sur B_n .

σ_i apparaît, mais σ_i^{-1} n'apparaît pas

- Deux ingrédients:
 - (A): Un mot de tresse contenant σ_1 et pas σ_1^{-1} ne représente pas 1 ;
 - (C): Toute tresse peut être représentée par un mot sans σ_1 ou sans σ_1^{-1} .

- OK, mais d'où vient cette réduction, pourquoi fonctionne-t-elle?

• **Théorème 2:** (D. 1992) Pour a, b dans B_n , déclarons $a < b$ si $a^{-1}b$ peut être représenté par un mot dans lequel le générateur σ_i d'indice minimal apparaît seulement positivement. Alors $<$ est un ordre total sur B_n .

σ_i apparaît, mais σ_i^{-1} n'apparaît pas

- Deux ingrédients:
 - (A): Un mot de tresse contenant σ_1 et pas σ_1^{-1} ne représente pas 1 ;
 - (C): Toute tresse peut être représentée par un mot sans σ_1 ou sans σ_1^{-1} .
- Théo. 1 **vient de** Théo. 2:

- OK, mais d'où vient cette réduction, pourquoi fonctionne-t-elle?

• **Théorème 2:** (D. 1992) Pour a, b dans B_n , déclarons $a < b$ si $a^{-1}b$ peut être représenté par un mot dans lequel le générateur σ_i d'indice minimal apparaît seulement positivement. Alors $<$ est un ordre total sur B_n .

σ_i apparaît, mais σ_i^{-1} n'apparaît pas

- Deux ingrédients:
 - (A): Un mot de tresse contenant σ_1 et pas σ_1^{-1} ne représente pas 1 ;
 - (C): Toute tresse peut être représentée par un mot sans σ_1 ou sans σ_1^{-1} .
- Théo. 1 **vient de** Théo. 2: (C) donne l'idée et (A) la convergence : quelque chose décroît décroît pour $<$ quand une réduction est effectuée.

- OK, mais d'où vient cette réduction, pourquoi fonctionne-t-elle?

• **Théorème 2:** (D. 1992) Pour a, b dans B_n , déclarons $a < b$ si $a^{-1}b$ peut être représenté par un mot dans lequel le générateur σ_i d'indice minimal apparaît seulement positivement. Alors $<$ est un ordre total sur B_n .

↙ σ_i apparaît, mais σ_i^{-1} n'apparaît pas


- Deux ingrédients:
 - (A): Un mot de tresse contenant σ_1 et pas σ_1^{-1} ne représente pas 1 ;
 - (C): Toute tresse peut être représentée par un mot sans σ_1 ou sans σ_1^{-1} .
- Théo. 1 **vient de** Théo. 2: (C) donne l'idée et (A) la convergence : quelque chose décroît décroît pour $<$ quand une réduction est effectuée.

↔ Question: **D'où** vient l'ordre des tresses?


... de l'étude de l'**auto-distributivité**.

- **Colorier** les tresses :

... de l'étude de l'**auto-distributivité**.

- **Colorier** les tresses : pour $(S, *)$ fixé, attribuer des couleurs de S aux brins en entrée, propager suivant , et comparer les sorties.


... de l'étude de l'**auto-distributivité**.

- **Colorier** les tresses : pour $(S, *)$ fixé, attribuer des couleurs de S aux brins en entrée, propager suivant  et comparer les sorties.

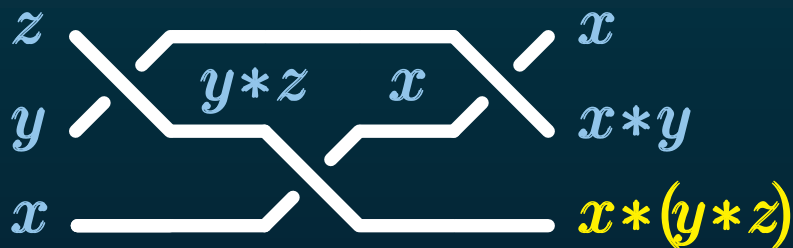
- Pour action de B_n sur S^n , compatibilité requise avec les relations de tresse :




... de l'étude de l'**auto-distributivité**.

- **Colorier** les tresses : pour $(S, *)$ fixé, attribuer des couleurs de S aux brins en entrée, propager suivant , et comparer les sorties.

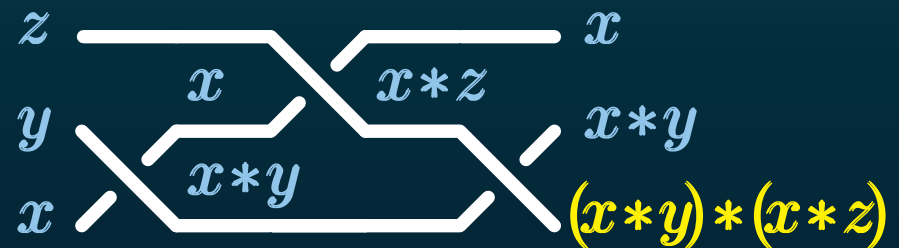
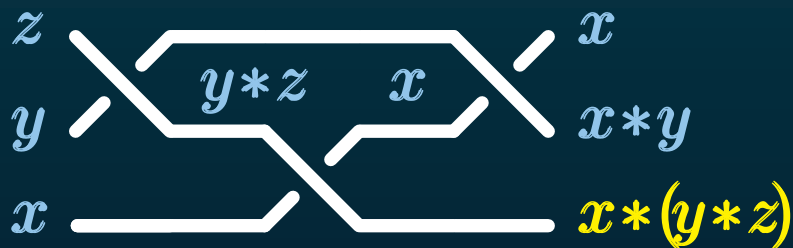
- Pour action de B_n sur S^n , compatibilité requise avec les relations de tresse :




... de l'étude de l'**auto-distributivité**.

- **Colorier** les tresses : pour $(S, *)$ fixé, attribuer des couleurs de S aux brins en entrée, propager suivant , et comparer les sorties.

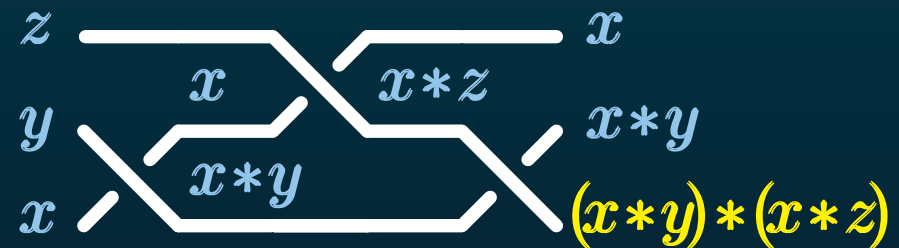
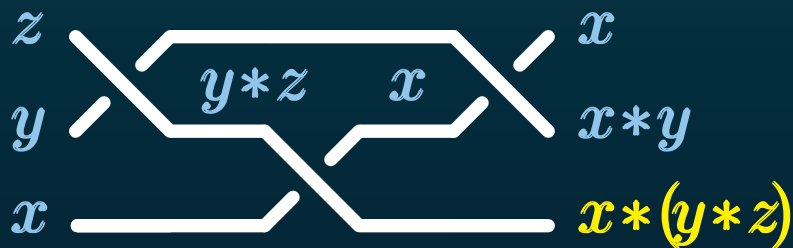
- Pour action de B_n sur S^n , compatibilité requise avec les relations de tresse :



... de l'étude de l'**auto-distributivité**.

- **Colorier** les tresses : pour $(S, *)$ fixé, attribuer des couleurs de S aux brins en entrée, propager suivant , et comparer les sorties.

- Pour action de B_n sur S^n , compatibilité requise avec les relations de tresse:



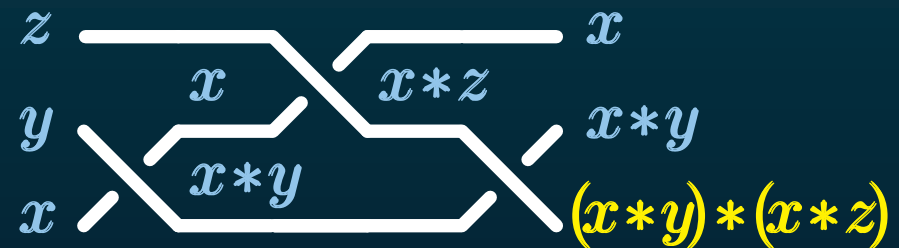
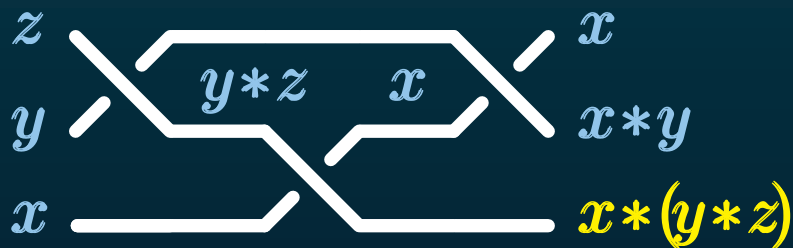
↪ $(S, *)$ doit être un **LD-système**, i.e., satisfaire la loi d'**autodistributivité**:

$$x * (y * z) = (x * y) * (x * z). \quad (\text{LD})$$

... de l'étude de l'auto-distributivité.

- Colorier les tresses : pour $(S, *)$ fixé, attribuer des couleurs de S aux brins en entrée, propager suivant $\begin{matrix} y & & x \\ & \diagdown & / \\ x & & x * y \end{matrix}$, et comparer les sorties.

- Pour action de B_n sur S^n , compatibilité requise avec les relations de tresse:



↔ $(S, *)$ doit être un LD-système, i.e., satisfaire la loi d'autodistributivité:

$$x * (y * z) = (x * y) * (x * z). \quad (\text{LD})$$

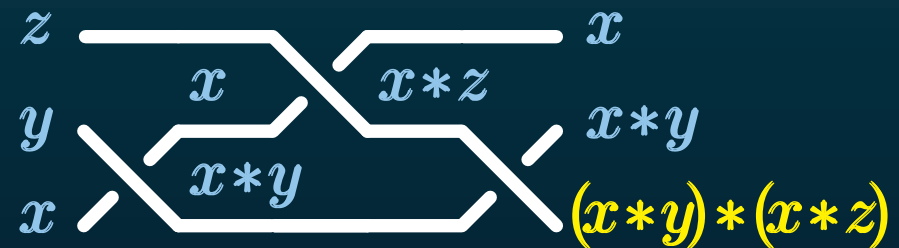
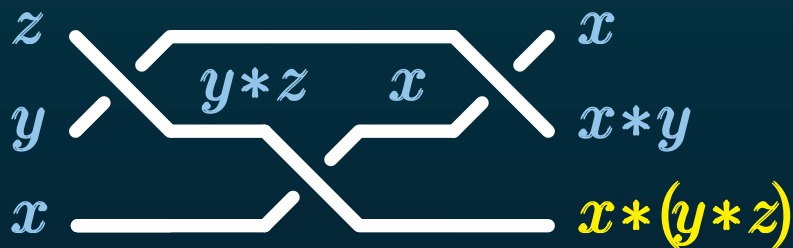
- Exemples standards :

- $x * y = y$, mène à $B_n \twoheadrightarrow S_n$

... de l'étude de l'auto-distributivité.

- Colorier les tresses : pour $(S, *)$ fixé, attribuer des couleurs de S aux brins en entrée, propager suivant $\begin{matrix} y & & x \\ & \diagdown & / \\ x & & x * y \end{matrix}$, et comparer les sorties.

- Pour action de B_n sur S^n , compatibilité requise avec les relations de tresse:




↔ $(S, *)$ doit être un LD-système, i.e., satisfaire la loi d'autodistributivité:

$$x * (y * z) = (x * y) * (x * z). \quad (\text{LD})$$

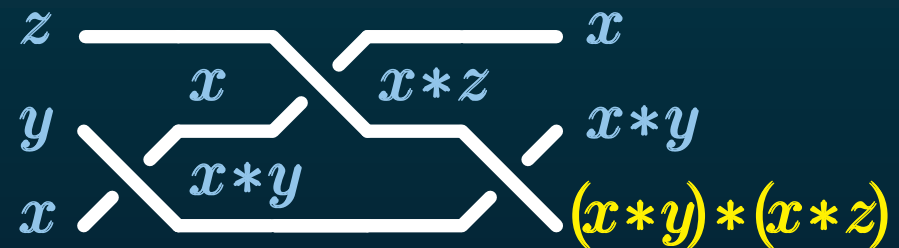
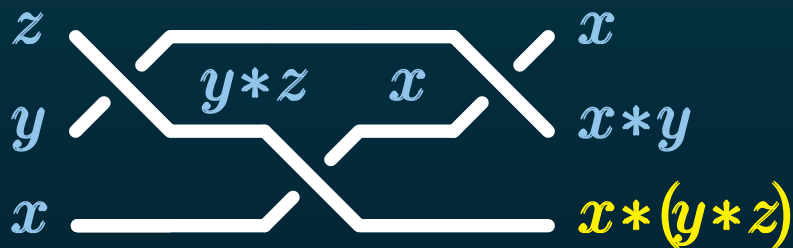
- Exemples standards :

- $x * y = y$, mène à $B_n \twoheadrightarrow S_n$
- $x * y = xyx^{-1}$, mène à $B_n \hookrightarrow \text{Aut}(F_n)$ (Artin)

... de l'étude de l'auto-distributivité.

- **Colorier** les tresses : pour $(S, *)$ fixé, attribuer des couleurs de S aux brins en entrée, propager suivant , et comparer les sorties.

- Pour action de B_n sur S^n , compatibilité requise avec les relations de tresse :



↔ $(S, *)$ doit être un **LD-système**, i.e., satisfaire la loi d'**autodistributivité** :

$$x * (y * z) = (x * y) * (x * z). \quad (\text{LD})$$

- Exemples standards :

- $x * y = y$, mène à $B_n \twoheadrightarrow S_n$
- $x * y = xyx^{-1}$, mène à $B_n \hookrightarrow \text{Aut}(F_n)$ (Artin)
- $x * y = (1 - t)x + ty$, mène à $B_n \rightarrow \text{GL}_n(\mathbb{Z}[t, t^{-1}])$ (Bureau)

- Dans les exemples précédents, on a toujours $x * x = x$; autres exemples?

- Dans les exemples précédents, on a toujours $x * x = x$; autres exemples?
- Déclarons un LD-système $(S, *)$ **ordonnable** s'il existe un ordre total sur S satisfaisant $x < x * y$ pour tous x, y . (\rightsquigarrow très différent : $x < x * x \neq x$)

- Dans les exemples précédents, on a toujours $x * x = x$; autres exemples?
- Déclarons un LD-système $(S, *)$ **ordonnable** s'il existe un ordre total sur S satisfaisant $x < x * y$ pour tous x, y . (\rightsquigarrow très différent : $x < x * x \neq x$)

- **Théorème 3:** (D. 1991) Il existe des LD-systèmes ordonnables.
(à savoir les LD-systèmes libres)

- Dans les exemples précédents, on a toujours $x * x = x$; autres exemples?
- Déclarons un LD-système $(S, *)$ **ordonnable** s'il existe un ordre total sur S satisfaisant $x < x * y$ pour tous x, y . (\rightsquigarrow très différent : $x < x * x \neq x$)

- **Théorème 3:** (D. 1991) Il existe des LD-systèmes ordonnables.
(à savoir les LD-systèmes libres)

- Théo. 2 **vient de** Théo. 3 : Colorier avec un LD-système ordonnable.

- Dans les exemples précédents, on a toujours $x * x = x$; autres exemples?
- Déclarons un LD-système $(S, *)$ **ordonnable** s'il existe un ordre total sur S satisfaisant $x < x * y$ pour tous x, y . (\rightsquigarrow très différent : $x < x * x \neq x$)

- **Théorème 3:** (D. 1991) Il existe des LD-systèmes ordonnables.
(à savoir les LD-systèmes libres)

- Théo. 2 **vient de** Théo. 3 : Colorier avec un LD-système ordonnable.

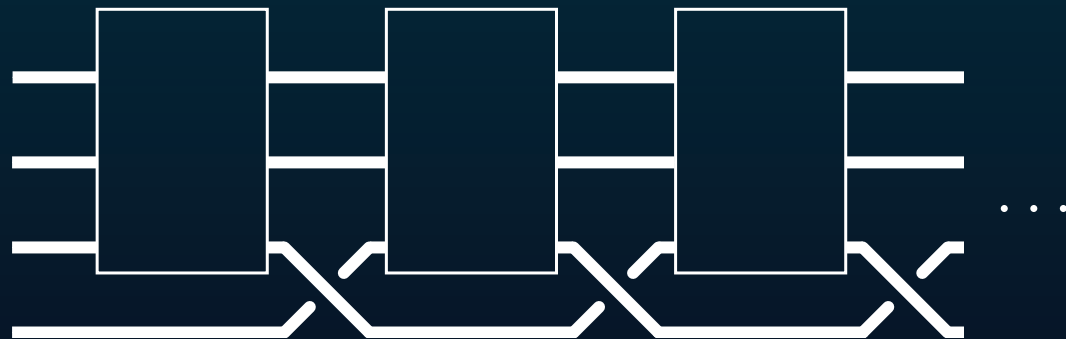
Propriété (A) : un mot
contenant σ_1 et pas σ_1^{-1}
ne représente pas 1

- Dans les exemples précédents, on a toujours $x * x = x$; autres exemples?
- Déclarons un LD-système $(S, *)$ **ordonnable** s'il existe un ordre total sur S satisfaisant $x < x * y$ pour tous x, y . (\rightsquigarrow très différent : $x < x * x \neq x$)

- **Théorème 3:** (D. 1991) Il existe des LD-systèmes ordonnables.
(à savoir les LD-systèmes libres)

- Théo. 2 **vient de** Théo. 3 : Colorier avec un LD-système ordonnable.

Propriété (A) : un mot contenant σ_1 et pas σ_1^{-1} ne représente pas 1

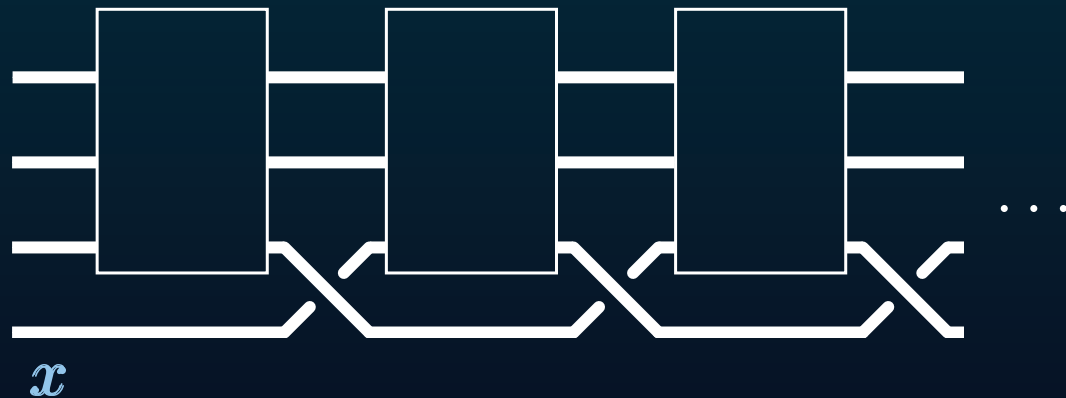


- Dans les exemples précédents, on a toujours $x * x = x$; autres exemples?
- Déclarons un LD-système $(S, *)$ **ordonnable** s'il existe un ordre total sur S satisfaisant $x < x * y$ pour tous x, y . (\rightsquigarrow très différent : $x < x * x \neq x$)

- **Théorème 3:** (D. 1991) Il existe des LD-systèmes ordonnables.
(à savoir les LD-systèmes libres)

- Théo. 2 **vient de** Théo. 3 : Colorier avec un LD-système ordonnable.

Propriété (A) : un mot contenant σ_1 et pas σ_1^{-1} ne représente pas 1

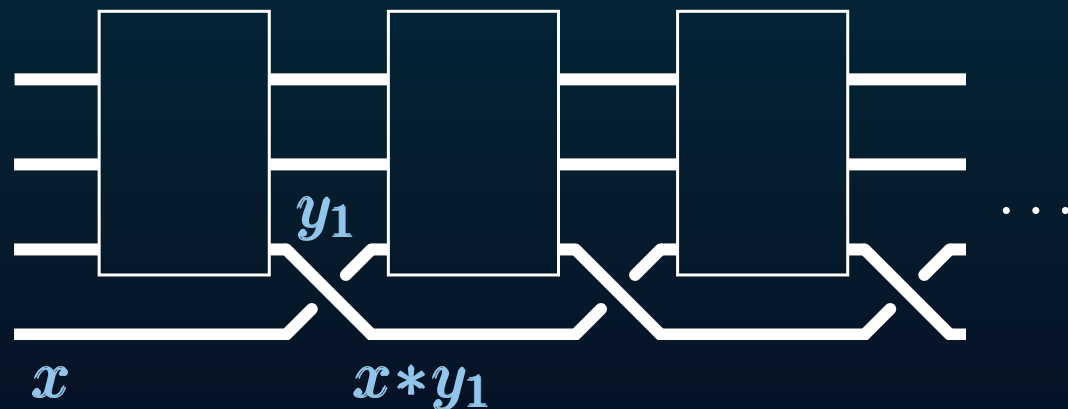


- Dans les exemples précédents, on a toujours $x * x = x$; autres exemples?
- Déclarons un LD-système $(S, *)$ **ordonnable** s'il existe un ordre total sur S satisfaisant $x < x * y$ pour tous x, y . (\rightsquigarrow très différent : $x < x * x \neq x$)

- **Théorème 3:** (D. 1991) Il existe des LD-systèmes ordonnables.
(à savoir les LD-systèmes libres)

- Théo. 2 **vient de** Théo. 3 : Colorier avec un LD-système ordonnable.

Propriété (A) : un mot contenant σ_1 et pas σ_1^{-1} ne représente pas 1

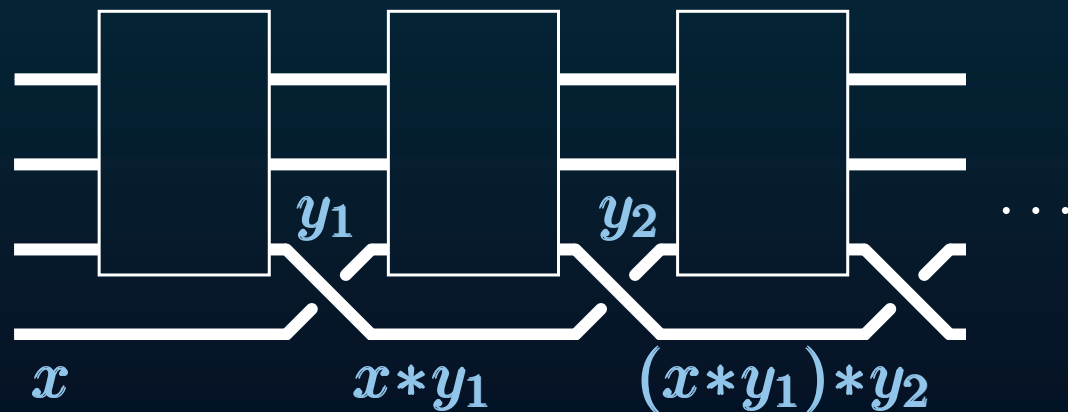


- Dans les exemples précédents, on a toujours $x * x = x$; autres exemples?
- Déclarons un LD-système $(S, *)$ **ordonnable** s'il existe un ordre total sur S satisfaisant $x < x * y$ pour tous x, y . (\rightsquigarrow très différent : $x < x * x \neq x$)

- **Théorème 3:** (D. 1991) Il existe des LD-systèmes ordonnables.
(à savoir les LD-systèmes libres)

- Théo. 2 **vient de** Théo. 3 : Colorier avec un LD-système ordonnable.

Propriété (A) : un mot contenant σ_1 et pas σ_1^{-1} ne représente pas 1

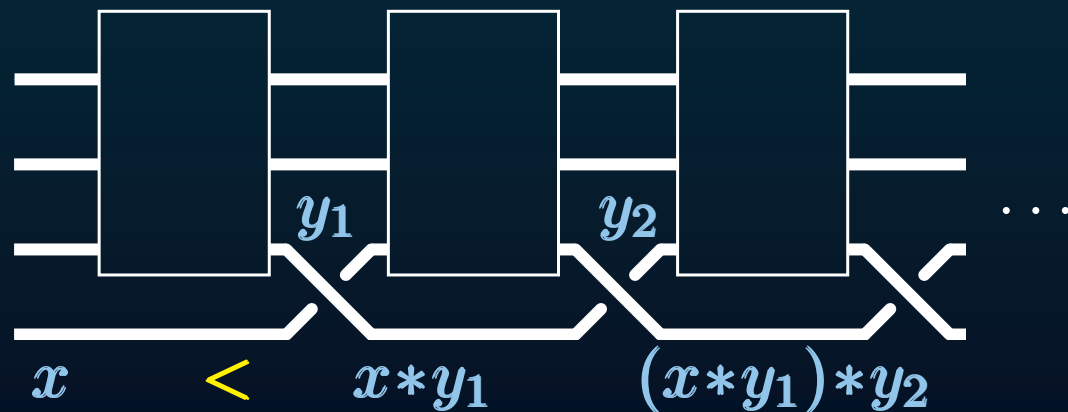


- Dans les exemples précédents, on a toujours $x * x = x$; autres exemples?
- Déclarons un LD-système $(S, *)$ **ordonnable** s'il existe un ordre total sur S satisfaisant $x < x * y$ pour tous x, y . (\rightsquigarrow très différent : $x < x * x \neq x$)

- **Théorème 3:** (D. 1991) Il existe des LD-systèmes ordonnables.
(à savoir les LD-systèmes libres)

- Théo. 2 **vient de** Théo. 3 : Colorier avec un LD-système ordonnable.

Propriété (A) : un mot contenant σ_1 et pas σ_1^{-1} ne représente pas 1

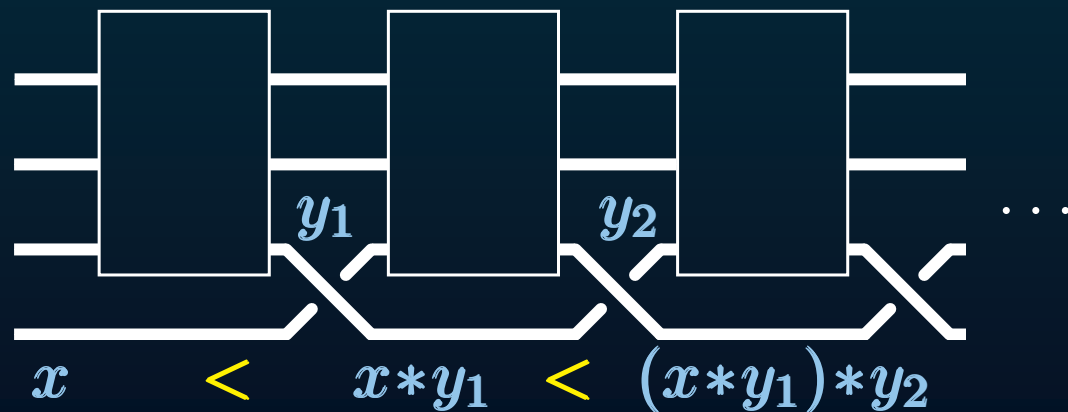


- Dans les exemples précédents, on a toujours $x * x = x$; autres exemples?
- Déclarons un LD-système $(S, *)$ **ordonnable** s'il existe un ordre total sur S satisfaisant $x < x * y$ pour tous x, y . (\rightsquigarrow très différent : $x < x * x \neq x$)

• **Théorème 3:** (D. 1991) Il existe des LD-systèmes ordonnables.
(à savoir les LD-systèmes libres)

- Théo. 2 **vient de** Théo. 3 : Colorier avec un LD-système ordonnable.

Propriété (A) : un mot contenant σ_1 et pas σ_1^{-1} ne représente pas 1

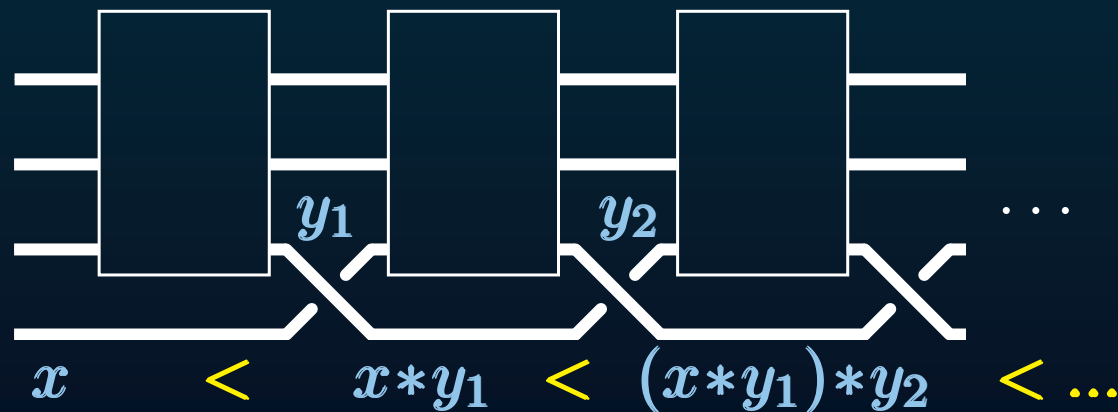


- Dans les exemples précédents, on a toujours $x * x = x$; autres exemples?
- Déclarons un LD-système $(S, *)$ **ordonnable** s'il existe un ordre total sur S satisfaisant $x < x * y$ pour tous x, y . (\rightsquigarrow très différent : $x < x * x \neq x$)

- **Théorème 3:** (D. 1991) Il existe des LD-systèmes ordonnables.
(à savoir les LD-systèmes libres)

- Théo. 2 **vient de** Théo. 3 : Colorier avec un LD-système ordonnable.

Propriété (A) : un mot contenant σ_1 et pas σ_1^{-1} ne représente pas 1

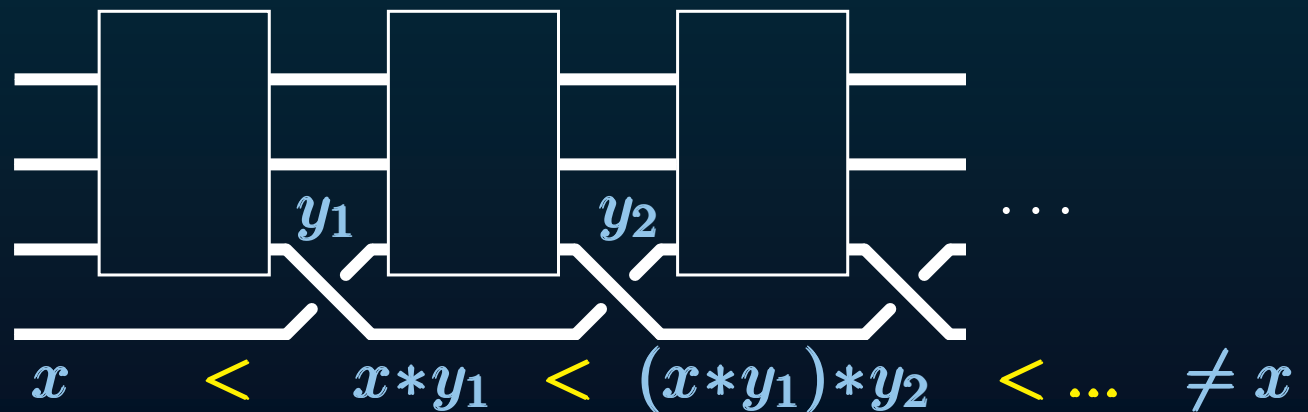


- Dans les exemples précédents, on a toujours $x * x = x$; autres exemples?
- Déclarons un LD-système $(S, *)$ **ordonnable** s'il existe un ordre total sur S satisfaisant $x < x * y$ pour tous x, y . (\rightsquigarrow très différent : $x < x * x \neq x$)

• **Théorème 3:** (D. 1991) Il existe des LD-systèmes ordonnables.
(à savoir les LD-systèmes libres)

- Théo. 2 **vient de** Théo. 3 : Colorier avec un LD-système ordonnable.

Propriété (A) : un mot contenant σ_1 et pas σ_1^{-1} ne représente pas 1

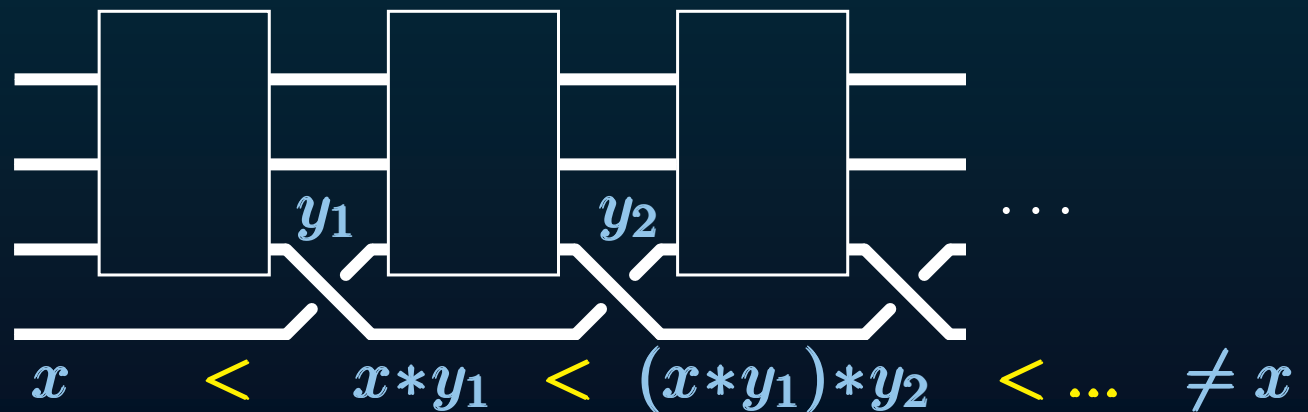


- Dans les exemples précédents, on a toujours $x * x = x$; autres exemples?
- Déclarons un LD-système $(S, *)$ **ordonnable** s'il existe un ordre total sur S satisfaisant $x < x * y$ pour tous x, y . (\rightsquigarrow très différent : $x < x * x \neq x$)

- **Théorème 3:** (D. 1991) Il existe des LD-systèmes ordonnables.
(à savoir les LD-systèmes libres)

- Théo. 2 **vient de** Théo. 3 : Colorier avec un LD-système ordonnable.

Propriété (A) : un mot contenant σ_1 et pas σ_1^{-1} ne représente pas 1



\rightsquigarrow Question: OK, mais pourquoi chercher des LD-systèmes ordonnables?

... parce que la **théorie des ensembles** le dit

... parce que la **théorie des ensembles** le dit

- La théorie des ensembles étudie l'infini, et introduit en particulier diverses notions d'ensembles "hyper-infinis" (**grands cardinaux**):

... parce que la **théorie des ensembles** le dit

- La théorie des ensembles étudie l'infini, et introduit en particulier diverses notions d'ensembles "hyper-infinis" (**grands cardinaux**):
 - ↔ typiq^t: renforcer " X est infini ssi $\exists j : X \rightarrow X$ injective non surjective" en " X est hyper-infini (**auto-similaire**) ssi $\exists j \dots$ et, de plus, j préserve tout ce qui est définissable à partir de \in (**plongement élémentaire**)".

... parce que la **théorie des ensembles** le dit

- La théorie des ensembles étudie l'infini, et introduit en particulier diverses notions d'ensembles "hyper-infinis" (**grands cardinaux**):
 \rightsquigarrow typiq^t: renforcer " X est infini ssi $\exists j : X \rightarrow X$ injective non surjective"
 en " X est hyper-infini (**auto-similaire**) ssi $\exists j \dots$ et, de plus, j préserve tout ce qui est définissable à partir de \in (**plongement élémentaire**)".
- Dans un contexte ad hoc ("**rangs auto-similaires**") on peut appliquer un plongement élémentaire i à un autre j , et obtenir un nouveau p.é. $i(j)$; comme "être l'image de" est définissable en \in , on a $i(j(k)) = i(j)(i(k))$.

... parce que la **théorie des ensembles** le dit

- La théorie des ensembles étudie l'infini, et introduit en particulier diverses notions d'ensembles "hyper-infinis" (**grands cardinaux**):

↔ typiq^t: renforcer " X est infini ssi $\exists j : X \rightarrow X$ injective non surjective" en " X est hyper-infini (**auto-similaire**) ssi $\exists j \dots$ et, de plus, j préserve tout ce qui est définissable à partir de \in (**plongement élémentaire**)".

- Dans un contexte ad hoc ("**rangs auto-similaires**") on peut appliquer un plongement élémentaire i à un autre j , et obtenir un nouveau p.é. $i(j)$; comme "être l'image de" est définissable en \in , on a $i(j(k)) = i(j)(i(k))$.

↔ Pour chaque p.é. j d'un rang auto-similaire, un **LD-système** $I(j)$:

$$I(j) = \{j, j(j), j(j)(j), \dots\};$$

... parce que la **théorie des ensembles** le dit

- La théorie des ensembles étudie l'infini, et introduit en particulier diverses notions d'ensembles "hyper-infinis" (**grands cardinaux**):

↔ typiq^t: renforcer " X est infini ssi $\exists j : X \rightarrow X$ injective non surjective" en " X est hyper-infini (**auto-similaire**) ssi $\exists j \dots$ et, de plus, j préserve tout ce qui est définissable à partir de \in (**plongement élémentaire**)".

- Dans un contexte ad hoc ("**rangs auto-similaires**") on peut appliquer un plongement élémentaire i à un autre j , et obtenir un nouveau p.é. $i(j)$; comme "être l'image de" est définissable en \in , on a $i(j(k)) = i(j)(i(k))$.

↔ Pour chaque p.é. j d'un rang auto-similaire, un **LD-système** $I(j)$:

$$I(j) = \{j, j(j), j(j)(j), \dots\};$$

- Proposition: (D. 1986) Si j est un p.é. d'un rang autosimilaire, la LD-structure de $I(j)$ entraîne la Π_1^1 -détermination. ↔ " $I(j)$ est non trivial."

UNE SITUATION ETRANGE

- **Théorème:** (D. 1989) S'il existe au moins un LD-système ordonnable, alors le problème de mot de LD est résoluble algorithmiquement.
↙ décider si deux termes sont équivalents modulo LD

- **Théorème:** (D. 1989) S'il existe au moins un LD-système ordonnable, alors le problème de mot de LD est résoluble algorithmiquement.
↙ décider si deux termes sont équivalents modulo LD
- **Théorème:** (Laver, 1989) Si j est un p.é. d'un rang autosimilaire, alors le LD-système $I(j)$ est ordonnable.

- **Théorème:** (D. 1989) S'il existe au moins un LD-système ordonnable, alors le problème de mot de LD est résoluble algorithmiquement.
↙ décider si deux termes sont équivalents modulo LD
- **Théorème:** (Laver, 1989) Si j est un p.é. d'un rang autosimilaire, alors le LD-système $I(j)$ est ordonnable.

- **Corollaire:** S'il existe un rang autosimilaire, alors le problème de mot de LD est résoluble algorithmiquement.

- **Théorème:** (D. 1989) S'il existe au moins un LD-système ordonnable, alors le problème de mot de LD est résoluble algorithmiquement.
↙ décider si deux termes sont équivalents modulo LD
- **Théorème:** (Laver, 1989) Si j est un p.é. d'un rang autosimilaire, alors le LD-système $I(j)$ est ordonnable.

● **Corollaire:** S'il existe un rang autosimilaire, alors le problème de mot de LD est résoluble algorithmiquement.

- **Mais** l'existence d'un rang autosimilaire est un axiome indémontrable

- **Théorème:** (D. 1989) S'il existe au moins un LD-système ordonnable, alors le problème de mot de LD est résoluble algorithmiquement.
↙ décider si deux termes sont équivalents modulo LD
- **Théorème:** (Laver, 1989) Si j est un p.é. d'un rang autosimilaire, alors le LD-système $I(j)$ est ordonnable.

● **Corollaire:** S'il existe un rang autosimilaire, alors le problème de mot de LD est résoluble algorithmiquement.

- **Mais** l'existence d'un rang autosimilaire est un axiome indémontrable
↪ Le corollaire n'est **pas** une solution au problème de mot de LD

- **Théorème:** (D. 1989) S'il existe au moins un LD-système ordonnable, alors le problème de mot de LD est résoluble algorithmiquement.
↙ décider si deux termes sont équivalents modulo LD
- **Théorème:** (Laver, 1989) Si j est un p.é. d'un rang autosimilaire, alors le LD-système $I(j)$ est ordonnable.

● **Corollaire:** S'il existe un rang autosimilaire, alors le problème de mot de LD est résoluble algorithmiquement.

- **Mais** l'existence d'un rang autosimilaire est un axiome indémontrable
 - ↗ Le corollaire n'est **pas** une solution au problème de mot de LD
 - ↗ Construire un **vrai** exemple de LD-système ordonnable

- **Théorème:** (D. 1989) S'il existe au moins un LD-système ordonnable, alors le problème de mot de LD est résoluble algorithmiquement.
↙ décider si deux termes sont équivalents modulo LD
- **Théorème:** (Laver, 1989) Si j est un p.é. d'un rang autosimilaire, alors le LD-système $I(j)$ est ordonnable.

● **Corollaire:** S'il existe un rang autosimilaire, alors le problème de mot de LD est résoluble algorithmiquement.

- **Mais** l'existence d'un rang autosimilaire est un axiome indémontrable
 - ↗ Le corollaire n'est **pas** une solution au problème de mot de LD
 - ↗ Construire un **vrai** exemple de LD-système ordonnable
 - ↗ Th. 3 (“ \exists LD-système ordonnable”) via **groupe de géométrie** de LD

- **Théorème:** (D. 1989) S'il existe au moins un LD-système ordonnable, alors le problème de mot de LD est résoluble algorithmiquement.
↙ décider si deux termes sont équivalents modulo LD
- **Théorème:** (Laver, 1989) Si j est un p.é. d'un rang autosimilaire, alors le LD-système $I(j)$ est ordonnable.

● **Corollaire:** S'il existe un rang autosimilaire, alors le problème de mot de LD est résoluble algorithmiquement.

- **Mais** l'existence d'un rang autosimilaire est un axiome indémontrable
 - ↗ Le corollaire n'est **pas** une solution au problème de mot de LD
 - ↗ Construire un **vrai** exemple de LD-système ordonnable
 - ↗ Th. 3 ("∃ LD-système ordonnable") via **groupe de géométrie** de LD
 - ↗ Comme G_{LD} est une extension de B_∞ , applications aux tresses.

DES APPLICATIONS DE LA THEORIE DES ENSEMBLES?

↪ un chemin **continu** des ensembles aux groupes de tresses.

DES APPLICATIONS DE LA THEORIE DES ENSEMBLES?

↪ un chemin **continu** des ensembles aux groupes de tresses.

- Question : Les résultats sur les tresses sont-ils des **applications** de la théorie des ensembles?

DES APPLICATIONS DE LA THEORIE DES ENSEMBLES?

↪ un chemin **continu** des ensembles aux groupes de tresses.

- Question : Les résultats sur les tresses sont-ils des **applications** de la théorie des ensembles?
- **Non** : les tresses apparaissent quand les ensembles disparaissent :

DES APPLICATIONS DE LA THEORIE DES ENSEMBLES?

↪ un chemin **continu** des ensembles aux groupes de tresses.

- Question : Les résultats sur les tresses sont-ils des **applications** de la théorie des ensembles?
- **Non** : les tresses apparaissent quand les ensembles disparaissent :
 - La théorie des ensembles donne un exemple (hypothétique) d'un objet (un LD-système ordonnable),

DES APPLICATIONS DE LA THEORIE DES ENSEMBLES?

↪ un chemin **continu** des ensembles aux groupes de tresses.

- Question : Les résultats sur les tresses sont-ils des **applications** de la théorie des ensembles?
- **Non** : les tresses apparaissent quand les ensembles disparaissent :
 - La théorie des ensembles donne un exemple (hypothétique) d'un objet (un LD-système ordonnable),
 - Les tresses et leur ordre apparaissent au cours de la construction d'un exemple alternatif (et "vrai").

DES APPLICATIONS DE LA THEORIE DES ENSEMBLES?

↪ un chemin **continu** des ensembles aux groupes de tresses.

- Question : Les résultats sur les tresses sont-ils des **applications** de la théorie des ensembles?
- **Non** : les tresses apparaissent quand les ensembles disparaissent :
 - La théorie des ensembles donne un exemple (hypothétique) d'un objet (un LD-système ordonnable),
 - Les tresses et leur ordre apparaissent au cours de la construction d'un exemple alternatif (et "vrai").
- **Oui** : si la théorie des ensembles n'avait pas montré que la loi LD est impliquée dans des phénomènes profonds et n'avait pas rendu l'existence de LD-systèmes ordonnables plausible, il est douteux qu'on ait cherché à construire de tels objets...

- En physique : à partir d'une intuition ou d'une évidence **physique**, **deviner** un énoncé, puis le passer au mathématicien pour une démonstration formelle;

- En physique : à partir d'une intuition ou d'une évidence **physique**, **deviner** un énoncé, puis le passer au mathématicien pour une démonstration formelle;
- Ici : à partir d'une intuition **logique** (existence d'un rang autosimilaire), **deviner** un énoncé (existence d'un LD-système ordonnable), puis le passer au mathématicien pour une démonstration formelle;

- En physique : à partir d'une intuition ou d'une évidence **physique**, **deviner** un énoncé, puis le passer au mathématicien pour une démonstration formelle;
 - Ici : à partir d'une intuition **logique** (existence d'un rang autosimilaire), **deviner** un énoncé (existence d'un LD-système ordonnable), puis le passer au mathématicien pour une démonstration formelle;
- ↪ Même si on ne **croit** pas à l'existence d'ensembles (hyper)-infinis, on doit reconnaître que, dans ce cas au moins, de tels objets ont mené à des mathématiques concrètes.