Factorisation dans les semi-groupes de diagrammes

Adrien Nolhier, encadré par Florent Hivert et Adeline Pierrot
au Laboratoire Interdisciplinaire des Sciences du Numérique

6 Septembre 2024, Gif-sur-Yvette, France

Résumé

In this document, we study the properties of the partition monoid (presented in [1]), an algebraic structure with notable submonoids such as the Temperley-Lieb monoid, and its generalization with labelled arcs.

We already know that elements of the partition monoid can be effectively represented by arc-diagrams (see definition 1.10 and following). By adding labels to the arcs of these diagrams, we can obtain other interesting structures. In recent works, Florent Hivert ([2]) etablished in 2024 a bijection between permutations and labelled arc-diagrams that follows some properties, by using Young-Fibonacci lattices. It allowed to prove in all generality that Okada's Algebra dimension is $n!$, where n is the size of associated diagrams.

This document found motivations on these findings. After introducing some definitions and results about diagrams and monoids in parts 1 and 2.1, we present the factorization theorem 2.17 that allows to compute efficiently all minimal factorizations for elements of Okada's monoid, which are labelled-version of Temperley-Lieb monoid. In part 3, we discuss algorithmic design used for experimentation with these monoids, and also size results about labelled versions of others submonoids of the partition monoid.

Table des matières

Remerciements

Ce travail a été réalisé au sein de l'équipe GALaC du Laboratoire Interdisciplinaire des Sciences du Numérique (LISN).

Je tiens tout d'abord à exprimer ma profonde gratitude à Florent Hivert, professeur d'université au sein de l'équipe GALaC, et à Adeline Pierrot, enseignante-chercheuse dans l'équipe Bio-Info, pour leur encadrement tout au long de mon stage, qui s'est déroulé en deux phases : d'abord à temps partiel (une demi-journée par semaine durant le second semestre), puis à temps plein durant les mois de mai, juin et juillet. Florent, grâce à son expertise, m'a initié à ses recherches actuelles et a su me proposer un sujet personnalisé, en partant de thématiques qui m'intéressaient et en les adaptant en fonction de mes choix d'étude. Adeline, par sa rigueur exemplaire et son esprit mathématique, ainsi que ses remarques toujours pertinentes, m'a particulièrement aidé à améliorer la clarté et la qualité de mon raisonnement scientifique. Leur accompagnement a permis de me former à la démarche de recherche.

Je tiens à remercier Benjamin Hellouin, maître de conférence et co-responsable de l'équipe GALaC, Nicolas Thiery et François Pirot, enseignants-chercheurs, Léo Kulinski, doctorant, et tous les autres membres de l'équipe GALaC pour leur accueil bienveillant. Vous m'avez permis de découvrir la vie en laboratoire, tout en m'introduisant à des sujets de recherche passionnants. Grâce à nos échanges ainsi qu'aux séminaires et soutenances auxquels j'ai eu la chance d'assister, j'ai pu enrichir ma compréhension du domaine.

J'ai également eu l'opportunité d'élargir mon expérience au-delà de Paris-Saclay, grâce à Jeanne Scott, chercheuse au département de mathématiques de l'université Brandeis dans le Massachusetts, aux États-Unis, et à James D. Mitchell, professeur de mathématiques à l'université de Saint Andrews en Écosse, tous deux collègues de Florent et par extension très au fait du sujet de mes travaux. Jeanne est venue au LISN et a nous présenté en avant-première son exposé pour FPSAC, et avec James nous avons pu discuter en visioconférence de l'implémentation de la composition de diagrammes.

Plus proche, j'ai aussi eu le plaisir de trouver la sympathie chez les voisins, au Laboratoire Méthodes Formelles (LMF), où j'ai pu échanger sur des questions pointues concernant le langage OCaml, l'algorithmique, et l'état de la science informatique en général. À ce titre, je remercie chaleureusement Sylvain Conchon, mon responsable de formation et professeur à l'université Paris-Saclay, Jean-Christophe Filliâtre, directeur de recherche et co-auteur avec Sylvain de la bibliothèque ocamlgraph utilisée dans ce rapport, Kim Nguyen, maître de conférence et mon professeur pour les cours et projets de programmation fonctionnelle avancée à l'université, Paul Patault, doctorant, et Arnaud Golfouse, doctorant et chargé de TD pour le cours de programmation avancée à l'ÉNS Paris-Saclay.

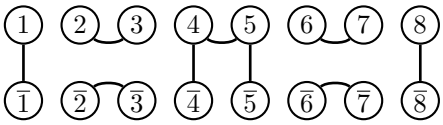
Enfin, je remercie mes amis, dont Maxime, Valéran, Clara, Adam, Hugo et Thomas (qui était aussi mon collègue de bureau) pour leur soutien. Leur patience et leur curiosité à propos de mon sujet de recherche ont très certainement contribué à clarifier mes idées et ont enrichi ma réflexion.

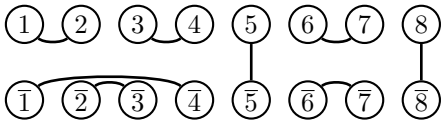
À ma famille ...

Adrien

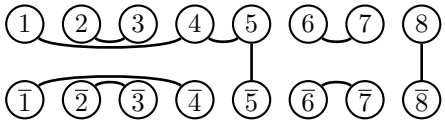
Introduction

Il y a longtemps que l'importance de la théorie des représentations est avérée. Ainsi, il est monnaie-courante d'entendre parler de groupe de permutations, de groupe symétrique ou de monoïde de transformations, et ce dès le premier cycle universitaire. En 2004, Tom Halverson et Arun Ram publient un papier [1] explorant les propriétés de l'algèbre de partition et en particulier, du monoïde de partition, et de son importante fonction de composition. Par exemple, des partitions P et P' de l'ensemble $[0..8] \cup [\overline{0}..\overline{8}]$ pourraient être représentées par les diagrammes D_P et $D_{P'}$:

$D_P =$ 

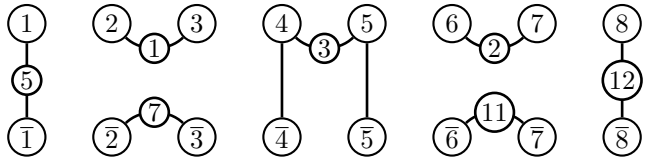
$D_{P'} =$ 

La définition établie dans [1] stipule que la partition obtenue par composition $P \cdot P'$ pourrait être représentée par le diagramme :

$D_{P \cdot P'} =$ 

Cette découverte est majeure car comme nous le décrirons en partie 3, ce monoïde et certains de ses sous-monoïdes ont des propriétés très intéressantes, par exemple l'un est isomorphe au groupe symétrique.

Dans un papier [2] paru en 2024 et présenté à la conférence FPSAC, Florent Hivert et Jeanne Scott construisent une correspondance entre le monoïde d'Okada et le monoïde des diagrammes de partition en **étiquetant ces derniers** suivant certaines règles. Par exemple, P de l'exemple précédent mais étiqueté, pourrait ressembler à cela :



Ensuite, ils prouvent dans le cas général le résultat démontré par Okada dans le cas semi-simple, à savoir que le monoïde d'Okada de paramètre n est en bijection avec les permutations de n éléments.

Mon stage a d'abord comporté une part théorique afin de comprendre les objets manipulés. Ainsi j'ai été introduit aux notions d'algèbre assez communes : présentation d'une structure algébrique par générateurs et relations, ensemble quotienté, puis aux objets manipulés plus précisemment par Florent : diagrammes et tableaux de Young-Fibonacci.

Ensuite, j'ai traversé une phase d'expérimentation où j'ai implémenté (dans le langage OCaml) les structures de diagrammes étiquetés ou non, ainsi que leur opération de composition. Puisque l'étude des diagrammes de partition étiquetés était toute nouvelle, Florent m'a proposé d'étudier les tailles des sous-monoïdes présentés dans [1] en version étiquetées.

Enfin, après avoir acquis une certaines familiarité avec ces objets, j'ai construis puis prouvé un algorithme de factorisation permettant de factoriser un diagramme en le simplifiant suivant n'importe quelle descente, généralisant l'algorithme que Florent avait déjà fait en simplifiant par la plus haute (ou basse) descente. Cela peut permettre de construire l'ensemble des factorisations minimales d'un diagramme, sans utiliser les relations du semi-groupe.

Dans une première partie nous définirons le monoïde de partition tel qu'il est présenté dans [1] et énoncerons des propriétés qui aiderons pour la suite.
Puis, nous définirons le sous-monoïde de partition qui nous intéresse, à savoir Temperley-Lieb, ainsi que sa version étiquetées (monoïde d'Okada) en rappelant de nombreuses définitions et propriétés contenues dans [2]. De plus, nous énoncerons et prouverons le théorème de factorisation permettant la factorisation suivant n'importe quelle descente d'un diagramme du monoïde d'Okada.
En conclusion nous préciserons les grandes lignes de l'architecture du code d'implémentation, puis présenterons les résultats trouvés pour les tailles de sous-monoïdes de partition étiquetés.

1 Monoïde de partitions

Définition 1.1 (Semi-groupe). *Un semi-groupe est un ensemble S muni d'une loi de composition interne associative $\cdot$, c'est à dire telle que pour tous éléments x, y, z de S,*

$$(x \cdot y) \cdot z = x \cdot (y \cdot z)$$

Cette loi sera notée multiplicativement ou omise si possible. ∘

Exemple 1. *Soit S un ensemble. L'ensemble des fonctions définies de S dans lui-même, muni de l'opération de composition des fonctions*

$$f \circ f' = x \mapsto f'(f(x))$$

forme un semi-groupe, appelé semi-groupe de transformations de l'ensemble S.

Définition 1.2 (Monoïde). *Un **monoïde** est un semi-groupe comportant un élément neutre, ou identité, noté e ou 1, c'est à dire tel que pour tout $x \in S$,*

$$e \cdot x = x \cdot e = x$$ ∘

Exemple 2. *Muni de la fonction identité*

$$\begin{aligned} Id_S : S &\to S \\ x &\mapsto x \end{aligned}$$

l'ensemble S de l'exemple 1 forme un monoïde : le monoïde de transformations de S.

Définition 1.3 (Description d'un semi-groupe par générateurs). *Un semi-groupe $(S, \cdot)$ est dit engendré par un ensemble de générateurs $G \subseteq S$ si tout élément $x \in S$ peut s'obtenir par application successive finie de $\cdot$ à des générateurs, c'est à dire qu'il existe un n-uplet $(g_1, ..., g_n)$ d'éléments de S pour $n \geq 1$ tel que*

$$g_1 \cdot \ldots \cdot g_n = x$$ ∘

Théorème 1.4 (Sous semi-groupe engendré). *Soit S un semi-groupe et $X \subseteq S$ un ensemble d'éléments de S. Il existe un plus petit ensemble A incluant X tel que A est un semi-groupe.*
C'est le sous-semi-groupe incluant X et stable par la loi de S. ♢

Définition 1.5 (Relation d'équivalence). *Une relation d'équivalence $\mathcal{R}$ est une relation **binaire** (d'arité 2), **réflexive** ($x\mathcal{R}x$), **symétrique** ($x\mathcal{R}y \Rightarrow y\mathcal{R}x$) et **transitive** ($x\mathcal{R}y \wedge y\mathcal{R}z \Rightarrow x\mathcal{R}z$).*∘

Exemple 3 (module sur les entiers). *On fixe $n \in \mathbb{N}$ et on pose $x\mathcal{R}y$ si $x \equiv y \mod n$.*

Définition 1.6 (Classe d'équivalence). *Soit E un ensemble et $\equiv$ une relation d'équivalence sur cet ensemble, on définit la classe d'équivalence d'un élement $x \in E$, notée $\mathrm{Cl}(x)$, comme l'ensemble*

$$\{y \in E : x \equiv y\}$$

Un représentant d'une classe d'équivalence est un élement fixé de cette classe. ∘

Définition 1.7 (Ensemble quotient). *Soit un ensemble E et une relation d'équivalence $\equiv$. On définit l'ensemble quotient $E/\equiv$ par*

$$E/\equiv = \{[x] : x \in E\}$$

On dit que l'on a quotienté E par la relation $\equiv$. ∘

Exemple 4 (Quotient des entiers relatifs). *$\mathbb{Z}/n\mathbb{Z}$ est l'ensemble quotient de $\mathbb{Z}$ par la relation de l'exemple 3.*

Définition 1.8 (Partition d'ensemble). *Une partition d'un ensemble E est un ensemble de parties non vides de E deux à deux disjointes dont l'union est E.* ∘

Proposition 1.9 (Correspondance entre partitions et relations d'équivalences). *Pour tout ensemble E, la fonction f définie* ***de l'ensemble des relations d'équivalences totales sur E vers l'ensemble des partitions de*** *E et qui à une relation (représentée comme un ensemble de classes) associe une partition :*

$$f : r = \{\mathrm{Cl}(x) : x \in E\} \mapsto \{\{y : y \in Cl(x)\} : Cl(x) \in r\}$$

est une ***bijection****.*

Preuve. Les images de f sont bien des partitions car les classes sont disjointes et puisque les relations sont totales, tout élément de E est exactement dans une part.

Supposons que deux relations $\mathcal{R}$ et $\mathcal{R}'$ distinctes donnent par la fonction la même partition. Par hypothèse, il existe une classe c de $\mathcal{R}$ qui n'est pas dans $\mathcal{R}'$. Ainsi, d'après la définition de f, $c \notin f(\mathcal{R}')$ donc les images sont bien différentes et f est injective.

Supposons qu'une partition $P \in \mathcal{P}(E)$ n'est l'image d'aucune relation par f. Soit $\mathcal{R}$ la relation telle que telle que $x\mathcal{R}y$ ssi il existe $C \in P$ tel que $x \in C$ et $y \in C$. On définit $r = \{\mathrm{Cl}(x) : x \in E\}$ où $\mathrm{Cl}(x)$ est la classe de x dans $\mathcal{R}$. Alors $f(r) = P$ ce qui est contradictoire, donc f est surjective.

Notation

Soit P une **partition**, on note $\equiv_p$ la relation correspondante.

L'ensemble qui nous intéresse dans la suite est $F_n = [n] \cup \overline{[n]}$ où :

$$\begin{cases} [n] = \{i : 1 \leq i \leq n\} \\ \overline{[n]} = \{\overline{i} : 1 \leq i \leq n\} \end{cases}$$

Nous ordonnons ses éléments avec la relation d'ordre totale $\prec$ définie par

$$1 \prec \cdots \prec n \prec \overline{n} \prec \cdots \prec \overline{1}$$

De plus, on attribue une valeur à chaque élément ainsi :

$$\begin{cases} \nu(i) = i, & \text{pour } i \in [n] \\ \nu(\overline{i}) = i, & \text{pour } \overline{i} \in \overline{[n]} \end{cases}$$

L'ordre sur ces valeurs est celui sur les entiers naturels ($<$).

Définition 1.10 (Bi-partition de taille n). *On note $\mathcal{P}_n$ l'ensemble des bi-partitions, c'est à dire l'ensemble des partition de l'ensemble F_n, mais on appelera simplement* ***partition de taille n*** *un élément de $\mathcal{P}_n$.* ∘

Remarque

On peut représenter les fonctions définie de $\mathbb{N}$ dans $\mathbb{N}$ par une bi-partition.
- l'ensemble de départ correspond aux éléments de $[n]$
- l'ensemble d'arrivé aux éléments de $\overline{[n]}$
- puis, si $y \in \overline{[n]}$ est l'image de $x \in [n]$, alors on unifie la part contenant y et la part contenant x.

Ainsi, la loi de composition définie plus loin pour les partitions (étiquetés) donne une généralisation de la composition de fonction classique.

En pratique, une partition se représente avec un graphe, appelé diagramme, où les éléments sont les sommets du graphe.
Les éléments de $[n]$ sont représentés en haut et numérotés croissamment de gauche à droite, et ceux de $\overline{[n]}$ en bas et numérotés de gauche à droite. Une arête entre deux sommets indiquent qu'ils sont dans la même part (ou classe d'équivalence).

Remarque

Pour représenter une classe, pas besoin de relier chaque élément à chaque autre, on prend la convention que $x \equiv_D y$ si et seulement s'il existe un chemin reliant x et y. A priori, il y a donc plusieurs manières de représenter un diagramme.
Pour plus de clarté, on mettra souvent le moins d'arêtes possible.

Les classes d'équivalences correspondent aux composantes connexes du diagrame.

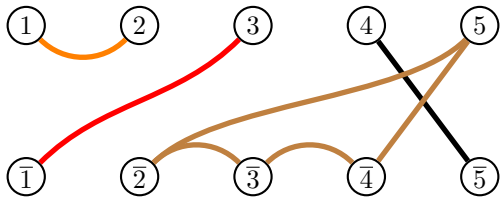


FIGURE 1 – Un diagramme représentant la partition :
$\{\{\overline{5}, 4\}, \{\overline{4}, -3, -2, 5\}, \{\overline{1}, 3\}, \{1, 2\}\}$

Dans la figure 1, on pourrait utiliser une arête de moins pour représenter la composante {-4, -3, -2, 5}.

Définition 1.11 (Loi de composition de partitions). *Soient deux partitions A et B de même taille n, et D_A, D_B des diagrammes qui les représentent. On représente les sommets de D_A par*

$$\begin{cases} a_i \text{ pour } i \in [n] \\ a_{\bar{i}} \text{ pour } i \in \overline{[n]} \end{cases}$$

et de la même manière ceux de D_B avec b_i et $b_{\bar{i}}$.

On va construire le diagrame D_C d'une partition C en ***collant*** *D_B* ***en dessous de*** *D_A et en fusionnant les $a_{\bar{i}}$ avec les b_i. Les sommets de D_C seront donc les a_i (en haut) et les $b_{\bar{i}}$ (en*

bas).
On commence par enlever les arêtes des composantes n'ayant pas de sommets dans

$$\{a_i : i \leq n\} \cup \{b_{\overline{i}} : i \leq n\}$$

qui sont les extrémités de D_C.
Ensuite, on relie deux sommets dans C si dans le diagramme collé ils sont reliés par un chemin.
***On note $\cdot$ la relation telle que** $A \cdot B = C$ où C est la relation représentée par D_C.* ∘

Par exemple avec les partitions $A, B \in \mathcal{P}(n)$ définie par

$$A = \{\{1, \overline{1}\},\ \{2, 3\},\ \{\overline{3}, \overline{2}\},\ \{4, 5, \overline{5}, \overline{4}\},\ \{6, 7\},\ \{\overline{6}, \overline{7}\},\ \{8, \overline{8}\}\}$$
$$B = \{\{1, 2\},\ \{3, 4\},\ \{5, \overline{5}\},\ \{6, 7\},\ \{8, \overline{8}\},\ \{\overline{7}, \overline{6}\},\ \{\overline{4}, \overline{1}\},\ \{\overline{3}, \overline{2}\}\}$$

On obtient :

$$C = \{\{1, 4, 5, \overline{5}\},\ \{2, 3\},\ \{6, 7\},\ \{8, \overline{8}\},\ \{\overline{7}, \overline{6}\},\ \{\overline{4}, \overline{1}\},\ \{\overline{3}, \overline{2}\}\}$$

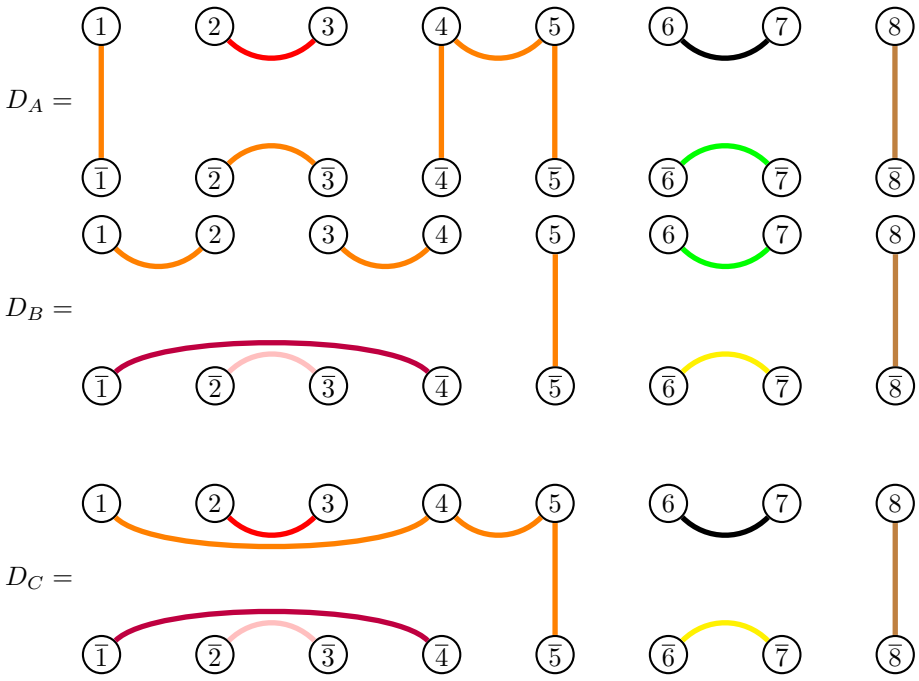


FIGURE 2 – Diagrammes représentant (respectivement, de haut en bas) les partitions A, B et C.

On a coloré les composantes de C, et identifié les futures composantes dans A et B en les colorant de la même manière que dans C.

Proposition 1.12. *Pour toute partition A et B de taille n, $C = A \cdot B$ est une partition de taille n, indépendamment du choix des D_A et D_B.* ♢

L'identité est le diagramme comportant uniquement les arêtes $(i, \overline{i})$, pour $i \leq n$.

Proposition 1.13 (Monoïde des partitions). *$\mathcal{P}_n$ muni de la loi $\cdot$ est un monoïde, appelé **monoïde des partitions**.* ♢

PREUVE. Il faut montrer l'associativité de $\cdot$, voir[1]. ■

Théorème 1.14. *Le monoïde des partitions est engendré par les $3n-2$ éléments :*

— *n éléments p_i, pour $1 \leq i \leq n$, où il y a la liaison $\{j, \overline{j}\}$ lorsque $j \neq i$, c'est à dire*

$$p_i = \bigcup_{j \in [n] \setminus \{i\}} \{j, \overline{j}\}$$

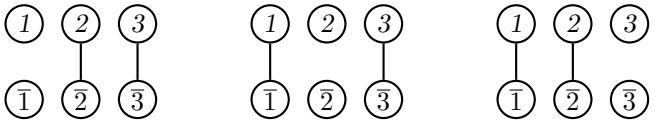


FIGURE 3 – Les diagrammes de p_1, p_2 et p_3 pour $n = 3$.

— *$n-1$ éléments s_i, pour $1 \leq i \leq n-1$, contenant un croisement en i, et des liaisons $j, \overline{j}$ lorsque j est différent de i et $i+1$, c'est à dire*

$$s_i = \{i, \overline{i+1}\} \cup \{i+1, \overline{i}\} \cup \bigcup_{j \in [n] \setminus \{i, i+1\}} \{j, \overline{j}\}$$

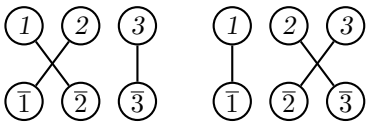


FIGURE 4 – Les diagrammes de s_1 et s_2 pour $n = 3$.

— *$n-1$ éléments b_i, pour $1 \leq i \leq n-1$, contenant le bloc $\{i, i+1, \overline{i+1}, \overline{i}\}$ ainsi que les liaisons $(j, \overline{j})$ pour i différent de i et $i+1$, c'est à dire*

$$b_i = \{i, i+1, \overline{i+1}, \overline{i}\} \cup \bigcup_{j \in [n] \setminus \{i, i+1\}} \{j, \overline{j}\}$$

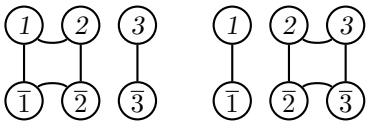


FIGURE 5 – les diagrammes de b_1 et b_2 pour $n = 3$.

Son identité (obtenue par exemple avec $Id = s_i \cdot s_i$ pour $n > 1$) se représente :

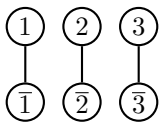


FIGURE 6 – Un diagramme de l'**identité**, pour $n = 3$.

Pour plus d'informations sur ce monoïde, voir [1].

Définition 1.15 (Partition étiquetté). *Soit une partition P de taille n, on peut l'étiqueter avec une application*

$$\text{label} : P \longrightarrow \mathbb{N}$$

qui étiquette les parts de P.

Du point de vu des diagrammes, cela revient à étiqueter une des arêtes de chacune des

composantes connexes, avec la condition que les arêtes d'une même composante ont toutes la même étiquette. ◦

Notation

Il est d'usage de représenter les diagrammes non étiquetés de haut en bas, par contre nous représenterons plutôt les diagrammes étiquetés de gauche à droite, et la composition se fera aussi dans ce sens.

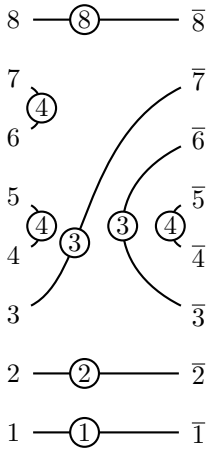
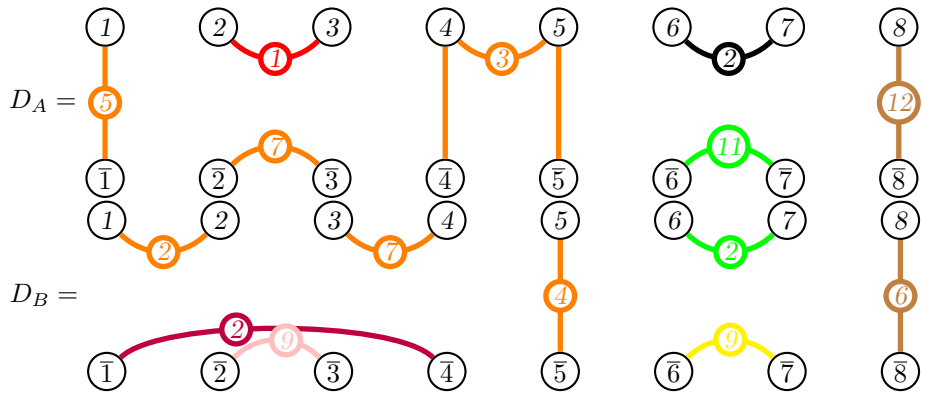


FIGURE 7 – Diagramme d'une partition étiquetée. En particulier c'est un diagramme d'Okada dont nous verrons la définition dans la suite.

Remarque

Pour définir la composition de partitions étiquetés P et P' de taille n, il suffit d'avoir une opération associative et commutative que l'on applique à toutes les étiquettes des arêtes des composantes du résultat.
Ainsi, lors du calcul des arêtes résultantes d'un diagramme de $P \cdot P'$, l'on pourra appliquer cette fonction aux étiquettes des arêtes composante pour donner l'étiquette de leur arête résultante.

Exemple 5. *On peut prendre* ***le minimum des étiquettes des arêtes composantes****.*



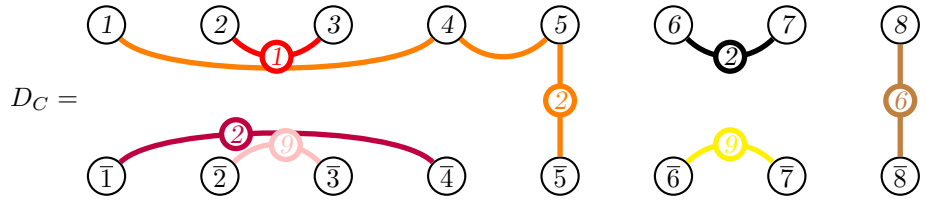


FIGURE 8 – Diagrammes de la figure 2 où nous avons étiqueté arbitrairement A et B puis étiqueté C selon le minimum des étiquettes des composantes.

L'étiquette de la composante $\{8, \overline{8}\}$ *dans* C *est* $\min(12,\ 6) = 6$, *l'étiquette de la composante* $\{1, 4, 5, \overline{5}\}$ *dans* C *est* $\min(5,\ 2,\ 7,\ 7,\ 3,\ 4) = 2$, *et les autres arêtes sont inchangées (pas de nouvelles arêtes dans la composante).*

On discutera de l'implémentation pour effectuer de telles compositions en partie 3.

2 Monoïdes de Temperley-Lieb et d'Okada

2.1 Préliminaires

Définition 2.1 (Monoïde de Temperley-Lieb). *On appelle* ***monoïde de Temperley-Lieb*** *le* ***sous-monoïde*** *du monoïde des partitions généré par les*

$$e_i = b_i p_i p_{i+1} b_i,\ 1 \leq i \leq n-1$$

muni de la loi $\cdot$ *.* ∘

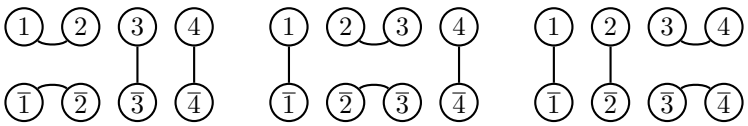


FIGURE 9 – Une base (e_1, e_2, e_3) générant le monoïde de Temperley-Lieb de paramètre $n = 4$.

Un diagramme (éventuellement étiqueté) est dit **non-croisé** s'il n'existe pas deux arêtes (x, y) et (x', y') telles que

$$x' \prec x \prec y' \prec y.$$

FIGURE 10 – À gauche, un diagramme croisé et à droite un diagramme non-croisé (de la partition e_2).

Le diagramme de gauche est croisé car l'arête (1, -3) est croisée avec les arêtes (2, -1) et (3, -2). On peut vérifier que le diagramme de droite n'est pas croisé.

Détail technique

On pourrait se demander si l'intuition visuelle du croisement des arêtes est juste, c'est à dire est-ce que deux arêtes d'un diagramme sont croisées si et seulement si elle se superposent à un endroit du dessin représentant le diagramme.
À cela j'avance que tout diagramme non croisé peut être représenté de manière à ce que visuellement, les arêtes ne soient pas non plus croisées. Il suffit de

- représenter par des droites les arêtes traversantes (ayant une extrémité dans $[n]$ et l'autre dans $\overline{[n]}$)
- représenter les arêtes non-traversantes par un arc de cercle d'une extrémité à l'autre passant par l'intérieur du dessin de diagramme, d'envergure la plus faible possible (càd sans superposer d'autres arêtes non-traversantes).

Réciproquement, si les dessins d'arêtes restent dans le rectangle délimité par les sommets $(1, n, \overline{n}, \overline{1})$ et que les arêtes ne se superposent pas, alors le dessin représente un diagramme non croisé.
Ainsi, tout diagramme non-croisé sera représenté sans superposition, afin d'être fidèle à l'intuition.

Remarque

En toute généralité, c'est possible qu'une partition puisse être représentée à la fois par un diagramme croisé, et par un diagramme non-croisé.

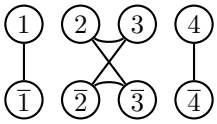


FIGURE 11 – Diagramme croisé représentant la partition e_2, pour n = 4.

Mais **dans le cas de Temperley-Lieb**, nous allons voir que les classes d'équivalences sont toutes de taille deux (donc un seul choix d'arête possible pour représenter les classes) et l'on pourra alors parler indifféremment de partition non croisée ou de diagramme non croisé.

Remarque

Les partitions dont les diagrammes sont croisés sont exactement les partitions nécessitant des symétries (les s_i) pour être construite.
Ainsi, l'ensemble généré par les (p_i, b_i) forme un sous-monoïde du monoïde des partitions, appelé "monoïde planaire", dont nous reparlerons dans la partie 3.

Théorème 2.2 (Caractérisation des diagrammes non-croisés parfaits). *Les diagrammes des partitions du monoïde de Temperley-Lieb (ou simplement "diagrammes de Temperley-Lieb") sont exactement* ***les diagrammes non-croisés dont les sommets sont couplés parfaitements****.*
C'est à dire que chaque sommet est relié à exactement un autre (distinct). C'est équivalent à avoir les classes d'équivalences comportant toutes exactement deux éléments. ♢

Puisque les classes d'équivalences (ou composantes connexes) contiennent une seule arête, il y a **unicité de la représentation d'une partition de Temperley-Lieb** par un diagramme, et nous pourrons alors confondre une partition de Temperley-Lieb et son diagramme.

Corollaire 2.3 (Dénombrement des partitions de Temperley-Lieb). *Soit T_n l'ensemble des partitions de Temperley-Lieb de taille n.*
Puisque cet ensemble correspond à l'ensemble des partitions non croisées de taille n, son cardinal est donné par :

$$|T_n| = C_n = \frac{(2n)!}{n!(n+1)!}$$

où les $(C_n)_{n\in\mathbb{N}}$ sont les ***nombres de Catalans*** *(voir [3]).* ♢

Remarque

L'opération de composition est plus simple dans le cas de Temperley-Lieb. En effet, puisque les composantes connexes sont composés d'exactement une arête, on construit un nouveau couplage en joignant les arêtes d'un chemin, alternant entre arête de D_A et arête de D_B.

Notation

On représente les parts des partitions par (x, y) où $x \prec y$.
Pour les diagrammes étiquetés, lorsqu'on voudra indiquer l'étiquette d'une arête on pourra la noter (x, l, y) où $label(x) = label(y) = l$.

Établissons un lemme suivant, que nous utiliserons dans la partie 2.2 pour démontrer le théorème de factorisation 2.17 en évitant des détails techniques.

Lemme 2.4 (Chirurgie sur les diagrammes par manipulation d'arêtes). *Soit D un diagramme de Temperley-Lieb de taille n étiqueté, et deux collections de m arêtes*

$$A, N \in (F_n * \mathbb{N} * F_n)^m$$

telles que :
— *les arêtes de A sont un sous-ensemble des arêtes de D*
— *les sommets de A et N forment le même sous-ensemble de F_n, c'est à dire :*

$$\bigcup_{(x,y)\in A} \{x, y\} = \bigcup_{(x,y)\in N} \{x, y\}$$

Alors le diagramme D' construit en enlevant dans D les arêtes de l'ancien ensemble A et en ajoutant celles du nouvel ensemble N conserve la structure de diagramme étiqueté couplé parfaitement. ♢

Corollaire 2.5. *Si de plus les arêtes n'induisent pas de croisements, alors le diagramme D' construit dans le précédent théorème est toujours de Temperley-Lieb.* ♢

On dit qu'une arête (x, y) **emboîte**, ou imbrique, une autre arête (x', y') si

$$x \prec x' \prec y' \prec y.$$

Dans l'autre sens, on dira que (x', y') est emboîtée par (x, y).

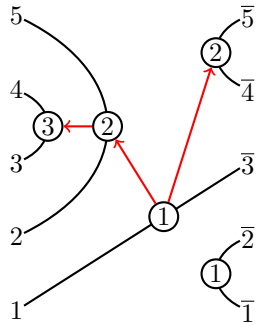


FIGURE 12 – Exemple de diagramme où les flèches rouges indiquent l'ordre d'emboîtement des arêtes.

Dans ce diagramme, $(1, \overline{3})$ emboîte $(2, 5)$ et $(\overline{5}, \overline{4})$, et $(2, 5)$ emboîte $(3, 4)$. On a la propriété de transitivité, donc $(1, \overline{3})$ emboîte aussi $(3, 4)$.

Définition 2.6 (Diagramme d'Okada [2]). *Un diagramme étiqueté non croisé est dit d'Okada s'il suit les propriétés suivantes, appelées règles d'Okada :*

— ***borne des sommets*** *: l'étiquette d'une arête (i, l, j) est comprise entre 1 et le minimum des valeurs de ses sommets, c'est à dire*

$$1 \leq l \leq \min(\nu(i), \nu(j))$$

— ***parité*** *: l'étiquette d'une arête doit être de la même parité que le minimum des sommets en valeur absolue, c'est à dire*

$$l \equiv \min(\nu(i), \nu(j)) \mod 2$$

— ***emboîtement*** *: Si une arête d'étiquette l emboîte une autre arête d'étiquette l', alors $l < l'$* ∘

Définition 2.7 (Loi de composition sur les diagrammes d'Okada). *On définit la loi de composition sur les diagrammes d'Okada (qui sont étiquetés, donc $\cdot$ attend une fonction sur les ensembles d'étiquettes pour être définie) avec la fonction* min.
On note $\cdot_{\mathbf{min}}$ cette loi. ∘

Lemme 2.8 (Stabilité sur les diagrammes d'Okada [2, Lemma 3.6]). *Soient deux diagrammes d'Okada D_a et D_b de taille n, alors $D_c = D_a \cdot_{min} D_b$ est aussi d'Okada.*

Définition 2.9 (générateurs d'Okada). *Dans le cadre des diagrammes étiquetés, E_i pour $i \in [n]$ désigne le générateur e_i de taille n auquel on a étiqueté les arêtes par le minimum de leurs extrémités.* ∘

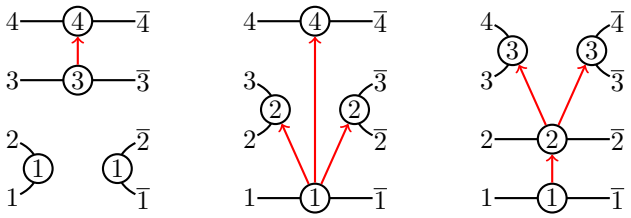


FIGURE 13 – Générateurs étiquetés e_i, pour $n = 4$.

Proposition 2.10. *Les E_i respectent les règles d'Okada.* ♢

Théorème 2.11 (Monoïde d'Okada [2]). *L'ensemble O_n des diagrammes d'Okada de taille n forme un semi-groupe, généré par les E_i, $1 \leq i < n$.*

En étiquetant les arêtes du diagramme identité de la même manière que les E_i (l'étiquette vaut le minimum des extrémités de l'arête), on obtient un élément neutre pour le semi-groupe d'Okada, nous donnant le monoïde d'Okada O_n^1. ♢

PREUVE. Le théorème 1.4 donne l'existence d'un tel ensemble, tandis que la propriété 2.8 assure que tous ses éléments soient d'Okada. ∎

Une arête d'un diagramme D_n est dite **traversante** si elle a une extrémité dans $[n]$ et l'autre dans $\overline{[n]}$.

Théorème 2.12 (Cardinalité du monoïde d'Okada [2]). *Soit O_n le monoïde d'Okada de taille n. Son cardinal est*

$$|O_n| = n!.$$

2.2 Factorisation

Factoriser un diagramme quelconque D c'est trouver une séquence de générateurs $D_1, \ldots, D_m$ de longueur m minimale telle que la composition de ses générateurs donne D c'est à dire :

$$D = D_1 \cdot D_2 \cdot \ \ldots \ \cdot D_m$$

Définition 2.13 (Longueur d'une partition). *On note* $\mathrm{len}(D)$ *la longueur des factorisations minimales du diagramme D.* ◦

Théorème 2.14 (Longueur d'un diagramme d'Okada [2]). *Pour $D \in O_n$ un diagramme du monoïde d'Okada, on a la propriété :*

$$\mathrm{len}(D) = \sum_{(i,l,j)\in D} \frac{|i| + |j| - 2l}{2}$$

Où / est la division rationnelle. ♢

Remarque

Bien que l'on manipule des nombres rationnels, on peut montrer que la somme obtenue est entière.

On appelle **descente** à gauche (resp. à droite) une arête $(i, i, i+1)$ (resp. $(\overline{i+1}, i, \bar{i})$).

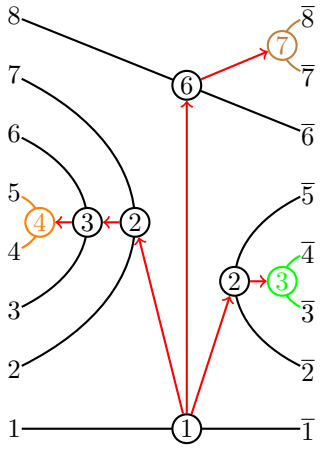


FIGURE 14 – Ce diagramme contient la descente à gauche 4 et les descentes à droite 3 et 7 .

Soit une arête $a = (i, l, j)$. On note $\omega(a) = \frac{i+j-2l}{2}$ son **indice de minimisation**. On dira qu'elle est **minimisable** lorsque $\omega(a) > 0$ (toujours positif ou nul puisque $l \leq \min(i, j)$)

Proposition 2.15. *Une diagramme D est factorisable (c'est à dire différent de l'identité) ssi il possède une arête minimisable.*

PREUVE. C'est une conséquence du théorème 2.14 énonçant que $\text{len}(D) = \sum_{a \in D} \omega(a)$. ∎

Définition 2.16 (Candidat de factorisation). *Soit un diagramme $D \in O_n$ comportant une descente à droite i. On appelle* ***candidat de factorisation pour*** *i (noté CF_i) l'arête ayant pour extrémité le plus grand sommet s tel que $s \prec \overline{i+1}$* ***et dont l'étiquette est inférieure ou égale à i.*** ∘

L'existence de ce candidat est évidente si le diagramme possède une descente à droite car l'arête ayant le sommet 1 pour extrémité est toujours d'étiquette $1 \leq i$. En notant y un sommet vérifiant les hypothèses de la définition, le candidat de factorisation est bien $CF_i = (x, l, y)$ car $x \prec y$.

On notera $\text{CF}_i = (x, l, y)$ avec donc $x \prec y$, mais il est possible que $x \prec i \prec y$ comme il est possible que $x \prec y \prec i$.
En effet, le sommet s peut être x comme il peut être y.

Exemple 6. *Dans le diagramme de gauche de la figure suivante, le candidat de factorisation de la descente à droite 6 est $CF_6 = (8, 6, \overline{8})$, car on a bien $i = 6 \leq 6$.*
Dans le diagramme de droite, la candidat de la descente 5 est $CF_5 = (7, 3, 8)$. En effet, l'arête $(\overline{8}, 7, \overline{7})$ ne convient pas car l'étiquette 7 est supérieure à i.

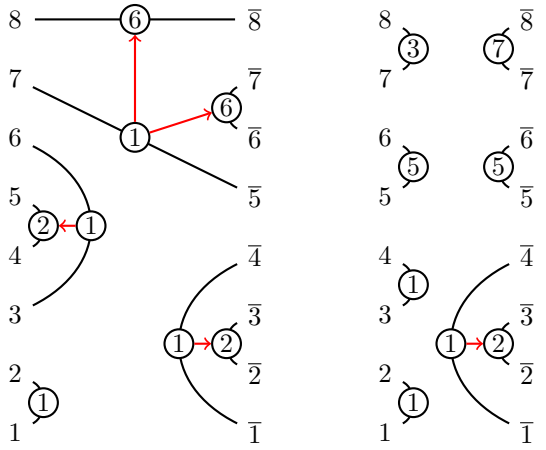


FIGURE 15 – Deux exemples de diagrammes d'Okada.

Dans la preuve du théorème suivant, seul le cas $x \prec y \prec i$ est traité, il faudrait aussi faire $x \prec i \prec y$.

Théorème 2.17 (Factorisation par descente à droite quelconque). *Soit $D \in O_n$ un diagramme de taille n comportant une descente à droite $d = (\bar{i}, i, \overline{i+1})$ d'indice $i, i \leq n-1$. Alors, si l'on note $\mathbf{CF_i} = \mathbf{a} = (\mathbf{x}, \mathbf{l}, \mathbf{y})$ les extrémités et l'étiquette du candidat pour i, le diagramme $D' \in O_n$ défini par :*

$$\begin{cases} D' = (D \setminus \{a, d\}) \cup \{b, c\} \\ b = (x, l, \bar{i}) \\ c = (y, i+1, \overline{i+1}) \\ d = (\bar{i}, i, \overline{i+1}) \end{cases} \tag{1}$$

Vérifie :

$$\begin{cases} D = D' \cdot E_i \\ \text{len}(D) = \text{len}(D') + 1 \end{cases} \tag{2}$$

Ce théorème décrit une chirurgie, à effectuer sur le diagramme D, afin d'obtenir D' par contruction. Nous montrerons que D' suit les bonnes propriétés, ce qui nous permettra de conclure.

On "découpe" a en deux autres arêtes, b et c, de telle sorte à obtenir D' en enlevant a et d à D et en ajoutant b et c.

Par exemple, le diagramme à gauche dans la remarque précédente peut se factoriser avec la descente à droite d'indice 6, et dans ce cas l'arête

$$a = (x, l, y) = (8, 6, \bar{8})$$

est découpée en

$$b = (x, l, \bar{i}) = (8, 6, \bar{6})$$

et

$$c = (y, i+1, \overline{i+1}) = (\bar{8}, 7, \bar{7}).$$

La descente est l'arête

$$d = (\bar{i}, i, \overline{i+1}) = (\bar{6}, 6, \bar{7}).$$

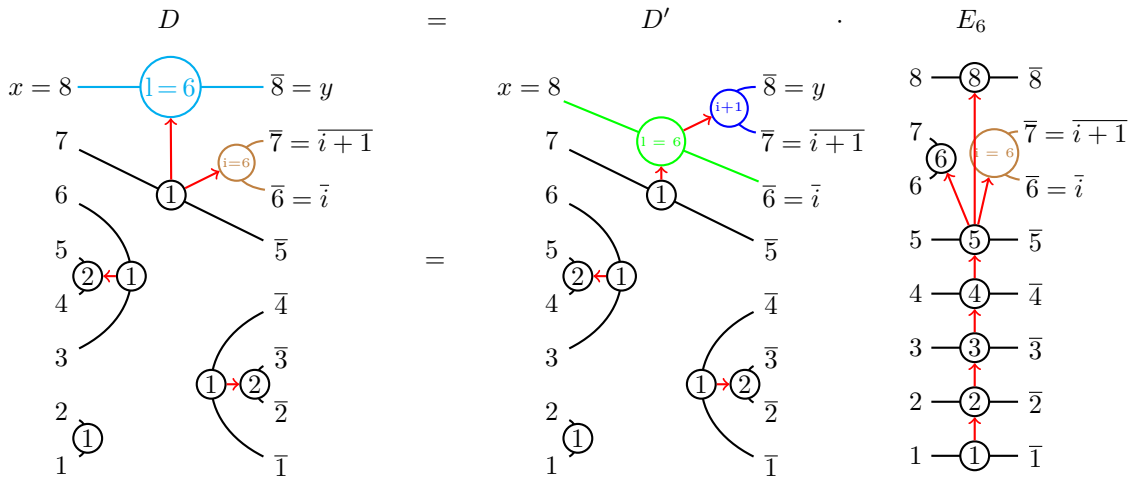


FIGURE 16 – Exemple de factorisation à droite d'un diagramme par la descente 6.

Schématiquement, on utilise E_i pour que $D' \cdot E_i$ conserve la descente i (grâce à la descente droite dans E_i) et joigne b et c (avec la descente gauche dans E_i) afin que $\text{len}(D')$ soit plus petit de 1 de $\text{len}(D)$, càd $\text{len}(D') = \text{len}(D) - 1$.

Lemme 2.18. *L'ensemble D' est un diagramme de Temperley-Lieb.* ♢

PREUVE. Les ensembles d'arêtes $\{a, d\}$ et $\{b, c\}$ contiennent les mêmes sommets, montrons que b et c n'induisent pas de croisement.

— $b = (x, \bar{i})$ et $c = (y, \overline{i+1})$ ne se croisent pas entre-elles car b emboîte c. En effet, $x \prec y$ par définition du candidat de factorisation, et $\overline{i+1} \prec \bar{i}$ d'après la structure du diagramme.

— Vérifions que a et b sont emboîtées par les mêmes arêtes. Il n'y a pas d'extrémité d'arête emboîtant a ni b dans $[\bar{i}, \overline{i+1}]$ car ces sommets sont utilisés par la descente d qui n'emboîte personne. Or a et b ont même extrémités gauche, et l'on vient de justifier que les deux sommets de différence dans l'ordre $\prec$ entre l'extrémité droite de l'arête a, y, et l'extrémité droite de b, $\bar{i}$ ne pouvaient appartenir à des arêtes emboîtante ce qui permet de conclure.

— Vérifions que a et b emboîtent les mêmes arêtes de D. De la même manière que le point précédent, b emboîte seulement la descente d en plus que a, qui n'est pas dans D donc elles emboîtent les mêmes arêtes.

— Vérifions que c n'est pas croisée avec une autre arête. Si $i + 1 < n$ alors $y = \overline{i+2}$ donc $c = (\overline{i+2}, \overline{i+1})$ est une descente et n'est pas croisée.
Sinon $i+1 = n$ alors $y = \overline{n}$ donc $c = (n, \overline{n})$ ne croise personne car il n'existe pas de sommet x tel que $n \prec x \prec \overline{n}$.

On peut maintenant appliquer le corollaire 2.5, ce qui nous permet de conclure. ∎

Proposition 2.19 (Parité des extrémités des arêtes). *Une arête (x, l, y) d'un diagramme d'Okada a ses extrémités de même parité ($x \equiv y \mod 2$) si et seulement si elle est* ***traversante****.* ♢

Lemme 2.20. *Le diagramme D' est un diagramme d'Okada.* ♢

PREUVE. Il suffit de vérifier les propriétés de borne des sommets et de parité pour b et c (puisque les autres arêtes les vérifiaient déjà dans D), puis que b et c respectent l'emboîtement entre-elles, et par rapport aux autres arêtes déjà dans D (entre-elles, les autres arêtes de D le respectaient déjà).

— **borne des sommets pour $\mathbf{b} = (\mathbf{x}, \mathbf{l}, \bar{\mathbf{i}})$:**
Par définition du candidat de factorisation, $\nu(y) \geq \nu(\bar{i})$. Donc on déduit du fait que $a = (x, l, y)$ est d'Okada, que $1 \leq l \leq \min(\nu(x),\ \nu(\bar{i}))$.

— **borne des sommets pour $c = (y, i+1, \overline{i+1})$:**
On a $\nu(y) \geq \nu(\bar{i})$ donc $1 \leq i + 1 \leq \min(y,\ \nu(i+1))$.

— **parité de b :**
— Si $i + 1 < n$, alors $y = \overline{i+2}$. Si b est traversante, alors d'après la propriété 2.19

$$x \equiv \bar{i} \equiv \overline{i+2} \equiv y \mod 2.$$

donc $l = \min(\nu(x),\ \nu(\bar{i}))$ car a est d'Okada impose $l = \min(\nu(x),\ \nu(y))$. Sinon b n'est pas traversante, donc puisque $\bar{i}$ est à droite, x l'est aussi. Or

$$x \prec y = \overline{i+2} \prec \bar{i},$$

donc $\nu(x) > \nu(\bar{i})$. Puisque $\bar{i}$ et $\overline{i+2}$ ont même parité,

$$\min(\nu(x),\ \nu(\bar{i})) = \nu(\bar{i}) \equiv l \mod 2.$$

— Sinon $y = n$. Donc b est traversante car $x \prec y$ et y est à gauche. D'après la propriété 2.19, x et $\overline{i}$ ont même parité. Puisque $x \prec y$ et que x et y sont à gauche, $\nu(x) < \nu(y)$ donc $l \equiv \nu(x) \mod 2$. Finalement,

$$l \equiv \nu(x) \equiv \nu(\overline{i}) \mod 2.$$

— **parité de c :**
On a $\begin{cases} y = \overline{i+2} \\ \text{ ou} \\ y = n \end{cases}$, et dans ces deux cas $\nu(y) \geq \nu(\overline{i+1})$. Donc

$$\min(\nu(y),\ \nu(\overline{i+1})) = \nu(\overline{i+1}) = i+1$$

et la condition est vérifiée.

— **emboîtement entre b et c** :
On a justifié lors de la preuve du lemme 2.18 que b emboîte c. On a bien $l < i+1$ car $\overline{i}$ est une extrémité de b donc $l \leq \overline{i}$.

— **emboîtement entre b et les arêtes de $\mathbf{D \setminus \{a, d\}}$:**
L'arête b dans $(D \setminus \{a, d\}) \cup \{b\}$ emboîte et est emboîtée par les mêmes arêtes que a dans $D \setminus \{d\}$ car elles ont la même extrémité gauche, et parmis les sommets y, $\overline{i+1}$, $\overline{i}$ seul y est utilisé dans le cas de a, et seul $\overline{i}$ est utilisé dans le cas de b. Intuitivement on pourrait fusionner ces trois sommets et les diagrammes obtenues auraient le même comportement.
Ainsi, puisque a et b ont la même étiquette et que a vérifie la condition d'emboîtement par rapport à toutes les autres arêtes dans D, b vérifie la condition d'emboîtement pour toutes les arêtes dans D'.

— **emboîtement entre c et les arêtes de $\mathbf{D \setminus \{a, d\}}$:**
Il n'y a pas d'arête emboîtant c qui n'emboîte pas b car il n'y a pas de sommet entre $\overline{i}$ et $\overline{i+1}$. Or puisque l'on a vérifié l'emboîtement pour b et que b emboîte c, par transitivité, c vérifie la propriété avec toute arête qui l'emboîte.
Ensuite, c n'emboîte aucune arête car c est soit une descente, soit l'arête $(n, \overline{n})$.

On conclut que D' est un diagramme d'Okada. ∎

Lemme 2.21 (Longueur de la factorisation). *On a l'égalité*

$$\text{len}(D) = \text{len}(D') + 1$$

PREUVE. On sait déjà que $\text{len}(E_i) = 1$. Ensuite, par construction,

$$\begin{cases} \text{len}(D') = \text{len}(D \setminus \{a, d\}) + w(b) + w(c) \\ \text{len}(D) = \text{len}(D \setminus \{a, d\}) + w(a) + w(c) \end{cases}$$

Sachant

$$\begin{cases} w(a) = \frac{x+y-2l}{2} \\ w(b) = \frac{x+i-2l}{2} \\ w(c) = \frac{i+1+y-2(i+1)}{2} \\ w(d) = \frac{i+i+1-2i}{2} \end{cases}$$

On veut donc

$$w(a) + w(d) = w(b) + w(c) + 1$$

En injectant les indices de minimisation, on obtient

$$x + y - 2l + 2i - 2i + 1 = x + i - 2l + i + 1 + y - 2(i+1) + 2 \leftrightarrow 1 = 2i + 1 - 2i - 2 + 2$$
$$\leftrightarrow 1 = 1$$

■

Lemme 2.22. *Pour D et D' les diagrammes d'Okada de taille n définis précédemment, et i l'indice de la descente à droite choisie dans D, on a l'égalité*

$$D = D' \cdot E_i$$

PREUVE. Vérifions d'abord que

$$D_{tl} = D'_{tl} \cdot e_i$$

pour D_{tl} et D'_{tl} les diagrammes non-étiquetté de Temperley-Lieb respectivement associés à D et D'.

Lors de la composition $\cdot$, les composantes ne comprennant pas les arêtes a, b, c ni d sont inchangées.
Montrons qu'après la composition, il y a bien une arête $(\overline{i+1}, \bar{i})$ (c'est d) et une arête (x, y) (c'est a). Nous avons indiqué dans la remarque suivant le corollaire 2.3 que dans le cas de Temperley-Lieb, les composantes sont des chemins. Or nous avons le chemin, dans le diagramme D'_{th} collé à e_i,

$$x \to \bar{i} \to \overline{i+1} \to y = (x, y)$$

car $b = (x, \bar{i})$ est incluse par construction, puis $(\bar{i}, \overline{i+1})$ correspond à la descente gauche i de e_i, et enfin $c = (\overline{i+1, y})$ est aussi comprise.

Puis l'arête $d = (\bar{i}, \overline{i+1})$ restant inchangée correspond à la descente i à droite dans e_i, qui reste inchangée lors de la composition. Donc

$$D_{tl} = D'_{tl} \cdot e_i$$

Pour finir la preuve, il faut justifier que les étiquettes des arêtes correspondent.

En dehors des arêtes a, b, c et d, les étiquettes des arêtes de E_i agissent comme l'identité : puisque dans E_i les arêtes (j, $\overline{j}$) pour $j \notin \{i, i+1\}$ sont étiquetés j, elles n'influent pas l'étiquettes de chemins utilisant ces arêtes car c'est l'étiquette maximum possible, et que nous utilisons le minimum des étiquettes composantes pour obtenir l'étiquette d'un chemin (les étiquettes des autres arêtes sont forcément inférieures ou égales).

Ensuite, a d'étiquette l est obtenue par le minimum des étiquettes du chemin (b, (i, i, i+1), c) et on a bien

$$l = \min(i, l, i+1)$$

car on a déjà montré que $i \leq l$.

Enfin, d est bien d'étiquette i puisqu'elle est induite par la descente à droite i de E_i. ■

PREUVE (THÉORÈME 2.17). Les lemmes précédents permettent de conclure la preuve de ce théorème. ■

3 Expérimentation dans d'autres sous-monoïdes

3.1 Organisation du code et fonctionnalités

On peut représenter en OCaml le monoïde des partitions avec un module instancié par un foncteur "Make", prenant en argument le paramètre n des bi-partitions du monoïde.

```
module type PartitionMonoid = sig
  type t = unlabelled_partition

  val id : t (* l'identité *)

  (* les générateurs définis dans Halverson-Ram *)
  val s : int -> t
  val p : int -> t
  val b : int -> t
  val e : int -> t
  val l : int -> t
  val r : int -> t
  (* Une fonction vérifie que l'entier n en argument est dans le
  bon intervalle, et renvoie une erreur à l'utilisateur sinon. *)

  (* [compose a b] renvoie la composition (ou concaténation) c = a . b *)
  val compose : t -> t -> t
  (* opérateur infixe pour la composition *)
  val (@) : t -> t -> t
  (* opérateur infixe pour l'égalité logique entre deux partitions *)
  val (===) : t -> t -> bool
end

module Make : functor (P: sig val n : int end) -> PartitionMonoid
```

Une fois un tel module instancié, nous pouvons par exemple manipuler les générateurs s_i avec la fonction `s` du module, en donnant l'indice i en argument. Ensuite, nous pouvons combiner des partitions avec la fonction `compose`.

L'implémentation que nous venons de décrire est suffisante pour calculer la taille d'un monoïde fixé, par exemple avec l'algorithme suivant (pour des petites valeurs de n) :

```
let generate_semigroup (elts : 'a list) (concat : 'a -> 'a -> 'a) =
  let cache = Hashtbl.create (List.length elts) in (* stocke les éléments du semi-groupe *)
  let rec loop (d: 'a option) : unit =
    match d with
    | None -> List.iter (fun concated -> loop (Some concated)) elts
    | Some d ->
      if not (Hashtbl.mem cache d) then
        begin
          Hashtbl.replace cache d true;
          let nexts = List.map (concat d) elts in
          List.iter (fun concated -> loop (Some concated)) nexts
        end
  in
  loop None; (* le type option permet d'initier le parcours d'une manière assez élégante
                mais l'on pourrait faire autrement pour gagner en performances *)
  let res = Hashtbl.to_seq cache |> List.of_seq |> List.map fst in
  res
```

Explication de l'algorithme

En partant d'un ensemble d'élements, on les combine exhaustivement entre eux pour obtenir de nouveaux éléments, et ce jusqu'à ce qu'on ne puisse plus obtenir de nouveaux éléments. Ainsi, on obtient l'ensemble des éléments du monoïde (pourvu qu'il soit fini) dont le cardinal est la valeur cherchée.

Il est possible d'être bien plus efficace avec l'algorithme de Froidure-Pin [4], qui utilise des relations sur les semi-groupes et monoïdes pour le générer plus finement. L'article [5] en donne des variantes.

Pour manipuler en particulier le monoïde d'Okada mais aussi plus généralement des partitions étiquettés, on ajoute en paramètre du foncteur une fonction symétrique d'arité 2 permettant :
— l'initialisation des générateurs (comment les étiquetter)
— l'implémentation de la composition de deux diagrammes étiquetés
Ainsi, il faut adapter la fonction `compose` qui doit aussi calculer les étiquettes des composantes. Dans nos expérimentations nous avons principalement utilisé la fonction min.

3.2 Structures de données et fonction de composition

Une structure de donnée efficace pour représenter et manipuler des partitions est la structure **Union-Find**, implémentant principalement deux primitives : **Unir** et **Trouver**.
Dans cette structure, une partition est représentée par une forêt (collection d'arbres). La partition $\{\{1\}, \{2\}, ..., \{n\}\}$ est représentée par n arbres de hauteur 1, dont les racines vont de 1 à n.

Puis, si dans notre partition 1 et 2 sont dans la même classe, on unifie (avec la fonction **Unir**) les arbres les contenant. Ainsi les deux arbres-racines 1 et 2 sont fusionnés en un seul, où 1 est la racine, 2 son unique enfant, ou l'inverse. Cette opération se fait même en temps constant.

Pour savoir si deux élements sont dans la même classe, on regarde si la racine de leurs arbres est la même (c'est la fonction **Trouver**). Ainsi, on peut déjà intuiter que (pourvu que nos arbres restent compacts, c'est à dire que les hauteurs des arbres est en $O(log(n_i))$ pour n_i est le nombre d'éléments du i-ème arbre), la primitive $Trouver$ s'effectue en un temps $O(log(n))$ où n est le nombre d'élements de la partition.

Malheureusement, l'on pourrait faire de "mauvais choix" lors de l'unification des arbres, donnant lieu à des arbres dégénérés : par exemple filiforme. Alors, en remontant à la racine depuis une feuille d'un tel arbre, nous pourrions avoir une complexité en $O(n)$. Comme remède à cela, nous pouvons faire de la **compression de chemin** : lorsque nous remontons d'une noeud vers la racine, nous re-branchons directement le noeud à la racine. Ainsi, à terme, nous obtiendrons des arbres "plats" (presque tous les noeuds sont à profondeur 1) et assurer de bonnes performances en *temps amorti*.

Pour les expérimentations, j'ai utilisé les algorithmes implémentant une structure **Union-Find** décrits dans le papier [6] (modulo quelques conversions de syntaxe, de caml light vers OCaml 4), qui ont l'avantage d'être **persistants** et qui couvrait largement mes besoins.

Afin de représenter une partition non étiquetée, on peut utiliser une liste de listes d'entiers :

```
type unlabelled_partition = int list list
```

Cette implémentation a l'avantage d'être facile à manipuler, mais il y a quelques risques de "bug" par exemple s'il manque ou s'il y a trop d'éléments.

Pour représenter une partition étiquettée j'ai choisi d'utiliser un type tel que :

```
type cl = Unique of int | Few of int * int list
type labelled_partition = cl list
```

Remarque

Il peut y avoir une discussion sur la représentation de l'étiquette d'un élement seul (unique). En effet, si à l'inverse nous avions choisi le type

```
type labelled_partition_bis = (int * int list) list
```

un élément isolé pourrait avoir une étiquette différente de sa valeur alors que les diagrammes que nous avons manipuler jusqu'ici correspondaient à des étiquettages seulement d'arête, et non de nœuds isolés.
C'est pourquoi nous préférerons la première implémentation (nous avons quand même testé les deux, et les résultats combinatoires étaient plus intéressants).

Voyons maintenant plus en détail l'algorithme de composition de deux diagrammes, d'abord en pseudo-code.

Remarque

Pour la représentation interne (dans notre code) des éléments des partitions, on numérotera :
- les $i \in [n]$ de 0 jusqu'à $n-1$.
- les $\bar{i} \in [n]$ de n jusqu'à $2n-1$.

Data: Deux partitions A et B de taille k représentée comme des ensembles (listes) de couples
Result: Une partition C telle que $C = A \cdot B$.

```
1  S ← structure Union-Find initialisée à 3 * n éléments;
2  for (x, y) ∈ A do
3  |  unifier(x, y);
4  end
5  for (x, y) ∈ B do
6  |  unifier(x+n, y+n);
7  end
   // On extrait la bi-partition induite par les éléments gauche [0, ..., n-1]
      et les éléments droit [2n-1 ,..., 3n-1] dans S
8  C ← [];
9  for x ∈ [n] do
10 |  for y ← 2n − 1 to 3n − 1 do
11 |  |  if Cl(x) = Cl(y) then
12 |  |  |  C ← (x, y − n) ∪ C
13 |  |  end
14 |  end
15 end
16 return C;
```

Idée de la structure Union-Find

Pour simuler le fait que dans notre composition, les sommets $a_{\bar{i}}$ sont collés (c'est à dire, appartient à la même composante) respectivement aux sommets b_i pour $i \in [n]$, on confond ces derniers deux-par-deux dans la structure Union-Find.
Ainsi, les éléments $\{0..n-1\}$ de la structure Union-Find correspondent aux a_i, les éléments $\{n..2n-1\}$ aux $a_{\bar{i}}$ **et** aux b_i, puis les éléments $\{2n..3n-1\}$ correspondent aux $b_{\bar{i}}$. D'où, au final, $3n$ éléments dans la structure.

Pour l'implémentation en OCaml, on défini des sous-procédures :

```
(* Soit une structure UF déjà définie, on unifie encore les éléments suivant une partition *)
val add_partition_to_uf : Uf.t -> t -> Uf.t

(* on construit une partition à partir d'une structure UF *)
val partition_of_uf : Uf.t -> partition

module type Utils = sig
  type t = unlabelled_partition
  (* fonction de mapping adaptée aux partitions non-étiquetés *)
  val map : (int -> int) -> t -> t
  (* fonction utilitaire *)
  val filter_map : (int -> int option) -> t -> t
  (* fonction de tri *)
  val sort : (int -> int -> int) -> t -> t
end
```

Remarque

On pourrait en toute généralité utiliser un type polymorphe `'a unlabelled_partition` où `'a` serait le type des éléments des partitions, mais dans ce que nous avons fait jusqu'à présent les entiers étaient suffisants, ce qui justifie les signatures de nos fonctions utilitaires.

Puis le code de la fonction `compose` est :

```
let compose (a: t) (b: t) : t =
    (* [0] initialize union-find structure *)
    let uf = Uf.create (3*P.k) in

    (* [1] unify depending on a *)
    let uf = add_partition_to_uf uf a in

    (* [2] unify depending on b *)
    let uf = add_partition_to_uf uf (Utils.map ((+)P.k) b) in

    (* [3] extract C from uf *)
    let c = partition_of_uf uf (3*P.k) in
    let f =
      function
      | n when n <   P.k -> Some n
```

```
  | n when n < 2*P.k -> None
  | n    (*n < 3*P.k*) -> Some (n-P.k)
in
let c = Utils.filter_map f c |> Utils.sort in
c
```

Le code complet (mais peu documenté) utilisé pour ces expérimentations (dont la version étiquetée de la fonction `compose`) est disponible ici [1].

3.3 Résultats

Le monoïde des partitions de taille n noté P_n possède plusieurs sous-monoïde de taille intéressante. Par exemple, $|P_n| = B_{2n}$ où les B_{2k} pour k entier sont les **nombres de Bell pairs**. De plus, nous avions déjà vu que T_n, le monoïde de Temperley-Lieb de paramètre n, avait pour taille le n-ième nombre de Catalan.
Les principaux sous-monoïdes de P_n sont étudiés dans [1] et dans cette partie expérimentale, nous allons regarder **leurs versions étiquetés**, en ré-utilisant la loi min pour la composition, et en étiquetant les générateurs suivant le minimum des sommets des composantes (comme nous l'avons fait dans Okada).

Ainsi, en définissant :

$$e_i = b_i p_i p_{i+1} b_i, \text{ pour i} < \text{n}$$
$$l_i = s_i p_i, \text{ pour i} < \text{n}$$
$$r_i = p_i s_i, \text{ pour i} < \text{n}$$

on note S_i, P_i, B_i, E_i, L_i, R_i les version étiquetés des générateurs (S_1 correspondant à s_1, L_4 à l_4 ect.).
L'identité étiquetée (la même que dans Okada) est notée Id.

Alors, en faisait varier le paramètre n, on calcule les tailles des ensembles générés par des familles de générateurs (S_i pour $i < n$, P_i pour $i \leq n$ ...) et on construit le tableau suivant :

Nom	Générateurs	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$
Partition	$(S_i,\ P_i,\ B_i,\ Id)$	2	17	338	12145	??
Planar partition	$(P_i,\ B_i, Id)$	2	15	173	2673	51030
Rook Brauer	$(S_i,\ E_i,\ P_i, Id)$	2	12	154	3426	117108
Motzkin	$(E_i,\ L_i,\ R_i, Id)$	1	9	77	819	10787
Brauer	$(S_i,\ E_i, Id)$	1	4	40	748	22396
Temperley-Lieb	$(E_i,\ Id)$	1	2	6	24	120
Rook	$(S_i,\ P_i,\ Id)$	2	9	75	1010	20077
Planar Rook	$(L_i,\ R_i,\ Id)$	1	6	29	145	771
Symmetric group	$(S,\ Id)$	1	3	19	209	3545

TABLE 1 – Taille de certains sous-monoïde de partition étiquetés, pour $1 \leq n \leq 5$.

Les valeurs trouvées pour Temperley-Lieb étiqueté (c'est à dire Okada) sont cohérentes avec le théorème 2.12. De plus, la séquence obtenue pour le groupe symétrique correspond à la suite $A204262$ référencée sur "OEIS". Par contre, les autres semblent nouvelles et pas encore étudiées.

1. Dépot contenant le code des expérimentations : `https://github.com/Adrien-No/partition-algebra`.

Ouverture

On pourrait essayer d'autres loi de composition que min pour combiner les étiquettes de deux parts lors de la composition de diagramme, et ainsi découvrir d'autres sous-monoïdes intéressants.

Références

[1] Tom Halverson and Arun Ram. Partition algebras. *European Journal of Combinatorics*, 26(6) :869–921, 2005. Combinatorics and Representation Theory.

[2] Florent Hivert and Jeanne Scott. Diagram model for the okada algebra and monoid. *Sém. Lothar. Combin.*, 50 :12 pages, 2024.

[3] Wikipédia. Partition non croisée — wikipédia, l'encyclopédie libre, 2024. [En ligne ; Page disponible le 5-juillet-2024].

[4] Jean-Eric Pin and Véronique Froidure. Algorithms for computing finite semigroups. pages 112–126, 1997.

[5] M̃arkus Pfeiffer Julius Jonušas, James D. Mitchell. Two variants of the froidure–pin algorithm for finite semigroups. *Port. Math.*, 74 no. 3 pp. 173–200, 2017.

[6] Sylvain Conchon and Jean-Christophe Filliâtre. A persistent union-find data structure. *Proceedings of the 2007 workshop on Workshop on ML*, pages 37–46, 2007.