Académie de Montpellier

Université Montpellier II

- Sciences et Techniques du Languedoc -

Mémoire de D.E.A.

présenté a l'Université des Sciences et Techniques du Languedoc
pour obtenir le Diplôme d'Etudes Approfondies

Formation Doctorale : **Informatique**
Ecole Doctorale : **Information Stuctures Systèmes**

Quelques variations sur la dimension des ensembles partiellement ordonnés

par

Christophe Crespelle

Encadrant : **M. Lhouari Nourine**, Professeur, Univ. Clermont-Ferrand
Correspondants : **M. Michel Habib**, Professeur, Univ. Montpellier II
M. Christophe Paul, Chargé de recherche C.N.R.S., L.I.R.M.M.

Année universitaire 2002-2003

Table des matières

Introduction

La structure de hiérarchie est une représentation fondamentale de l'informatique. Les ordres partiels sont la formalisation mathématique de ces structures. Que ce soit pour représenter la hiérarchie des classes des langages objets, la relation de précédence causale dans les exécutions réparties, ou plus généralement par quotientage des relations transitives, les ordres apparaissent de manière répétée dans tous les domaines de l'informatique. Notons aussi l'utilisation grandissante des ordres particuliers que sont les treillis dans des domaines aussi variés que l'imagerie (avec les complexes simpliciaux), les structures multi-échelles, la représentation des connaissances en intelligence artificielle. Avec la fréquence d'utilisation qui en est faite, on comprend aisément que la question du stockage en mémoire d'ordres partiels est cruciale. Depuis les travaux de Lamport et de Mattern, un des domaines les plus actifs en ce qui concerne le codage d'ordres est l'algorithmique répartie. Cela a constitué le point de départ du travail présenté ici.

La complexité des exécutions réparties dues à l'échange de messages rend difficile leur compréhension. Pour vérifier leur bon déroulement, on a besoin de savoir quels événements peuvent influencer un événement donné. La relation "a pu influencer" sur les événements, appelée relation de précédence causale, est une relation d'ordre. Pour capturer cette relation, on cherche à attribuer à chaque élément une étiquette qui permette de répondre à la question "2 éléments ont ils pu s'influencer lors de l'exécution" ? C'est à dire sont-ils comparables, et dans quel sens, dans l'ordre de précédence causale ? Le codage de la relation d'ordre doit ici répondre à des exigences très fortes : on doit coder l'ordre de manière répartie, par affectation d'étiquette aux événements, on doit pouvoir effectuer ce travail "à la volée" (lors de l'exécution) et la taille des étiquettes doit être assez faible pour ne pas gêner le déroulement de l'exécution, puisque les étiquettes sont transmises dans les messages entre processus. La taille des étiquettes est un point absolument crucial du fait du caractère gigantesque de l'ordre de causalité : le nombre d'événements d'un calcul réparti est colossal.

Les codages d'ordres émanant de l'algorithmique répartie sont des variations autour de la notion fondamentale de dimension d'un ordre partiel introduite en 1941 par Dushnik et Miller (voir [10]). La littérature présentant de tels codages est assez abondante et peu unifiée. Les codages proposés sont parfois très ressemblants sans être vraiment identiques. Beaucoup sont liés entre eux mais il est souvent difficile de comparer leurs performances. Nous avons essayé de clarifier la situation autant que possible et de proposer d'autres variations de la notion de dimension sur lesquelles nous montrons quelques propriétés.

Dans le chapitre 1, nous rappelons quelques définitions sur les ordres partiels et nous donnons une formalisation de la notion de codage par étiquette, en précisant le cadre restreint dans lequel nous nous placerons. Nous produirons plusieurs exemples de codages existants.

Dans le chapitre 2, nous abordons la question de la comparaison des performances des codages. Nous introduisons les notions de codage plus général et de codage plus expressif qui permettent une hiérarchisation des codages.

Dans le chapitre 3, nous précisons ces notions dans le cadre formel de la $\lambda(\mathcal{E})$-réalisation, que nous introduisons. Nous montrons comment les codages présentés et ceux que nous proposons entrent dans ce cadre et nous construisons la hiérarchie qui en découle.

Dans le chapitre 4, nous proposons une approche transversale au codage par intersection de la théorie de la dimension : le codage par union. Nous inscrivons ce codage dans la hiérarchie déjà construite.

Dans le chapitre 5, nous présentons deux nouveaux codages qui sont des variations de la notion de dimension et nous en montrons quelques propriétés.

Nous terminons ce rapport par une conclusion sur le travail effectué et nous énonçons quelques questions ouvertes qui en découlent.

Chapitre 1

Ordres et codages

1.1 Ordres partiels

Relation binaire

Une relation binaire $\mathcal{R}$ sur un ensemble X est une partie de $X \times X$.
On note $x\mathcal{R}y$ pour $(x, y) \in \mathcal{R}$.

$\mathcal{R}$ est dite :
- réflexive si $\forall x \in X, x\mathcal{R}x$
- transitive si $\forall (x, y, z) \in X^3, x\mathcal{R}y$ et $y\mathcal{R}z \Longrightarrow x\mathcal{R}z$
- antisymétrique si $\forall (x, y) \in X^2, x\mathcal{R}y$ et $y\mathcal{R}x \Longrightarrow x = y$

Ordre partiel

Un ordre partiel $\mathcal{P}$ est la donnée d'un ensemble X d'éléments et d'une relation binaire P sur X qui est réflexive, transitive et antisymétrique, noté $\mathcal{P} = (X, P)$.

P vérifiant ces trois propriétés est appellé une relation d'ordre et X est l'ensemble sousjacent à la relation d'ordre.
On note :
$x \leq_{\mathcal{P}} y$ pour $(x, y) \in P$
$x <_{\mathcal{P}} y$ pour $x \leq_{\mathcal{P}} y$ et $x \neq y$
$x||y$ pour $(x, y) \notin P$ et $(y, x) \notin P$

Remarque : On peut déduire $\leq_{\mathcal{P}}$ de $<_{\mathcal{P}}$ en ajoutant la réflexivité, on peut donc définir un ordre $\mathcal{P}$ par la donnée de $<_{\mathcal{P}}$ que par $\leq_{\mathcal{P}}$.

Nous ne considérons dans la suite, sans le préciser, que des ordres partiels sur des ensembles sous-jacents finis.

Ordre linéaire

Un ordre linéaire (ou total) est un ordre partiel dans lequel tous les couples d'éléments sont comparables :

$$\forall (x, y) \in X, x \neq y \Longrightarrow x < y \; ou \; y < x$$

Extension d'un ordre

Un ordre partiel $\mathcal{Q} = (X, Q)$ est une extension d'un ordre partiel $\mathcal{P} = (X, P)$ si :

$$\forall (x, y) \in X, x \leq_{\mathcal{P}} y \Longrightarrow x \leq_{\mathcal{Q}} y$$

Une extension linéaire d'un ordre partiel $\mathcal{P}$ est un ordre linéaire qui est une extension de $\mathcal{P}$.

Ordre partiel biparti

Un ordre partiel $\mathcal{P} = (X, P)$ est un ordre biparti s'il existe une partition de $X = L \sqcup U$ en 2 parties (disjointes) L et U telle que $P \subseteq L \times U$.
On note $\mathcal{P} = (L, U, P)$.
Pour L, U ensembles finis et $L \cap U = \varnothing$, on note $\mathcal{B}(L, U)$ l'ensemble des ordres partiels bipartis $\mathcal{P} = (L, U, P)$sur $L \sqcup U$ tels que $P \subseteq L \times U$.

Isomorphisme d'ordres partiels

Deux ordres partiels $\mathcal{P}_1 = (X_1, P_1)$ et $\mathcal{P}_2 = (X_2, P_2)$ sont isomorphes si et seulement s'il existe une bijection f de X_1 dans X_2 telle que :

$$\forall (x_1, y_1) \in X_1^2, x_1 \leq_{P_1} y_1 \Longleftrightarrow f(x_1) \leq_{P_2} f(y_1)$$

La relation d'isomorphisme sur les ordres partiels est une relation d'équivalence notée $\mathcal{I}so$. Remarquons que tout ordre partiel sur X ensemble fini de cardinal n est isomorphe à un ordre de $\mathcal{O}([\![1, n]\!])$, par simple numérotation des éléments de X. On choisit $\mathcal{O}_n$ un système de représentant des classes d'équivalence $\mathcal{O}([\![1, n]\!])/\mathcal{I}so$. Tout ordre partiel sur X ensemble fini de cardinal n est isomorphe à un unique ordre de $\mathcal{O}_n$. On note $\mathcal{O} = \bigcup_{n \in \mathbb{N}} \mathcal{O}_n$. Tout ordre partiel sur X ensemble fini est isomorphe à un unique ordre de $\mathcal{O}$.
On note $\mathcal{B} = \{\mathcal{P} \in \mathcal{O} \mid \mathcal{P} \; est \; un \; ordre \; biparti\}$.

Pour X un ensemble fini d'éléments, $\mathcal{O}(X)$ est l'ensemble des ordres partiels ayant comme ensemble sous-jacent X.
De même pour $\mathcal{O}' \subseteq \mathcal{O}$, on note $\mathcal{O}'(X)$ l'ensemble des ordres ayant comme ensemble sous-jacent X et qui sont isomorphes à un ordre de $\mathcal{O}'$.

Restrictions d'un ordre partiel $\mathcal{P} = (X, P)$

Restriction par les arêtes

Soit P' une partie de X^2, on note $\mathcal{P}(P')$ ($\mathcal{P}$ restreint à P') la relation d'ordre $\mathcal{Q}$ définie de la manière suivante :
$\mathcal{Q} = (X, Q)$ avec $Q = P \cap P'$ c'est à dire $Q = \{(x, y) \in P' \mid (x, y) \in P\}$

Restriction par les sommets

Soit $Y \subseteq X$, on note $\mathcal{P}[Y]$ ($\mathcal{P}$ restreint à Y) l'ordre partiel $\mathcal{Q}$ définie de la manière suivante :

$$\mathcal{Q} = (Y, Q) \text{ avec } Q = P \cap Y^2$$

Un sous-ensemble X' de X tel que $P[X']$ est un ordre linéaire est appelé une chaîne.

Représentation d'un ordre partiel et relation de couverture

Un ordre partiel $\mathcal{P} = (X, P)$ étant une relation binaire P sur X, il peut naturellement être représenté par un graphe orienté $G = (V, E)$ où $V = X$ et $E = P$; c'est à dire qu'il y a un arc orienté de x vers y si et seulement si $x < y$. Comme la relation P est réflexive, il y a des boucles sur tous les points de V ; comme elle est transitive, le graphe G est transitif ; enfin comme ele est antisymétrique (et transitive), G est sans cycle de longueur au moins 2.

Pour tout $(x, y) \in X^2$, on dit que x est couvert par y ou que y couvre x si $x < y$ et $\nexists z \in X, x < z < y$. On note alors $x \prec y$. Une arête de couverture est une paire (x, y) avec $x \prec y$. Les relations de couvertures suffisent à définir un ordre : les autres arêtes se déduisent par transitivité (et il faut ajouter les boucles). Pour simplifier le dessin d'un ordre , on ne représente que les relations de couvertures, et on s'arrange pour que si $a < b$, alors a soit plus bas que b, afin de ne pas avoir à préciser le sens sur chaque arête. Un tel dessin sera appelé diagramme de Hasse. Voir figure 1.1.

1.2 Coder un ordre

En informatique, le besoin de stocker des ordres partiels en mémoire est fréquent. Il apparaît dans nombre d'applications tels que les langages objets (hiérarchie des classes), l'algorithmique répartie[1] (relation de causalité sur les événements) ainsi que dans de nombreuses structures de données. De la variété des applications faisant naturellement appel au concept d'ordre naît la multiplicité des fonctionnalités qu'on exige, selon les cas d'utilisation, du codage d'ordres. Ces fonctionnalités sont des requêtes que l'on peut faire sur l'ordre et ses éléments, comme la comparaison d'éléments, la recherche des éléments

[1] cette problématique sera traitée plus en détail dans la section 1.3.3

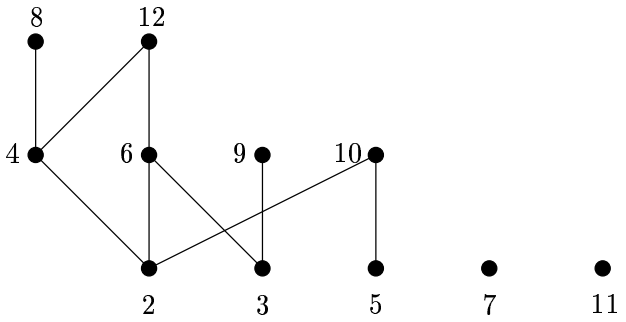

FIG. 1.1 – Diagramme de Hasse de la relation de divisibilité sur $[\![2, 12]\!]$

maximaux, la recherche d'idéaux. Dans toute la suite, nous nous intéresserons uniquement à la comparaison d'éléments, bien que les formalismes de codages dont nous discuterons pourraient fournir d'autres fonctionnalités.

En plus des algorithmes de calcul sur l'ordre, on peut demander à un codage d'autres propriétés tel que l'aspect dynamique, c'est à dire la possibilité de mettre à jour le codage de manière peu coûteuse lors de la modification de l'ordre, ou encore l'aspect réparti, c'est à dire qu'on souhaite distribuer l'information associée à l'ordre sur ses éléments ; c'est naturellement le cas en algorithmique distribuée. Nous nous intéresserons dans la suite uniquement aux codages présentant cet aspect réparti.

Précisément, on cherchera à attribuer une étiquette à chacun des éléments de l'ordre, de telle sorte qu'étant donnés deux éléments, on puisse effectuer la comparaison par le seul examen de leurs étiquettes.

Nous nous limitons à des cas où le traitement des étiquettes est simple. Nous ne ferons pas cas de la complexité de calcul du résultat de la comparaison à partir des étiquettes car dans les codages que nous étudierons, cela se fera toujours en temps linéaire par rapport à la taille des 2 étiquettes.

On distinguera deux aspects du codage :

d'une part, le formalisme des étiquettes et les algorithmes s'y rapportant

d'autre part, les méthodes permettant d'associer à un ordre une information dans le formalisme en question

Nous donnons ci-dessous les définitions constituant le cadre général dans lequel nous inscrivons notre démarche, bien que nous nous placerons dans un cas très restreint de ce cadre.

Définition 1 *Soit F un ensemble de fonctions prenant comme argument un sous-ensemble des éléments d'un ordre. Un codage par étiquettes C offrant les fonctionnalités de F est la donnée d'un ensemble d'étiquettes E et d'un ensemble de fonctions de décodages $D = \{d_f | f \in F\}$ prenant pour argument un ensemble d'éléments de E. On note $C = (E, D)$.*

Exemples : Dans F, on pourrait trouver des fonctions telles que la comparaison de 2 éléments, les éléments maximaux d'un sous-ensemble d'éléments, le

plus petit idéal contenant un sous ensemble de points. En général, l'ensemble des étiquettes E est une partie de $\{0,1\}^*$.

Remarques :

L'ensemble E sera le plus souvent donné implicitement par le formalisme utilisé pour les étiquettes. On ne donnera que ce formalisme, étant sous entendu que E est l'ensemble des étiquettes écrites dans ce formalisme. ex : pour la théorie de la dimension : les vecteurs d'entiers.

La notation $C = (E, D)$ ne fait pas apparaître explicitement l'ensemble F des fonctionnalités de C. Cela ne sera pas génant pour les cas que nous étudierons car F sera toujours réduit à un élément : la fonctionnalité "comparaison de 2 éléments". Il en sera donc de même pour D réduit à un élément d. On notera alors $C = (E, d)$.

Définition 2 *Un code e de $\mathcal{P} = (X, P)$ dans le formalisme du codage C dont l'ensemble de fonctionnalités est F est une application bijective :*

$$\begin{array}{rccc} e: & X & \to & E \\ & x & \mapsto & e(x) \end{array}$$

telle que :

$$\forall f \in F, f(x_1, \ldots, x_n) = d_f(e(x_1), \ldots, e(x_n))$$

On note $C(\mathcal{P})$ l'ensemble des codes de $\mathcal{P}$ dans le formalisme de C et K_C l'ensemble des codes dans le formalisme de C, $K_C = \bigcup_{\mathcal{P} \in \mathcal{O}} C(\mathcal{P})$.

Définition 3 *Une fonction d'encodage K, associée à un codage C, est une fonction qui à tout ordre associe un de ses codages dans le formalisme de C :*

$$\begin{array}{rccc} k: & \mathcal{O} & \to & K_C \\ & \mathcal{P} & \mapsto & k(\mathcal{P}) \in C(\mathcal{P}) \end{array}$$

Nous considèrerons principalement la question du codage sans nous soucier de la fonction d'encodage. Autrement dit, nous nous intéresserons seulement aux performances qu'autorise le codage, sans chercher une fonction d' encodage nous permettant de les atteindre. Cet aspect apparaîtra néanmoins, de manière relative, dans le chapitre 5.

Maintenant posée la définition des codages auxquels nous nous intéresserons dans la suite, se pose la question de l'évaluation de leurs performances et de leur comparaison. De manière générale, l'efficacité d'un codage est évaluée par la place qu'il occupe en mémoire et le temps d'exécution des algorithmes répondant aux différentes requêtes sur l'ordre. D'autre critères peuvent entrer en compte, comme le temps de mis à jour lorsqu'on considère l'aspect dynamique.

Dans le contexte où nous nous plaçons, celui des codages par étiquettes avec comparaison en temps linéaire, le seul critère d'évaluation de l'efficacité d'un codage est la taille des étiquettes. Quelles performances en espace peut on attendre du codage d'ordre par étiquettes ? Sans prétendre répondre à la question, on peut faire les remarques suivantes.

Un ordre partiel étant une relation binaire particulière, il peut être vu comme un graphe orienté. On cherche donc à coder l'adjacence dans une famille particuliaire de graphes orientés. Par conséquent, les méthodes de codage usuelles pour les graphes orientés sont valables et leurs performances sont acquises. Cependant, puisqu'on se restreint à une famille particulière contrainte par des propriétés fortes, on est en droit d'attendre des méthodes de codages pour les ordres plus efficaces que celles existant dans le cas plus général des graphes orientés.

A contrario, un codage réparti (ou par étiquettes) présente des propriétés plus fortes qu'un codage global, et il est donc à prevoir que ses performances en espace globales (si on considère l'ensemble des étiquettes) seront moindres que celles du codage centralisé.

Il paraît donc assez difficile de prévoir les performances que l'on peut atteindre avec le codage par étiquettes.

Nous allons voir dans la suite de ce chapitre quelques codages existants et leurs efficacités respectives.

1.3 Quelques codages existants

1.3.1 Un codage simple

Pour coder un ordre de manière répartie, une solution triviale consiste à répartir l'information globale du graphe orienté correspondant sur les éléments sans tirer profit de la structure d'ordre. Cette solution peut paraître au premier abord sans intérêt, mais il se trouve que la taille des étiquettes dans le pire des cas est nettement meilleure que celle de la plupart des autres méthodes existantes. Ces dernières étant très avantageuses sur des classes particulières d'ordres.

Cette solution répartie consiste à attribuer à chaque élément son numéro codé sur $log\, n$ bits (où n est le nombre d'éléments de l'ordre) et sa ligne dans la matrice d'adjacence codant le graphe orienté représentant l'ordre, dans lequel on a retiré les boucles. On rappelle que dans la matrice d'adjacence, il y a un 1 à l'intersection de la ligne i et de la colonne j si il y a un arc de l'élément numero i vers l'élément numéro j, et il y a un 0 sinon. Ainsi pour comparer deux éléments distincts x et y de l'ordre, on lit le numéro de x et la colonne correspondante dans la ligne de y, s'il y a un 1 alors $x < y$. S'il y a un 0, on lit le numéro de y et la case correspondante dans la ligne de x, s'il y a un 1 alors $y < x$, sinon x et y sont incomparables. Il est à noter qu'ici le traitement des étiquettes ne coûte qu'un temps de l'ordre de $log\, n$.

Exemple :
un codage de l'ordre N_6 défini sur la figure 1.2 est :

$$e(a) = (1, 011011)$$
$$e(b) = (2, 000010)$$
$$e(c) = (3, 000001)$$
$$e(d) = (4, 000011)$$
$$e(e) = (5, 000000)$$

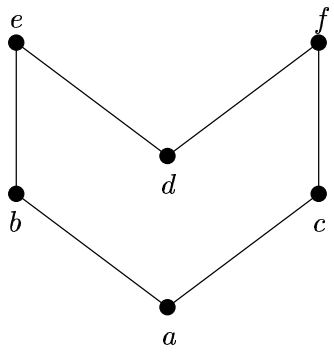

FIG. 1.2 – L'ordre N_6 dit le chevron

$$e(f) = (6, 000000)$$

Cette méthode n'est pas spécifique aux ordres et peut être pratiquée sur n'importe quel graphe orienté. Elle constituera notre référence de complexité en espace dans le pire des cas.

1.3.2 Théorie de la dimension

Nous allons maintenant introduire un codage dont la taille des étiquettes est en $O(log\, n)$ sur de grandes familles d'ordres : celles dont les ordres ont une dimension bornée. Sur ces familles, le codage issu de la théorie de la dimension est bien plus efficace que le codage simple en $O(n)$ que nous avons vu précédemment. En surcroît des performances qu'elle permet d'atteindre, la notion de dimension à un intérêt théorique majeur. La dimension d'un ordre reste la mesure de sa complexité la plus couramment utilisée. Elle permet aussi de donner des caractérisations fortes de certaines familles d'ordres, par exemple, les ordres de dimension 2 sont exactement les orientations transitives des graphes de permutation. Cette théorie joue un rôle central dans l'étude des ordres et tous les codages présentés et introduits dans ce rapport sont des variations plus ou moins amples autour de la notion de dimension. Les notions fondamentales de la théorie de la dimension ont été introduites en 1941 par Dushnik et Miller dans [10].

Réaliseurs d'un ordre partiel

On appelle réaliseur d'un ordre $\mathcal{P} = (X, P)$, une famille $\mathcal{F} = (\mathcal{P}_1, \ldots, \mathcal{P}_t)$ d'ordres linéaires sur X, telle que $P = \bigcap_{i=1}^{t} P_i$.
Remarquons que, nécessairement, les $\mathcal{P}_i$ sont des extensions linéaires de $\mathcal{P}$. On dit aussi que $\mathcal{F}$ réalise $\mathcal{P}$. Une telle famille existe toujours car tout ordre est l'intersection de l'ensemble de ses extensions linéaires. Ce qui justifie la définition suivante.

Dimension d'un ordre partiel

La dimension d'un ordre partiel $\mathcal{P}$, notée $dim(\mathcal{P})$, est le minimum des cardinaux t des familles $\mathcal{F} = (\mathcal{P}_1, \ldots, \mathcal{P}_t)$ qui réalisent $\mathcal{P}$.

Remarque importante sur le vocabulaire :

Un ordre partiel $\mathcal{P}$ est dit de dimension n s'il admet un réaliseur de cardinal n. La dimension d'un ordre partiel $\mathcal{P}$ est donc le minimum des n tel que $\mathcal{P}$ est de dimension n.

$$\mathcal{P} \textit{ est de dimension } n \quad \neq \quad \textit{La dimension de } \mathcal{P} \textit{ est } n$$

Ce vocabulaire, bien que délicat à manipuler, est non ambigü et se révèle très pratique à l'usage. Pour $n \geq 1$, on note D_n l'ensemble des ordres partiels de dimension n.

propriété 1 *La dimension de la restriction d'un ordre à un sous ensemble de ses points est inférieure à celle de l'ordre lui même.*

Le codage

Soit $\mathcal{P} = (X, P)$ un ordre et soit $\mathcal{F} = (\mathcal{P}_1, \ldots, \mathcal{P}_t)$ un réaliseur de $\mathcal{P}$. On attribue à chaque élément de l'ordre une étiquette qui est un vecteur de $[\![1, n]\!]^t$ où $n = |X|$. La i-ième composante du vecteur étant la position de l'élément dans la i-ième extension linéaire du réaliseur. Ainsi, en notant $v(x)$ l'étiquette associée à l'élément x et $v_i(x)$ sa i-ième composante, on a

$$x < y \text{ si et seulement si } \quad \forall i \in [\![1, t]\!], v_i(x) < v_i(y)$$

Exemple : D'après le réaliseur du chevron (ou N_6) de la figure 1.2, les codes des éléments a, b et c sont :

$$\begin{aligned} e(a) &= (0, 1, 0) \\ e(b) &= (1, 4, 1) \\ e(a) &= (2, 2, 3) \end{aligned}$$

Comme $0 < 1$, $1 < 4$ et $0 < 1$, on en déduit $a < b$, et comme $1 < 2$ mais $4 > 2$, on en déduit $b||c$.

Un réaliseur de cardinal t donne donc un codage remplissant les conditions fixées dont la taille des étiquettes est en $t \log n$ bits (voir exemple sur la figure 1.3). En notant $k = dim(\mathcal{P})$, il existe donc un codage avec des étiquettes de $k \log n$ bits. C'est pourquoi sur les familles dont les ordres sont de dimension bornée par k, on a un codage avec des étiquettes en $O(\log n)$, ou plus précisemment en $k \log n$ bits. Ceci revêt un grand intérêt si on code des familles d'ordres dont la dimension est faible devant le nombre de points, ce qui est le cas pour certaines applications du codage d'ordre, typiquement pour le cas de l'algorithmique répartie que nous présenterons dans la prochaine section. La question que

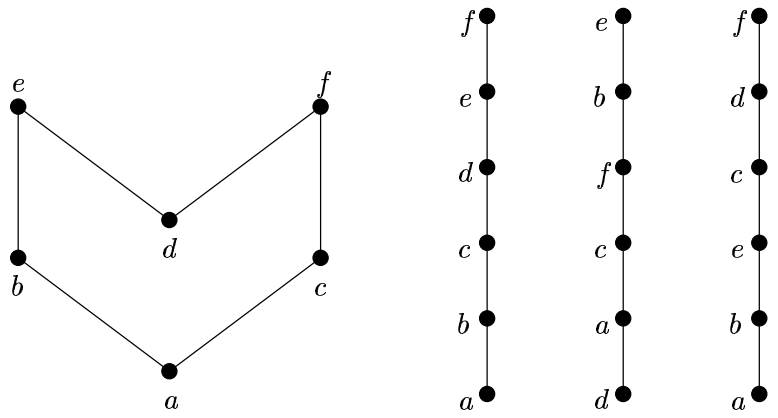

FIG. 1.3 – Code par réaliseur de N_6

l'on se pose alors est : la dimension d'un ordre peut-elle être grande comparée à son nombre de points ? Kelly et Trotter montrent dans [8] que pour $\mathcal{P} = (X, P)$ avec $|X| = n$, $dim(\mathcal{P}) \leq \lceil \frac{n}{2} \rceil$. Et cette borne est atteinte, même sur les bipartis, sur la famille des couronnes $(S_n)_{n \geq 1}$ qui est une famille dont les éléments S_n sont des ordres bipartis avec $2n$ points tels que $dim(S_n) = n$. Nous présenterons cette famille plus en détail dans le chapitre 3.

En terme de codage, cela signifie que, dans le pire des cas, les étiquettes ont $\frac{n}{2} log\, n$ bits, ce qui est nettement moins bon que le codage simple en $O(n)$ vu dans la section 1.3.1. L'intérêt du codage issu de la théorie de la dimension ne réside pas dans ses performances dans le pire des cas mais dans ses performances sur des familles d'ordres particulières : celles dont les ordres ont une dimension bornée.

1.3.3 Codage d'ordres pour les applications réparties

Avec le développement des réseaux de communication, il est devenu courant qu'une application soit exécutée de manière répartie sur plusieurs processeurs qui communiquent entre eux. La conception de telles applications est une tâche ardue. L'orsque l'on distribue un algorithme, il est difficile de prendre en compte tous les comportements possibles. Aussi est il particulièrement intéressant de disposer d'outils pour la mise au point et le test d'un programme réparti. Nous allons voir comment ces outils reposent fondamentalement sur le codage d'ordres partiels.

Modèle d'exécution répartie

Le modèle suivant d'une exécution répartie est courament admis (cf [2] et [1]). Une exécution réparti est constitué d'un ensemble $P_1, \ldots, P_n$ de processus communiquant entre eux exclusivement par échange de messages. Les communications sont supposées point à point, asynchrones et fiables. Les messages arrivent toujours en un temps indéterminé mais fini et un message parti après peut arriver avant un autre. Sur chaque processus, se produisent des événements

considérés comme atomiques, c'est à dire ayant un temps d'exécution nul. Ces événements sont de trois types :

émission de message

réception de message

événement interne au processus

Difficultés d'observation d'une exécution répartie

Les principales difficultés auxquelles on se heurte dans l'analyse des exécutions réparties sont :

- l'impossibilité d'observer un état global instantané du système réparti
- l'impossibilité d'ordonner avec certitude deux événements s'étant produit sur deux processeur géographiquement éloignés

Ces deux problèmes viennent de notre incapacité à obtenir une horloge commune à l'ensemble des processus. Il est à noter que cette incapacité ne relève pas seulement de l'informatique mais de la physique : l'éloignement des processus ne nous permet pas de considérer le temps comme une échelle absolue. Contrairement au cas d'un programme séquentiel, il n'y a pas d'ordre total sur les événements d'une exécution répartie.

Relation de causalité

Si on ne peut pas ordonner totalement l'ensemble E des événements d'une exécution répartie, il n'en existe pas moins des relations de précédence temporelle entre certains événements :

- les événements se produisant sur un même processus sont totalement ordonnés par l'horloge locale du processeur sur lequel il s'exécute
- de plus, l'émission d'un message en précède nécessairement la réception

Les événements d'une exécution répartie sont donc partiellement ordonnés par la fermeture transitive de ces relations binaires.

On appelle cet ordre partiel l'ordre de causalité, il est défini ainsi (voir [2]) :

Pour deux événements e_1 et e_2 de l'ensemble E des événements d'un calcul réparti, on dit que e_1 précede causalement e_2 et on note $e_1 < e_2$ si l'une au mois des trois conditions est vraie :

1. e_1 et e_2 se sont produits sur le meme processus et e_1 a eu lieu avant e_2.
2. e_1 est l'emission d'un message et e_2 est la réception correspondante.
3. Il existe un événement e_3 de E tel que $e_1 < e_3$ et $e_3 < e_2$.

En 1980, Fidge et Mattern (dans [3] et [4]) ont proposé un algorithme pour capturer l'ordre de causalité au cours d'une application répartie. Cela est réalisé en associant sur chaque processus, de manière locale, une étiquette de n entiers (où n est le nombre de processus). Il s'agit donc d'un codage par étiquettes au sens où nous l'avons défini précédemment.

Dans [5] , Garg et Skawratananond reprennent et exploitent ce résultat en termes d'ordres partiels. On rappelle ici le cadre formel dans lequel ils inscrivent ce codage.

String

Un ordre partiel $\mathcal{P} = (X, P)$ est une string si et seulement si $\exists f : X \to \mathbb{N}$ telle que $\forall x, y \in X : x < y \ ssi \ f(x) < f(y)$.

On note S l'ensemble des strings. La notion de string est la même que celle d'ordre faible introduite par Kim et Lee dans **??** (voir section 3.2).

String Realizer

On modifie légèrement la définition présente dans l'article original [5], car elle ne correspond pas à l'usage qui en est fait dans cet article. On remplace $\forall s \in S : x \leq_s y$ par $\forall s \in S : y \not<_s x$.

Pour tout ordre $\mathcal{P} = (X, P)$, un ensemble de strings S est appelé un string realizer ssi $\forall x, y \in X : x < y$ dans $\mathcal{P}$ si et seulement si

1. $\forall s \in S : y \not<_s x$, et
2. $\exists t \in S : x <_t y$.

String Dimension

Pour tout ordre $\mathcal{P} = (X, P)$ la string dimension de $\mathcal{P}$, notée $sdim(\mathcal{P})$, est le cardinal du plus petit ensemble de strings S tel que S est un string realizer de $\mathcal{P}$.

Le codage

Si $\mathcal{S} = (X, S)$, avec |X|=n, est une string alors il existe une unique fonction f de X dans $\mathbb{N}$ satisfiant la définition de string et telle que l'image de f est un segment $[\![1, k]\!]$, de plus on a $k \leq n$.

k est appelé la hauteur de la string, et on appelle place de $x \in X$ dans la string, l'entier associé à x par cette fonction f.

Etant donné un string realizer de $\mathcal{P} = (X, P)$, on obtient un codage en attribuant à chaque élément x de l'ordre une étiquette $e(x) = (e_1(x), \ldots, e_t(x))$ qui est un vecteur d'entier dont la $i^{ème}$ composante $e_i(x)$ est la place de x dans la $i^{ème}$ string du string realiseur qui est de cardinal t. On a alors $x < y$ si et seulement si :

$e(x) < e(y)$ c'est à dire $\forall i \in [\![1, t]\!], e_i(x) \leq e_i(y) \quad$ et $\quad \exists j \in [\![1, t]\!], e_j(x) < e_j(y)$

Exemple : d'après le string realizer de N_6 de la figure 1.4. Les codes des éléments a, b, c sont :

$$e(a) = (0, 1, 0)$$
$$e(b) = (0, 2, 0)$$
$$e(c) = (0, 1, 1)$$

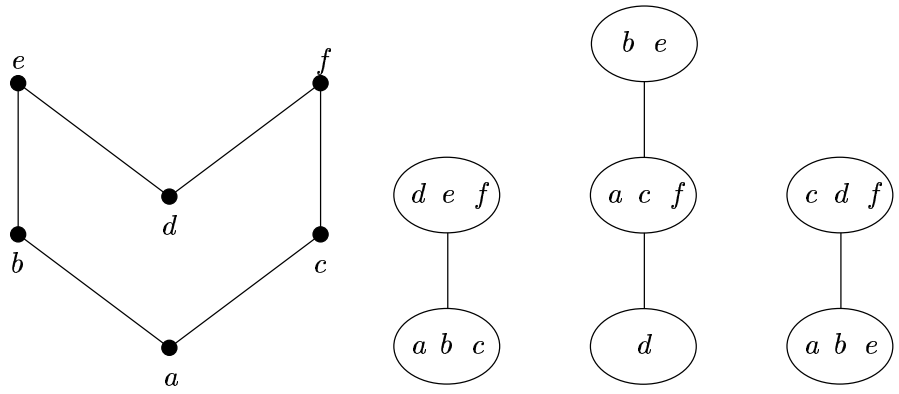

FIG. 1.4 – Code par string realizer de N_6

$a < c$ car $0 \leq 0$, $1 \leq 1$ et $0 < 1$. Par contre, comme $2 > 1$ et $0 < 1$, b et c sont incomparables.

L'algorithme de Fidge et Mattern construit bien un codage par string realizer au cours de l'exécution répartie, localement et à la volée.

On peut aussi remarquer que le codage par string realizer est une généralisation du codage issu de la théorie de la dimension. En effet, il est aisé de voir qu'un ordre linéaire est une string. De plus, si on applique la définition du string realizer sur des ordres qui sont linéaires et non pas des strings, on obtient exactement la définition de réaliseur. Tout réaliseur d'un ordre est un string realizer particulier. C'est en cela que nous dirons que la notion de string realizer et le codage qui en découle sont une généralisation de la notion de réaliseur et du codage qui en découle.

Le problème de définir la notion de généralisation de codage est l'objet principal des sections 2.2 et 3.1.

Tout code en terme de dimension est donc aussi un code en terme de string dimension. La question qui est alors naturelle de se poser est : la notion de string dimension permet-elle de coder un ordre de manière plus efficace que celle de dimension ? Nous y répondrons dans le chapitre 3.

Garg et Skawratananond montrent dans [5] que :

$$\forall \mathcal{P} \in \mathcal{O}, dim(\mathcal{P}) \geq 2 \Longrightarrow sdim(\mathcal{P}) = dim(\mathcal{P})$$

Il faut donc autant de strings que de chaînes pour coder les ordres de dimension supérieure à 2. Cependant, les strings peuvent être de hauteur inférieure à n et donc nécessiter moins de bits pour coder la place d'un élément. C'est de cette façon que le codage en strings peut être meilleur que le codage en chaînes.

1.3.4 Réalisation d'un ordre partiel à l'aide d'ordres de dimension 2

Réalisation

Une famille $\mathcal{F} = (\mathcal{P}_1, \ldots, \mathcal{P}_t)$ d'ordres partiels de dimension 2 sur X est une réalisation de $\mathcal{P} = (X, P)$ si :

$$x <_{\mathcal{P}} y \quad \text{si et seulement si} \quad \exists i \in [\![1,t]\!], x <_{\mathcal{P}_i} y \text{ et } \nexists j \in [\![1,t]\!], y <_{\mathcal{P}_j} x$$

On dit aussi que $\mathcal{F}$ réalise $\mathcal{P}$.

Epaisseur d'un ordre partiel en ordres de dimension 2

Soit $\mathcal{P} = (X, P)$ un ordre partiel, on appelle épaisseur en ordres de dimension 2 de $\mathcal{P}$ le minimum des cardinaux t des familles d'ordres de dimension 2 $\mathcal{R} = (\mathcal{R}_1, \ldots, \mathcal{R}_t)$ qui réalisent $\mathcal{P}$.

On note : $epais_2(\mathcal{P}) = t$.

Le codage

Pour coder les ordres de dimension 2 de la réalisation, on utilise la théorie de la dimension : on les code par deux chaînes. On peut faire mieux en utilisant la string dimension et en les codant par deux strings. De la Higuera et Nourine montrent dans [7] comment obtenir un string realizer optimal pour les ordres de dimension 2.

Etant donné une réalisation en ordres de dimension 2 de $\mathcal{P} = (X, P)$, on obtient un codage en attribuant à chaque élément x de l'ordre une étiquette $e(x) = ((a_1(x), b_1(x)), \ldots, (a_t(x), b_t(x)))$ qui est un vecteur de couples d'entiers dont le $i^{ème}$ couple est le code de x dans le $i^{ème}$ ordre de la réalisation qui est de cardinal t. On a alors $x < y$ si et seulement si :

$$\exists i \in [\![1,t]\!], (a_i(x), b_i(x)) < (a_i(y), b_i(y)) \quad \text{et} \quad \forall j \in [\![1,t]\!], (a_j(y), b_j(y)) \not< (a_j(x), b_j(x))$$

Exemple : D'après la réalisation à l'aide d'ordres de dimension 2 de N_6 donnée sur la figure 1.5, les codes des éléments a, b, c sont :

$$\begin{aligned} e(a) &= ((1,0),(0,5)) \\ e(b) &= ((5,1),(1,5)) \\ e(a) &= ((2,4),(3,4)) \end{aligned}$$

Comme $(1,0) < (5,1)$ et $(1,5) < (0,5)$ alors $a < b$. Par contre $(5,1)||(2,4)$ et $1,5)||(3,4)$ d'où $b||e$.

Le codage par réalisation à l'aide d'ordres de dimension 2 est une généralisation du codage par string realizer, et donc du codage issu de la théorie de la dimension. En effet, une string est un ordre de dimension 2 et si on applique la définition de la réalisation sur des ordres qui sont des strings et non pas des ordres de dimension 2, on obtient exactement la définition du string realizer. Tout string realizer d'un ordre est une réalisation à l'aide d'ordres de dimension 2 particulière. Nous dirons donc que la notion de réalisation à l'aide d'ordres de dimension 2 et le codage qui en découle sont une généralisation de la notion de string realizer et du codage qui en découle. Tout code par string realizer est

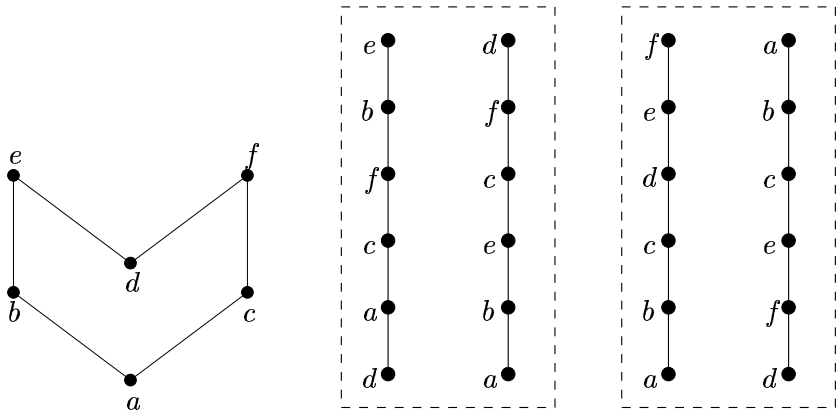

FIG. 1.5 – Réalisation en ordre de dimension 2 de N_6

donc aussi un code par réalisation à l'aide d'ordres de dimension 2. La question que l'on se pose alors est celle de savoir si la notion de réalisation en ordres de dimension 2, qui code toujours au moins aussi bien que la notion de string realizer, permet de coder certains ordres de manière plus efficace que ne le font les string realizers ? Nous répondrons à cette question dans le chapitre 3.

Forts de ces quelques exemples, nous allons maintenant aborder le problème de la comparaison des codages entre eux.

Chapitre 2

Classification et comparaison des codages

Dans cette section, nous essayons de donner une idée des difficultés auxquelles on se heurte lorsqu'on essaye de comparer les performances de différentes méthodes de codages, même dans le cadre très restreint de notre étude. Nous discuterons d'abord des divers critères de comparaison quantitatifs utilisés ainsi que de leurs limites, puis nous proposerons une méthode de comparaison qualitative pour des codages ayant des liens forts tissés par la relation de généralisation que nous introduirons.

2.1 Comparaison quantitative des performances

Il existe quantité de critères d'efficacité sur lesquels on peut évaluer les codages et les comparer entre eux. Dans [11], Eric Thierry présente une synthèse des codages existants et de leurs performances au regard de ces critères d'évaluations. Les deux principaux sont la taille de l'information stockée en mémoire pour décrire l'ordre et le temps de calcul des requêtes s'y appliquant. La plupart du temps, on cherche à coder un ordre statique sur lequel on veut faire un grand nombre de calculs, c'est pourquoi ces deux critères ont la plus grande importance. Cependant, il arrive qu'on ne fasse pas une quantité de calculs assez importante pour négliger le temps de génération du code de l'ordre, c'est à dire le temps de calcul de la fonction d'encodage (cf définition 3 page 7). Cette caractéristique constitue un autre point de comparaison. Il y en a encore bien d'autres qui naissent des besoins spécifiques auxquels répondent les codages, citons par exemple le temps de mise à jour dans le cas dynamique.

La classification et la comparaison des méthodes de codage d'ordre dans le cas général est donc un problème difficile. Il peut être surprenant de voir que dans le cadre très restreint dans lequel nous nous plaçons (les codages par étiquettes avec comparaison d'éléments en temps linéaire, cf section 1.2), la question de la comparaison conserve bonne part de sa difficulté. Pourtant, un seul critère de comparaison subsiste[1] : celui de la taille du codage, c'est à dire ici de la taille des étiquettes.

[1] encore une fois, nous écartons la question du temps de génération du code

La difficulté apparaît dans la comparaison entre le codage simple de la section 1.3.1 et le codage issu de la théorie de la dimension, section 1.3.2. Dans le premier, les étiquettes ont une taille dans le pire des cas en $n+log\,n = O(n)$, alors que dans le second, la taille dans le pire des cas est en $O(n\,log\,n)$. Cependant, sur toutes les familles d'ordres de dimension bornée, le second codage est en $O(log\,n)$ alors que le codage simple reste en $O(n)$.

L'efficacité d'un codage dépend en fait grandement de l'utilisation qu'on en fait. La taille des étiquettes dans le pire des cas n'est pas toujours le critère pertinent pour juger de l'efficacité d'un codage, puisque dans la plupart des cas on ne codera que des ordres ayant des structures particulières pour lesquels on peut trouver des codages plus efficaces que le codage général, bien qu'on n'ait pas une garantie de performance aussi bonne dans le pire des cas. Un codage intéressant sera un codage qui a de bonnes performances sur des familles d'ordres de taille importante. La difficulté étant de jauger la taille d'une famille (infinie) d'ordres.

En marge de cette comparaison quantitative, difficile mais qui permet la comparaison de codages par étiquettes très différents, nous allons introduire, dans la section suivante, un autre outil de comparaison permettant de hiérarchiser des codages très proches.

2.2 Une hiérarchie qualitative de codages

L'objet de cette section est de comparer des codages assez similaires, présentant de faibles variations autour de codages donnés. Nous effectuerons ce travail dans le chapitre 3 pour des codages proches de celui issu de la théorie de la dimension. Les codages que nous considérons sont liés par la relation de généralisation, qui est apparue lors de l'étude des quelques codages de la section 1.3.

Intuitivement, on pourrait poser la définition suivante.

Définition 4 *Un codage C_2 est plus général qu'un codage C_1 si le formalisme dans lequel sont écrites les étiquettes de C_2 contient le formalisme de C_1 comme cas particulier et les fonctions de décodage de C_2 appliquée aux cas particuliers de C_1 donne les mêmes résultats que les fonctions de décodage de C_1.*

Cette définition, satisfaisante sur un plan intuitif, est difficile à formaliser. Cependant, nous donnerons une définition rigoureuse de la relation de généralisation entre codages lorsque les 2 codages à comparer sont des codages de type $\lambda(\mathcal{E})$-réalisation, une notion que nous introduirons dans la section 3.1.

Exemple :
Le codage C_2 par réalisation en ordres de dimension 2 est une généralisation du codage isssu de la théorie de la dimension C_1. En effet, un ordre linéaire est aussi de dimension 2. De plus, si on applique la définition de la réalisation (section 1.3.4, page 14) aux ordres linéaires de C_1, comme dans un ordre linéaire $x < y$ ou $y < x$, on obtient exactement la définition de l'intersection de ces ordres.

Si un codage est plus général qu'un autre, il est au moins aussi efficace puisqu'il contient tous les codes de l'autre comme cas particuliers. La question est de savoir s'il est vraiment plus efficace, on dira plus expressif.

Définition 5 *Pour un codage C_2 plus général qu'un codage C_1, on dira que C_2 est plus expressif que C_1 si et seulement si il existe une famille $\mathcal{F} = (\mathcal{P}_n)_{n\in\mathbb{N}}$ d'ordres partiels qui soient codables par C_2 avec des étiquettes de taille assymptotiquement négligeable devant celle de tout codage par C_1, soit de manière formelle :*

$$\exists \mathcal{F} = (\mathcal{P}_n)_{n\in\mathbb{N}} \in \mathcal{O}^{\mathbb{N}}, \exists (c_n^2)_{n\in\mathbb{N}} \in K_{C_2}^{\mathbb{N}},\ (\forall n \in \mathbb{N}, c_n^2 \in C_2(\mathcal{P}_n)) \quad et$$

$$\forall (c_n^1)_{n\in\mathbb{N}} \in K_{C_1}^{\mathbb{N}},\ (\forall n \in \mathbb{N}, c_n^1 \in C_1(\mathcal{P}_n)) \implies \lim_{n\to+\infty} \frac{\max\limits_{x\in X_n}\{|c_n^2(x)|\}}{\max\limits_{x\in X_n}\{|c_n^1(x)|\}} = 0$$

Exemples : le lecteur peut se reporter dès maintenant à la section 3.2 où nous mettons en oeuvre la notion de codage plus efficace.

Dans le chapitre suivant, nous appliquons ces notions, en les précisant, dans le but de construire une hiérarchie de quelques codages variant autour de la notion de dimension.

Chapitre 3

Relations hiérarchiques de quelques codages

Comme nous l'avons vu dans la section 1.3, bon nombre de codages existants sont naturellement liés par des relations de généralisation. Nous avons montré qu'un codage plus général permet d'atteindre les performances d'un codage particularisé. nous avons introduit la notion de codage plus expressif pour différencier les généralisations qui apportent un pouvoir de codage nettement supérieur à celui de leur particularisation, des généralisations qui ont des performances équivalentes à leur particularisation.

Dans ce chapitre, nous classons en hiérarchie des codages rencontrés dans la littérature et nous montrons que ces codages entrent dans un formalisme que nous intoduisons, celui de la $\lambda(\mathcal{E})$-réalisation, plus restreint que le formalisme général introduit dans la section 1.2. Ce formalisme nous permet notamment de donner une définition rigoureuse de la relation de généralisation entre codages.

3.1 Cadre formel de la $\lambda(\mathcal{E})$-réalisation

Dans la section 1.3, nous avons vu différentes méthodes de codage s'appuyant sur les notions de réaliseur, de string realizer et de réalisation à l'aide d'ordres de dimension 2. Nous voulons maintenant fournir un cadre formel commun à ces notions, dans lequel s'inscriront également les codages que nous introduirons. Le but de ce cadre commun est de faciliter la comparaison qualitative de codages telle que nous l'avons définie en 2.2.

On peut remarquer que dans les types de codages présentés, les réaliseurs, les string realizers et les réalisations à l'aide d'ordres de dimension 2 sont des familles finies d'ordres.

Ces familles d'ordres sont différenciées par les caractéristiques suivantes :

- elles sont composées d'ordres particuliers : les ordres linéaires pour les réaliseurs, les strings pour les string realizers et les ordres de dimension 2 pour les réalisations à l'aide d'ordres de dimension 2.
- la manière de déduire l'ordre $\mathcal{P} = (X, P)$ qu'elles représentent, c'est à dire de déduire la relation entre deux éléments $(x, y) \in X^2$, est différente d'une notion à l'autre.

Exemple :

pour les réaliseurs, $x <_{\mathcal{P}} y$ ssi $x < y$ dans tous les ordres du réaliseur.

pour les string realizers, $x <_{\mathcal{P}} y$ ssi $x < y$ dans au moins un ordre de la famille et dans aucun des ordres de la famille on a $y < x$.

Dans le cadre formel que nous proposons, cette méthode de déduction de la relation entre x et y est décrite par un prédicat $\lambda(\mathcal{F}, x, y)$ où $\mathcal{F}$ est la famille considérée.

Remarquons aussi que dans ces différentes notions, les ordres des familles considérées ont toujours le même ensemble sous-jacent X qui est celui de l'ordre $\mathcal{P} = (X, P)$ qu'ils représentent.

λ-réalisation d'un ordre partiel

Pour A un ensemble quelconque, on note $\mathcal{P}_f(A)$ l'ensemble des parties finies de A.

On pose

$D = \{(\mathcal{F}, x, y) |$ il existe un ensemble fini X tel que $\mathcal{F} \in \mathcal{P}_f(\mathcal{O}(X))$ et $(x, y) \in X^2\}$

Définition 6 *Soit un prédicat*

$$\begin{array}{rccc} \lambda: & D & \rightarrow & \{vrai, faux\} \\ & (\mathcal{F}, x, y) & \mapsto & \lambda(\mathcal{F}, x, y) \end{array}$$

On dit qu'une famille $\mathcal{F}$ d'ordres partiels sur X est une λ-réalisation de $\mathcal{P} = (X, P)$ si et seulement si :

$$\forall (x, y) \in X^2, x <_{\mathcal{P}} y \Longleftrightarrow \lambda(\mathcal{F}, x, y)$$

Définition 7 *Soit $\mathcal{P} = (X, P)$ un ordre partiel, on appelle λ-épaisseur de $\mathcal{P}$, notée λ-epais$(\mathcal{P})$, le minimum des cardinaux t des familles d'ordres qui sont une λ-réalisation de $\mathcal{P}$.*

Restriction d'un prédicat $\lambda(\mathcal{F}, x, y)$

Pour $\mathcal{E} \subseteq \mathcal{O}$ un ensemble d'ordres partiels, on pose :

$D_{\mathcal{E}} = \{(\mathcal{F}, x, y) |$ il existe un ensemble fini X tel que $\mathcal{F} \in \mathcal{P}_f(\mathcal{E}(X))$ et $(x, y) \in X^2\}$

Pour un prédicat λ défini sur D, on note $\lambda(\mathcal{E})$ la restriction de λ à $D_{\mathcal{E}}$.

$\lambda(\mathcal{E})$-réalisation d'un ordre partiel

La notion de $\lambda(\mathcal{E})$-réalisation d'un ordre partiel est la même que celle de λ-réalisation à la seule différence que les ordres de la famille doivent appartenir à $\mathcal{E}$. Ce qui donne les définitions suivantes :

Définition 8 *Soit un prédicat défini sur D et soit $\mathcal{E} \subseteq \mathcal{O}$ un ensemble d'ordres partiels. On dit qu'une famille $\mathcal{F}$ d'ordres partiels de $\mathcal{E}$ sur X est une $\lambda(\mathcal{E})$-réalisation de $\mathcal{P} = (X, P)$ si et seulement si :*

$$\forall (x, y) \in X^2, x <_{\mathcal{P}} y \Longleftrightarrow \lambda(\mathcal{E})(\mathcal{F}, x, y)$$

Définition 9 *Soit $\mathcal{P} = (X, P)$ un ordre partiel, on appelle $\lambda(\mathcal{E})$-épaisseur de $\mathcal{P}$, notée $\lambda(\mathcal{E})$-epais$(\mathcal{P})$, le minimum des cardinaux t des familles d'ordres qui sont une $\lambda(\mathcal{E})$-réalisation de $\mathcal{P}$.*

propriété 2 *Pour $\mathcal{E} \subseteq \mathcal{E}'$, si $\mathcal{F}$ est une $\lambda(\mathcal{E})$-réalisation de $\mathcal{P}$ alors $\mathcal{F}$ est une $\lambda(\mathcal{E}')$-réalisation de $\mathcal{P}$, d'où $\lambda(\mathcal{E}')$-epais$(\mathcal{P}) \leq \lambda(\mathcal{E})$-epais$(\mathcal{P})$.*

On a $\lambda = \lambda(\mathcal{O})$. Dans la suite on précisera toujours l'ensemble d'ordres qu'on s'autorise dans la réalisation et on parlera donc uniquement de $\lambda(\mathcal{E})$-réalisation avec éventuellement $\mathcal{E} = \mathcal{O}$.

Exemple : le codage issu de la théorie de la dimension
Soit le prédicat i défini sur D par :

$$i(\mathcal{F}, x, y) = \forall \mathcal{P} \in \mathcal{F}, x <_{\mathcal{P}} y$$

Une $i(D_1)$-réalisation de $\mathcal{P}$ n'est autre qu'un réaliseur de $\mathcal{P}$ (voir section 1.3.2) et $i(D_1)$-epais$(\mathcal{P})$ est donc exactement $dim(\mathcal{P})$.

Pour d'autres exemples, le lecteur pourra se reporter dès maintenant à la section suivante dans laquelle nous montrons que la plupart des codages introduits dans le chapitre 1.2 s'inscrivent dans ce formalisme. Le seul n'étant pas dans ce cas est le codage simple de la section 1.3.1.

Relation de généralisation entre codages par $\lambda(\mathcal{E})$-réalisation

Nous sommes maintenant en mesure de donner une définition rigoureuse de la généralisation de codage pour des codages issus de $\lambda(\mathcal{E})$-réalisation.

Définition 10 *Soient un codage C_1 par $\lambda_1(\mathcal{E}_1)$-réalisation et un codage C_2 par $\lambda_2(\mathcal{E}_2)$-réalisation. On dit que C_2 est une généralisation de C_1 si et seulement si :*

$$\mathcal{E}_1 \subseteq \mathcal{E}_2$$

Pour X un ensemble fini quelconque,

$$\forall \mathcal{R} \in \mathcal{P}_f(\mathcal{E}_1(X)), \mathcal{R} \text{ est une } \lambda_1(\mathcal{E}_1)\text{-réalisation} \Longrightarrow \mathcal{R} \text{ est une } \lambda_2(\mathcal{E}_2)\text{-réalisation}$$

Nous conservons la même définition d'un codage plus expressif que celle donnée en 2.2.

Maintenant posées les notions de $\lambda(\mathcal{E})$-réalisation et de relation de généralisation entre les codages de ce type, nous pouvons commencer à chercher lesquels sont les plus expressifs. C'est le travail que nous commençons dans la section suivante.

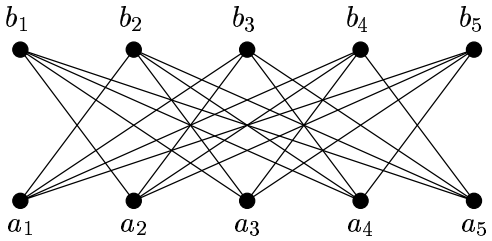

FIG. 3.1 – S_5

3.2 Construction de la hiérarchie

Dans cette partie, nous examinons dans le détail les relations de généralisation entre les codages introduits en 1.3 et d'autres que nous présentons ici. Nous cherchons à déterminer, en les comparant deux à deux, lesquels sont les plus expressifs. Tous ces codages sont des variations plus ou moins amples autour de la notion de dimension.

Codage par réaliseur

Le codage par réaliseur est celui issu de la théorie de la dimension présentée en 1.3.2. il entre dans le cadre des codages par $\lambda(\mathcal{E})$-réalisation. le prédicat utilisé est le prédicat i défini sur D par :

$$i(\mathcal{F}, x, y) = \forall \mathcal{P} \in \mathcal{F}, x <_{\mathcal{P}} y$$

les ordres utilisés dans les réalisations sont les ordres linéaires (ordres de dimension 1) notés D_1.

Le codage par réaliseur est donc, dans le cadre formel introduit, le codage par $i(D_1)$-réalisation et $i(D_1)$-$epais(\mathcal{P})$ est une notion parfaitement identique à $dim(\mathcal{P})$.

Le codage par réaliseur, issu de la théorie de la dimension constitue le point de départ de la hiérarchie que nous construisons (voir figure 3.6).

Codage par string realizer

Le codage par string realizer, introduit en 1.3.3, s'inscrit dans le formalisme de la $\lambda(\mathcal{E})$-réalisation.

On définit le prédicat c sur D par :

$$c(\mathcal{F}, x, y) = \exists \mathcal{P} \in \mathcal{F}, x <_{\mathcal{P}} y \text{ et } \nexists \mathcal{Q} \in \mathcal{F}, y <_{\mathcal{Q}} x$$

On rappelle qu'on note S l'ensemble des strings (voir section 1.3.3). Ainsi, les notions de string realizer et de $c(S)$-réalisation sont confondues, de même que celles de $c(S)$-épaisseur et de string dimension.

Nous allons maintenant montrer que le codage par $c(S)$-réalisation est plus général et plus expressif que celui par $i(D_1)$-réalisation.

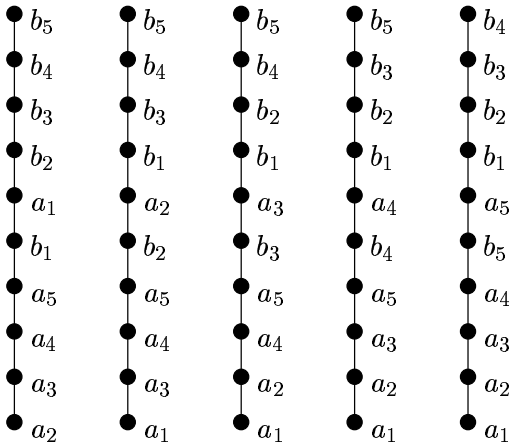

FIG. 3.2 – Un réaliseur de S_5

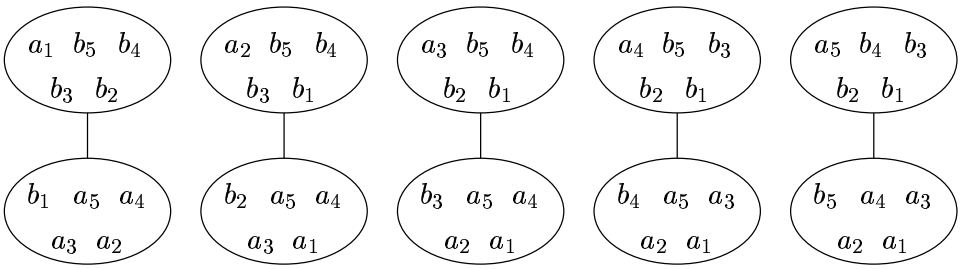

FIG. 3.3 – Un string realizer de S_5

Tout d'abord, remarquons que $D_1 \subset S$. Si $\mathcal{F}$ est une $i(D_1)$-réalisation de $\mathcal{P} = (X, P)$, alors tous les ordres $\mathcal{P}$ de $\mathcal{F}$ sont linéaires. c'est à dire :

$$\forall (x, y) \in X^2, \forall \mathcal{P} \in \mathcal{F}, x \neq y \Longrightarrow x <_{\mathcal{P}} y \text{ ou } y <_{\mathcal{P}} x$$

D'où

$$\forall (x, y) \in X^2, (\nexists \mathcal{P} \in \mathcal{F}, y <_{\mathcal{P}} x) \Longleftrightarrow (\forall \mathcal{P} \in \mathcal{F}, x <_{\mathcal{P}} y)$$

D'où, si $\mathcal{F}$est une $i(D_1)$-réalisation, alors c'est une $c(S)$-réalisation. Le codage par $c(S)$-réalisation est plus général que le codage par $i(D_1)$-réalisation.

Pour montrer qu'il est aussi plus expressif, nous introduisons la famille $(S_n)_{n\geq 1}$ définie ainsi (voir figure 3.1) :
pour $n \geq 1$, $S_n = (A_n, B_n, P_n)$ avec $A_n = \{a_k | 1 \leq k \leq n\}$ et $B_n = \{b_k | 1 \leq k \leq n\}$ et :

$$\forall k \in [\![1, n]\!], \quad \forall l \in [\![1, n]\!], l \neq k \Longrightarrow a_k < b_l \quad \text{et} \quad a_k || b_k$$

Or $dim(S_n) = n$, voir figure 3.2. Il faut donc seulement $n\,log\,2n$ bits par étiquette pour coder S_n avec le codage par $i(D_1)$-réalisation. or, il existe un string realizer de S_n en n srtings de hauteur 2 (voir [5] et figure 3.3). Il faut donc seulement $n\,log\,2 = n$ bits par étiquette pour coder S_n par $c(S)$-réalisation. Comme $\lim\limits_{n\to+\infty} \frac{n}{n\,log\,2n} = 0$, le codage par $c(D_1)$-réalisation est plus expressif que celui par $i(D_1)$-réalisation.

Codage par réalisation en ordres de dimension 2

On note D_2 l'ensemble des ordres de dimension 2. On utilise le prédicat c défini précédemment. Une réalisation en ordres de dimension 2 est une $c(D_2)$-réalisation et $epais_2(\mathcal{P})$ est exactement $c(D_2)$-$epais(\mathcal{P})$.

Montrons que le codage par $c(D_2)$-réalisation est plus général et plus expressif que le codage par $c(S)$-réalisation.

Une string est de dimension 2. Il est donc immédiat que toute $c(S)$-réalisation est une $c(D_2)$-réalisation (voir propriéte 2 page **??**).

Pour montrer que la $c(D_2)$-réalisation est plus expressive que la $c(S)$-réalisation, nous utilisons encore la famille $(S_n)_{n\geq 1}$. Comme nous l'avons déjà dit, $c(S)$-$epais(S_n) = n$, il faut donc au moins n bits par étiquette pour coder S_n par $c(S)$-réalisation(et il en suffit de n). Nous allons montrer que

$$\forall n \in \mathbb{N}^*, c(D_2)\text{-}epais(S_n) = 2$$

Pour cela nous exhibons une $c(D_2)$-réalisation de S_n à l'aide de 2 ordres de dimension 2 (voir figure 3.4).

Ces 2 ordres S_n^+ et S_n^- définis ainsi :
$S_n^+ = (A_n, B_n, P_n)$ avec $A_n = \{a_k | 1 \leq k \leq n\}$ et $B_n = \{b_k | 1 \leq k \leq n\}$ et :

$$\forall k \in [\![1, n]\!], \forall l \in [\![1, n]\!], a_k < b_l$$

$S_n^- = (B_n, A_n, P_n)$ avec $A_n = \{a_k | 1 \leq k \leq n\}$ et $B_n = \{b_k | 1 \leq k \leq n\}$ et :

$$\forall k \in [\![1, n]\!], b_k < a_k$$

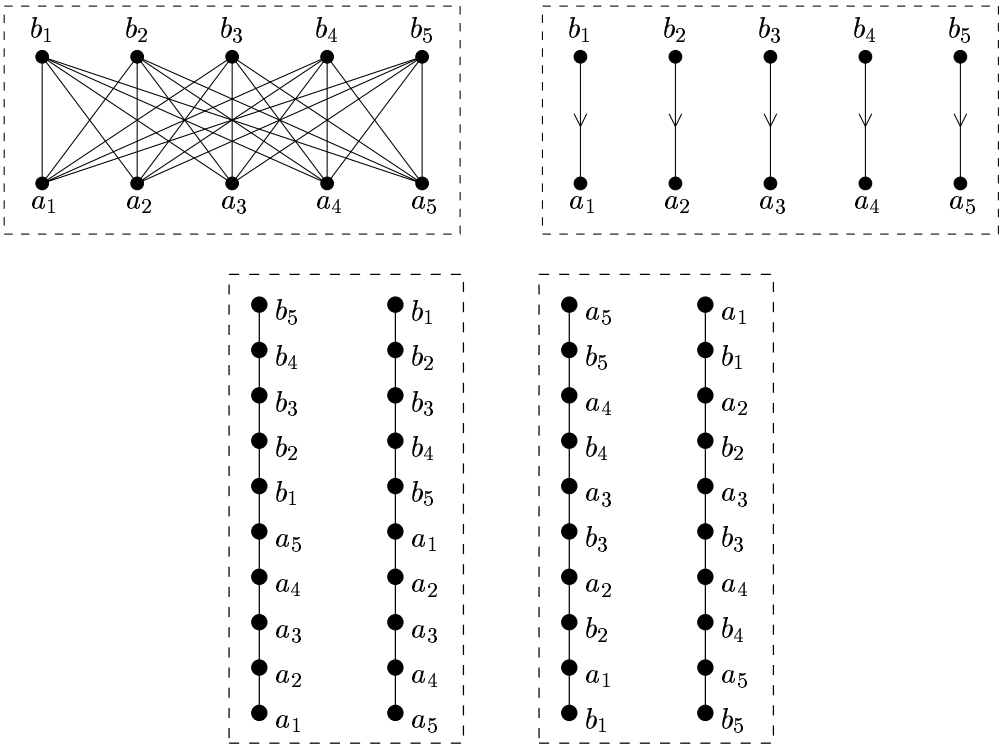

FIG. 3.4 – S_5 comme différence de 2 ordres de dimension 2 et la réalisation correspondante

S_n^+ est de dimension 2 car c'est une string. S_n^-est de dimension 2 car c'est l'union d'arêtes disjointes. En évidence, (S_n^+, S_n^-) est une $c(D_2)$-réalisation de S_n.

Comme $c(D_2)$-$epais(S_n) = 2$, il suffit de $4\,log\,n$ bits par étiquette pour coder S_n par $c(D_2)$-réalisation. Or $\lim\limits_{n\to+\infty} \frac{4\,log\,n}{n} = 0$, le codage par $c(D_2)$-réalisation est donc plus expressif que celui par $c(S)$-réalisation.

codage par réaliseur en ordres faibles et réaliseur en ordres chaine-faibles

Somme disjointe d'ordres partiels

Pour $\mathcal{P} = (X, P)$et $\mathcal{Q} = (Y, Q)$ deux ordres partiels sur des ensembles sous-jacents disjoints $(X \cap Y = \varnothing)$, on appelle somme disjointe $P + Q$ la relation d'ordre sur $X \sqcup Y$ définie par :

$$\forall (x, y) \in (X \sqcup Y)^2, x <_{P+Q} y \iff x <_P y \text{ ou } x <_Q y$$

Somme linéaire d'ordres partiels

Pour $\mathcal{P} = (X, P)$et $\mathcal{Q} = (Y, Q)$ deux ordres partiels sur des ensembles sous-jacents disjoints $(X \cap Y = \varnothing)$, on appelle somme linéaire $P \oplus Q$ la relation binaire obtenue à partir de $P + Q$ en ajoutant les nouvelles relations $x <_{P\oplus Q} y$ pour tous $(x, y) \in (X, Y)$.

Ordre faible

$\mathcal{P} = (X, P)$ est appellé un ordre faible s'il est la somme linéaire d'antichaînes ; c'est à dire $P = A_1 \oplus A_2 \oplus \cdots \oplus A_n$ avec pour $1 \leq i \leq n$, A_i est une antichaîne.

Remarque : les ordres faibles sont exactement les strings définies en 1.3.3.

Dimension faible

La dimension faible d'un ordre partiel $\mathcal{P} = (X, P)$, notée $wdim(\mathcal{P})$, est le nombre minimal d'ordres faibles dont l'intersection est $\mathcal{P}$.
Cette définition a bien un sens car une chaîne est un ordre faible.

Ordre chaîne-faible

$\mathcal{P} = (X, P)$ est appellé un ordre chaîne-faible s'il est la somme linéaire de sommes disjointes de chaînes ; c'est à dire $P = A_1 \oplus A_2 \oplus \cdots \oplus A_n$ avec pour $1 \leq i \leq n$, A_i est une somme disjointe de chaînes.

On note CW l'ensemble des ordres de $\mathcal{O}$ qui sont chaîne-faibles, par abus de langage on dira que CW est l'ensemble des ordres chaine-faibles.

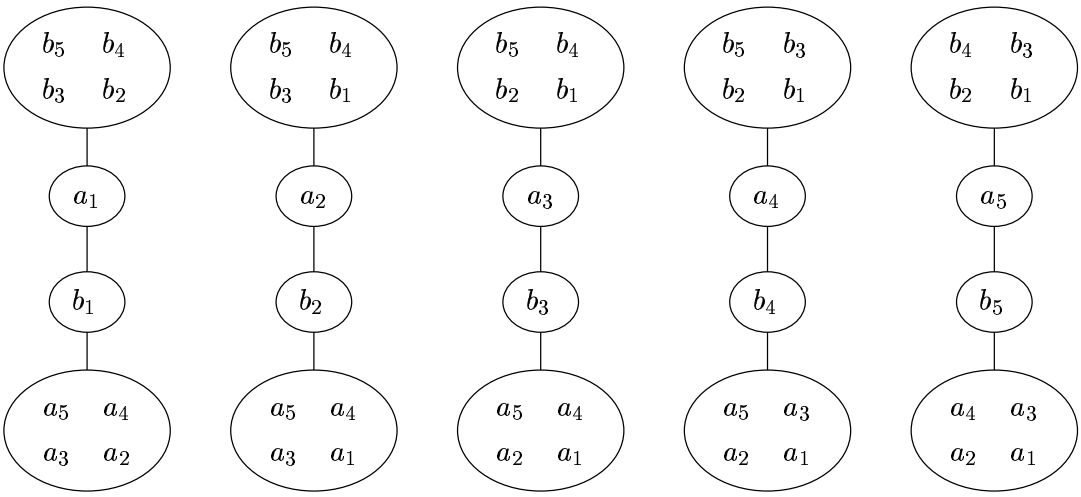

FIG. 3.5 – Un codage de S_5 par réaliseur en ordres faibles

Remarque : Un ordre faible est un ordre chaine-faible.

Dimension chaine-faible

La dimension chaîne-faible d'un ordre partiel $\mathcal{P} = (X, P)$, notée $cwdim(\mathcal{P})$, est le nombre minimal d'ordres chaîne-faibles dont l'intersection est $\mathcal{P}$.
Cette définition a bien un sens car une chaine est un ordre chaine-faible.

On utilise le prédicat i déjà défini. Un réaliseur en ordres faibles n'est autre qu'une $i(S)$-réalisation, et un réaliseur en ordres chaîne-faibles une $i(CW)$-réalisation. De même, $wdim(\mathcal{P})$ est $i(S)$-$epais(\mathcal{P})$ et $cwdim(\mathcal{P})$ est $i(CW)$-$epais(\mathcal{P})$.

Montrons d'abord que le codage par $i(S)$-réalisation est plus général et plus expressif que celui par $i(D_1)$-réalisation.

Comme $D_1 \subset S$ et comme les 2 codages reposent sur le même prédicat, il est évident qu'ils sont liés par généralisation. On montre grace à la famille $(S_n)_{n\geq 1}$ que la $i(S)$-réalisation est plus expressive. Pour cela on produit une $i(S)$-réalisation de S_n par n strings de hauteur 4 $(T_1, \ldots, T_n)$. Voir figure 3.5.

Pour $1 \leq i \leq n$, T_i est défini comme la somme linéaire des 4 antichaînes constituées des points :

- $A_n \setminus \{a_i\}$
- b_i
- a_i
- $B_n \setminus \{b_i\}$

Remarque : On voit bien dans le codage de S_n, la différnce entre le codage par string présenté en 1.3.3 et celui par les ordres faibles, elle repose sur le prédicat employé : c pour les strings et i pour les ordres faibles. Par exemple, le codage par string realizer de S_n proposé sur la figure 3.3 n'est pas un codage par réaliseur au sens des ordres faibles. En effet, dans la première string, on a

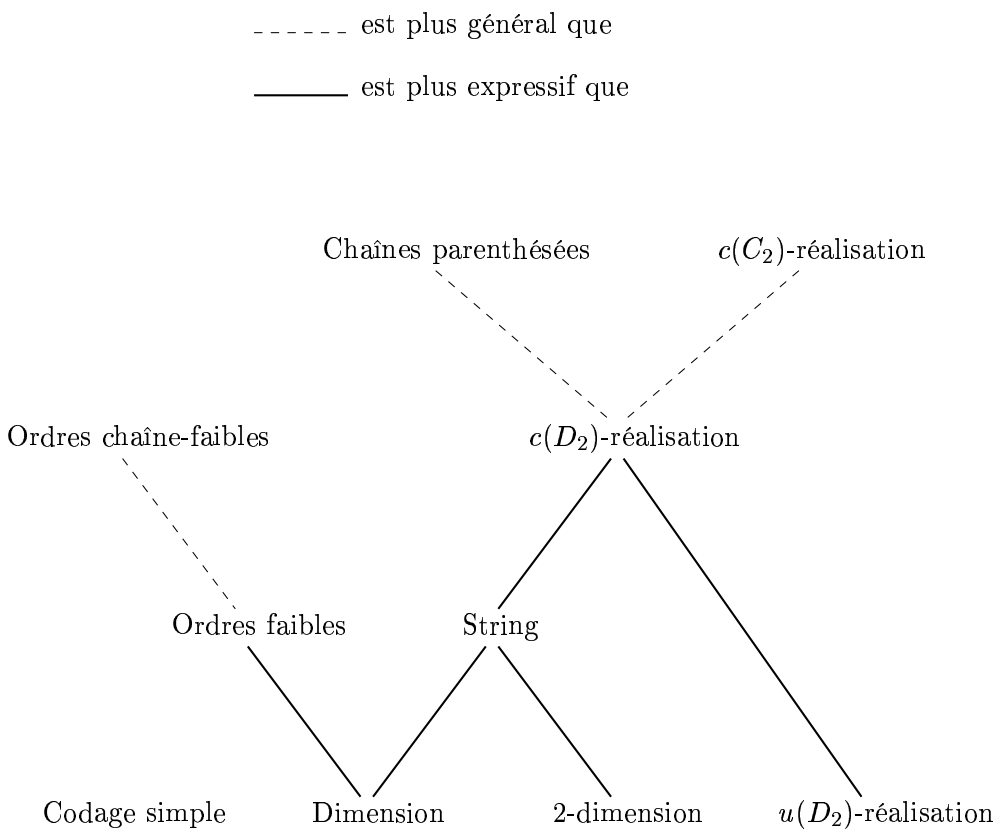

FIG. 3.6 – Relations de hiérarchisation de quelques codages

$a_1||b_5$ donc on ne peut pas avoir $a_1 < b_5$ dans l'intersection des strings du string realizer et ce n'est donc pas un réaliseur au sens des ordres faibles (c'est à dire avec le prédicat i).

Il suffit donc de $n\,log\,4 = 2n$ bits par étiquettes pour coder S_n par $i(S)$-réalisation, alors qu'il en faut $n\,log\,2n$ par $i(D_1)$-réalisation. $\lim_{n\to+\infty} \frac{2n}{n\,log\,2n} = 0$ donc le codage par $i(S)$-réalisation est plus expressif que celui par $i(D_1)$-réalisation.

Conclusion

Nous avons maintenant obtenu une hiérarchie de quelques codages existants. L'intérêt de cette hiérarchie est de synthétiser le travail effectué dans un certain nombre d'articles. Ces articles présentent des codages généralisés de la dimension et montrent qu'ils permettent de mieux coder certains ordres (en général la famille $(S_n)_{n\geq 1}$). Ce qu'on peut reprocher à ces articles, c'est de montrer le pouvoir de codage potentiel des codages proposés mais de ne pas en donner de moyen d'exploitation, c'est à dire d'algorithme d'encodage avec des garanties de performance intéressantes. La hiérarchie que nous avons construite et qui peut être étendue avec de nouveaux codages est un outil qui peut se révéler

utile pour chercher de nouveaux codages aux performances exploitables. En particulier, on peut se demander si les codages les plus pertinents sont ceux du haut de la hiérarchie. Les codages pour lesquels on dispose d'algorithmes de codage intéressants, en $O(n)$ bits par étiquette dans le pire des cas, sont dans le bas de la hiérarchie (comme le codage simple ou la 2-dimension). Le problème de ces codages est qu'ils codent peu efficacement des ordres que l'on sait coder beaucoup mieux par la théorie de la dimension. Ce qu'on souhaiterait, c'est pouvoir concilier les deux : pouvoir coder efficacement les ordres de faible dimension et avoir le pire des cas en $O(n)$ bits par étiquette. La classification comporte nombre de codages dont le formalisme regroupe les 2 avantages, tous ceux au dessus de la dimension et de la 2-dimension. Cependant, pour aucun on ne sait donner un algorithme polynomial qui soit en $O(log\, n)$ bits par étiquette sur les ordres de dimension bornée et en $O(n)$ dans le pire des cas général. Cela n'est pas vraiment étonnant puisque le problème qui consiste à décider si la dimension d'un ordre partiel est inférieure à k est NP-complet. Il ne paraît donc pas très raisonnable d'attendre ces performances de codages qui sont des généralisations de la dimension.

Par contre, on peut chercher d'autres codages qui donneraient des performances en $O(log\, n)$ sur de grandes familles d'ordres (qui ne soient pas ceux de dimension bornée) et qui garantiraient un codage en $O(n)$ bits par étiquette dans le pire des cas ; et pour lesquels on puisse trouver des algorithmes polynomiaux garantissant ces performances. C'est là une exigence très forte, peut-être la plus forte qu'on puisse avoir sur un codage de la comparaison dans un ordre. Sans prétendre atteindre ces objectifs, compte tenu de la discussion qui précède, le codage par $u(D_2)$-réalisation introduit dans le chapitre suivant semble digne d'intérêt.

Chapitre 4

Codage par intersection ou par union ?

Dans le codage issu de la théorie de la dimension, le prédicat utilisé est le prédicat i qui traduit l'intersection des ordres du réaliseur. Tous les ordres utilisés pour réaliser $\mathcal{P} = (X, P)$ sont des extensions linéaires de $\mathcal{P}$ et contiennent donc toutes les relations de P. On s'arrange pour que l'intersection de ces ordres linéaires ne contiennent pas les relations qui ne sont pas dans $\mathcal{P}$. Une famille d'ordres est un réaliseur de $\mathcal{P}$ si et seulement si tous les couples d'éléments incomparables x, y sont inversés, c'est à dire $x < y$ dans un ordre du réaliseur et $y < x$ dans un autre. P est alors exactement la partie commune à tous les ordres.

Nous allons, dans cette partie, nous intéresser à la démarche inverse : on va coder un ordre par une union d'ordres de dimension 2, c'est à dire qu'on va recouvrir, sans déborder, l'ordre $\mathcal{P}$ par des ordres de dimension 2. C'est la $u(D_2)$-réalisation que nous introduisons ci-après. Dans cette approche, si $x||y$ dans $\mathcal{P}$, alors ce sera le cas dans tous les ordres de la $u(D_2)$-réalisation, et si $x < y$ dans $\mathcal{P}$, alors ce sera le cas dans au moins un de ces ordres (et on n'aura jamais $y < x$).

En fait, nous avons déjà un codage s'approchant de cette définition : c'est celui par $c(D_2)$-réalisation. Le prédicat c utilise à la fois le fait que si $x < y$ dans un ordre de la $c(D_2)$-réalisation et $y < x$ dans un autre alors $x||y$, et le fait que si $x < y$ dans un ordre et $x||y$ dans tous les autres alors $x < y$ dans $\mathcal{P}$. La $c(D_2)$-réalisation permet à la fois de coder par intersection et par union. D'ailleurs, on peut remarquer que le codage de S_5 proposé en , utilise les 2 aspects (voir figure 3.2). Cependant nous aurions pu proposer un codage de S_5 (ou S_n) par union pure (voir figure 4.1). Peut on toujours coder aussi bien par union que par la combinaison de l'union et de la contradiction ? Autrement dit, la $u(D_2)$-réalisation est elle aussi expressive que la $c(D_2)$-réalisation ? Nous montrons dans ce qui suit que ce n'est pas le cas.

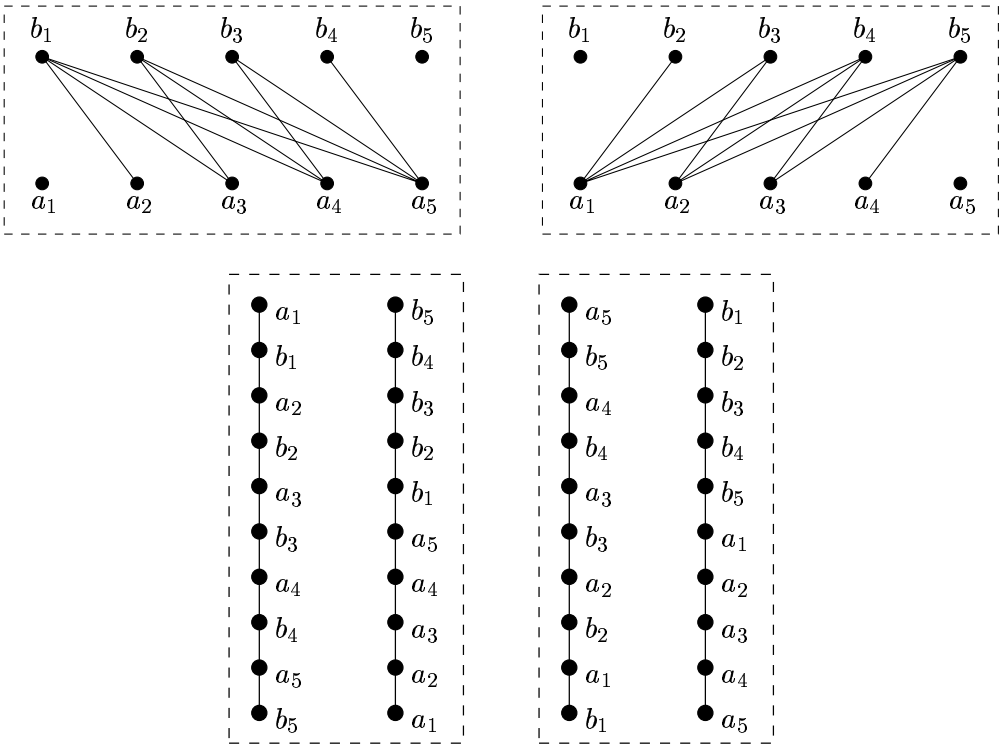

FIG. 4.1 – S_5 comme union de 2 ordres de dimension 2 et la réalisation correspondante

4.1 Préliminaires

$u(D_2)$-réalisation d'un ordre partiel

On défini le prédicat u par :

$$\forall(\mathcal{F}, x, y) \in D, u(\mathcal{F}, x, y) = \exists \mathcal{P} \in \mathcal{F}, x <_{\mathcal{P}} y$$

On rappelle la définition du prédicat c :

$$c(\mathcal{F}, x, y) = \exists \mathcal{P} \in \mathcal{F}, x <_{\mathcal{P}} y \text{ et } \nexists \mathcal{Q} \in \mathcal{F}, y <_{\mathcal{Q}} x$$

Le codage par $c(D_2)$-réalisation est une généralisation du codage par $u(D_2)$-réalisation. En effet :

- $D_2 \subseteq D_2$
- Soit $\mathcal{R}$ une $u(D_2)$-réalisation de $\mathcal{P}$, alors $x <_{\mathcal{P}} y \iff \exists \mathcal{P} \in \mathcal{R}, x <_{\mathcal{P}} y$. Soit $x <_{\mathcal{P}} y$, alors $y \not<_{\mathcal{P}} x$ d'où $\nexists \mathcal{Q} \in \mathcal{R}, y <_{\mathcal{Q}} x$. Or $\exists \mathcal{P} \in \mathcal{R}, x <_{\mathcal{P}} y$, d'où

  $$x <_{\mathcal{P}} y \Longrightarrow c(D_2)(\mathcal{R}, x, y)$$

  Réciproquement, si $c(D_2)(\mathcal{R}, x, y)$, alors en particulier $\exists \mathcal{P} \in \mathcal{R}, x <_{\mathcal{P}} y$ et on a donc $x <_{\mathcal{P}} y$. Finalement, $x <_{\mathcal{P}} y \iff c(D_2)(\mathcal{R}, x, y)$.
  $\mathcal{R}$ est donc une $c(D_2)$-réalisation de $\mathcal{P}$.

Nous allons montrer dans ce chapitre que le codage par $c(D_2)$-réalisation est plus expressif que celui par $u(D_2)$-réalisation.

Ordres bipartis et convexité

Définition 11 *Un ordre biparti $\mathcal{P} = (L, U, P)$ est convexe supérieur s'il existe un ordre total $\mathcal{T}$ sur les éléments de U tel que les successeurs dans $\mathcal{P}$ des éléments de L sont des intervalles dans $\mathcal{T}$.*

Définition 12 *Un ordre biparti $\mathcal{P} = (L, U, P)$ est convexe supérieur fermé s'il existe un ordre total $\mathcal{T}$ sur les éléments de U tel que les successeurs des éléments de L sont des intervalles dans $\mathcal{T}$ et tel que si 2 de ces intervalles sont inclus l'un dans l'autre alors ils ont une borne commune.*

On définit de même les convexes inférieurs et les convexes inférieurs fermés. On appelle biconvexe fermé un ordre biparti convexe supérieur fermé et convexe inférieur fermé.

Théorème 1 *Un ordre biparti est de dimension 2 si et seulement si c'est convexe supérieur fermé, il est alors biconvexe fermé.*

Comme la $c(D_2)$-réalisation est une généralisation de la $u(D_2)$-réalisation, $\forall \mathcal{P} \in \mathcal{O}, c(D_2)\text{-epais}(\mathcal{P}) \leq u(D_2)\text{-epais}(\mathcal{P})$. On peut se demander si : $\forall \mathcal{P} \in \mathcal{O}, c(D_2)\text{-epais}(\mathcal{P}) = u(D_2)\text{-epais}(\mathcal{P})$.

On va montrer que ce n'est pas le cas en construisant un ordre biparti B_3 tel que $c(D_2)\text{-epais}(B_3) = 2$ et $u(D_2)\text{-epais}(B_3) = 3$.

Théorème 2 $\exists \mathcal{P} \in \mathcal{O}, c(D_2)\text{-}epais(\mathcal{P}) < u(D_2)\text{-}epais(\mathcal{P})$

Nous donnons la preuve de ce théorème dans la section suivante.

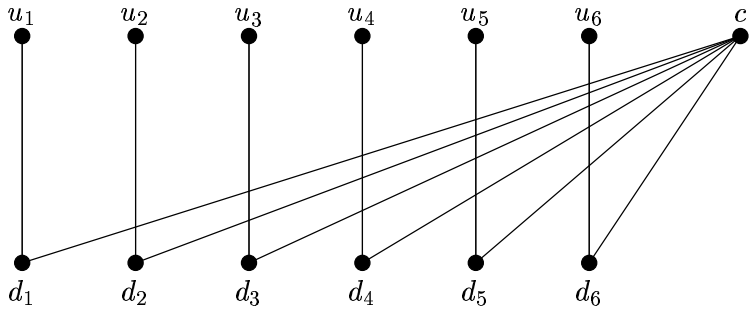

FIG. 4.2 – L_6

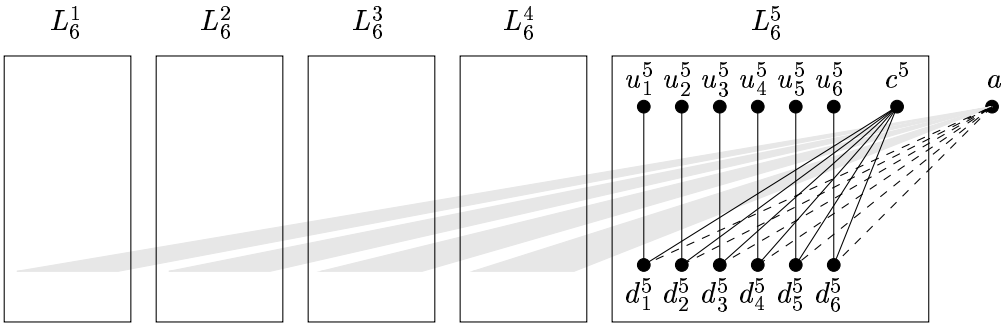

FIG. 4.3 – B_3

4.2 Le biparti B_3

Construction de B_3

On introduit pour commencer le graphe L_6 défini sur la figure 4.2.

Proposition 1 $\forall (i,j,k) \in [\![1,6]\!]^3$, $L_6[\{c, d_i, d_j, d_k, u_i, u_j, u_k\}]$ *n'est pas convexe supérieur.*

On construit B_3 avec 5 exemplaires de L_6 et un autre point a comme décrit sur la figure 4.3.

$u(D_2)$-$epais(B_3) = 3$

B_3 n'est pas l'union de 2 ordres de dimension 2

On note $B_3 = (D_3, U_3 \cup \{a\}, P_3)$

Preuve :

Nous allons le démontrer par l'absurde.

Supposons donc qu'il existe une $u(D_2)$-réalisation de B_3 en 2 ordres de dimension 2, $R1 = (D_3 \cup U_3 \cup \{a\}, P_1)$ et $R2 = (D_3 \cup U_3 \cup \{a\}, P_2)$. Commençons par remarquer que comme $P_3 = P_1 \cup P_2$, nécessairement R_1 et R_2 sont bipartis avec la même partition que B_3. Alors, dans un ordre $R_i = R_1$ ou R_2, un point $d \in D_3$ peut être de 3 types :

1. Type 1 : $d <_{R_i} a$ et $\exists u \in U, d <_{R_i} u$
   Dans l'ordre total $\mathcal{T}$ réalisant la convexité de l'ordre R_i, a a au plus 1 prédécesseur $u_p \in L_6^p$ et 1 successeur $u_s \in L_6^s$. Si d est de type 1, comme $d < u$ et $d < a$ et comme les successeurs de d dans $\mathcal{P}$ forment un intervalle dans $\mathcal{T}$ alors $d < u_p$ ou $d < u_s$ et par conséquent $d \in L_6^p$ ou $d \in L_6^s$. Donc les points de type 1 dans R_i sont répartis dans au plus 2 exemplaires de L_6 différents.
2. Type 2 : $d \not<_{R_i} a$
3. Type 3
   $d <_{R_i} a$ et $\nexists u \in U, d <_{R_i} u$

Comme dans un R_i il y a au plus 2 exemplaires de L_6 contenant des points de type 1, il y a au plus 4 exemplaires de R_i contenant des points de type 1 dans au moins un des ordres R_1 et R_2. Il existe donc au moins un exemplaire L_6^j de L_6 sur les 5 exemplaires que compte B_3 qui ne possède que des points de type 2 ou 3 dans chacun des 2 ordres R_1 et R_2. Il y a un R_i qui possède au moins 3 points $\{d_x^j, d_y^j, d_z^j\}$ de $D \cap L_6^j$ de type 3 (L_6 a 6 points sources). On peut supposer en renomant les ordres que cet ordre R_i est R_1. Si un point d est de type 3 dans l'ordre R_1, alors dans l'autre ordre R_2, on a $\forall u \in U_3, d <_{B_3} u \Longrightarrow d <_{R_2} u$. Ce qui signifie que $R_2[\{c^j, d_x^j, d_y^j, d_z^k, u_x^j, u_y^j, u_z^j\}] = L_6^j[\{c^j, d_x^j, d_y^j, d_z^k, u_x^j, u_y^j, u_z^j\}]$. Or cette dernière restriction n'est pas de dimension 2, elle ne peut pas être une restriction de R_2 qui est de dimension 2 (voir propriété 1, page 10). **Absurde**

B_3 est l'union de 3 ordres de dimension 2

On donne une décomposition de B_3 en 3 ordres de dimension 2 :

Ordre 1 : la restriction de B_3 à a et ses prédecesseurs, figure 4.4.

Ordre 2 : la restriction de B_3 aux $(c^j)_{1 \leq j \leq 5}$ et leurs prédecesseurs, figure 4.5.

Ordre 3 : la restriction de B_3 aux $(u_i^j)_{1 \leq i \leq 6 \ et \ 1 \leq j \leq 5}$ et leurs prédecesseurs, figure 4.6.

Nous laissons au lecteur le soin de vérifier que ces ordres sont bien de dimension 2 (ils le sont parce que ce sont des forêts dont les arbres sont orientés vers la racine).

$c(D_2)$-$epais(B_3) = 2$

On peut montrer que B_3 est la différence au sens des arêtes de 2 ordres de dimension 2. En fait, il existe 2 ordres $S_3 = (D_3, U_3 \cup \{a\}, P_{S_3})$ et $T_3 = (D_3, U_3 \cup \{a\}, P_{T_3})$ de dimension 2 tels que $P_{S_3} \cap P_3 = \varnothing$ et $P_{S_3} \sqcup P_3 = P_{T_3}$. On cherche donc à compléter B_3 en lui ajoutant des arêtes pour en faire un ordre de dimension 2.

Dans chaque copie de L_6 on complète comme le montre la figure 4.7 dans laquelle les relations en pointillés sont celles que l'on ajoute.

De plus, on relie chaque point source d'une copie de L_6 à tous les points puits des copies de L_6 suivantes (voir figure 4.9).

Le graphe complété est bien celui d'un ordre de dimension 2 puisque c'est un biparti convexe fermé : tous les points sources ont a comme borne de leur intervalle dans l'ordre de convexité (voir figure 4.10).

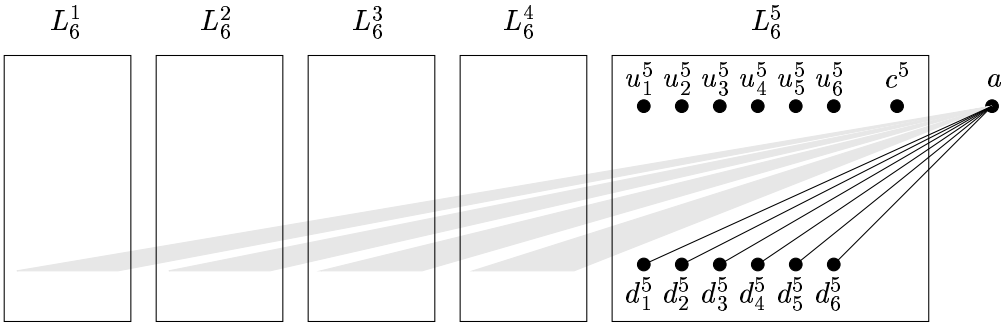

FIG. 4.4 – B_3 restreint aux relations entre a et ses prédécesseurs

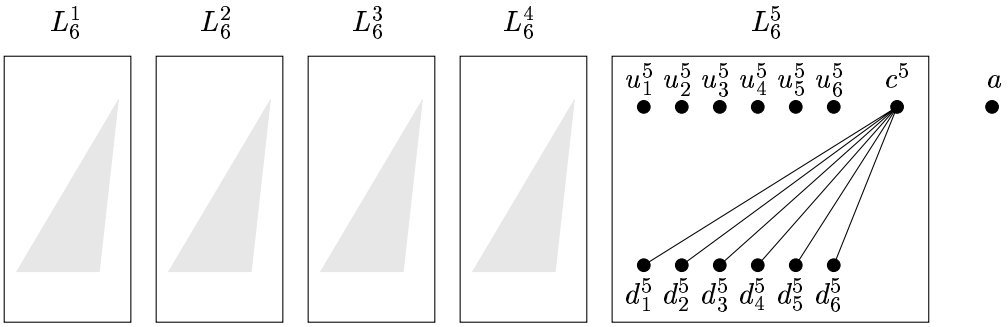

FIG. 4.5 – B_3 restreint aux relations entre les c^j et leurs prédécesseurs

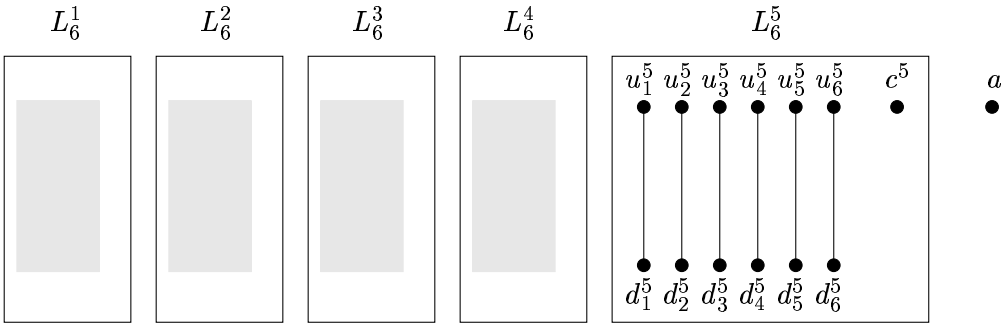

FIG. 4.6 – B_3 restreint aux relartions entre les u_i^j et leurs prédécesseurs

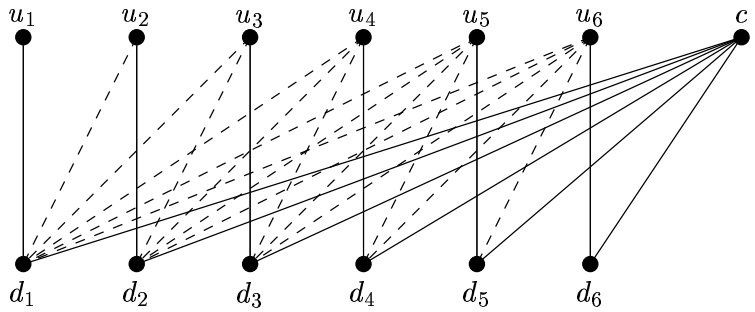

FIG. 4.7 – L_6 complété en un convexe supérieur fermé

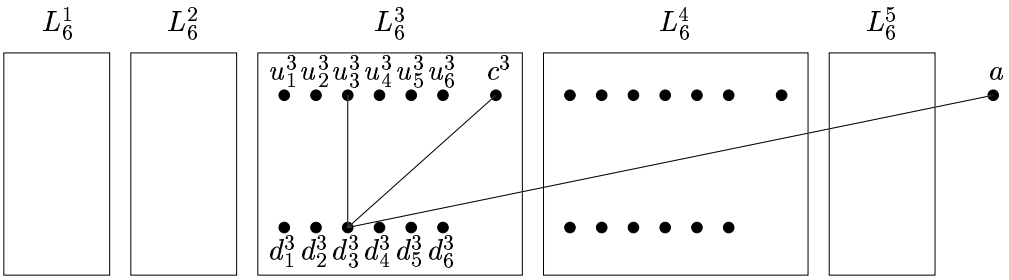

FIG. 4.8 – Les relations concernant d_3^3 dans B_3

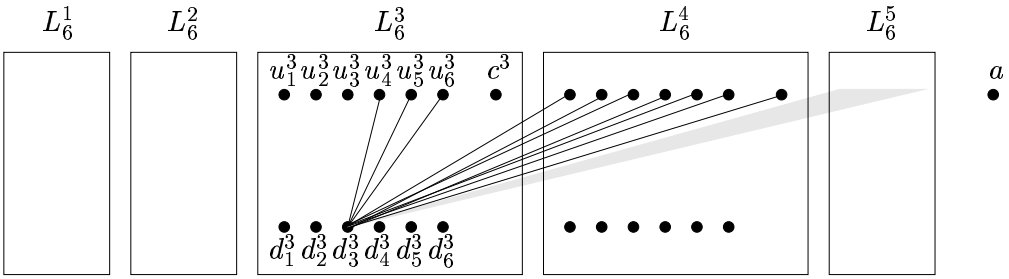

FIG. 4.9 – Les relations concernant d_3^3 dans S_3

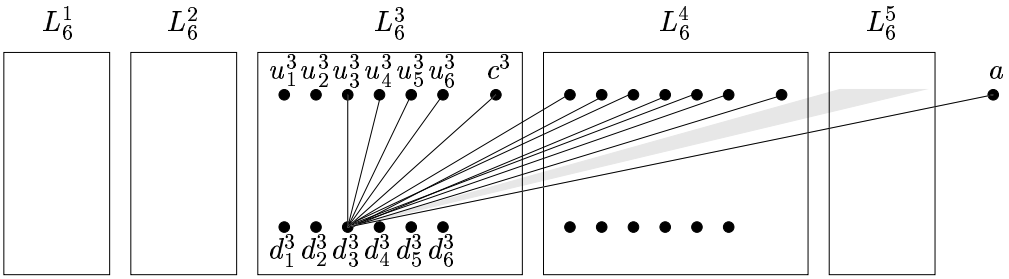

FIG. 4.10 – Les relations concernant d_3^3 dans T_3

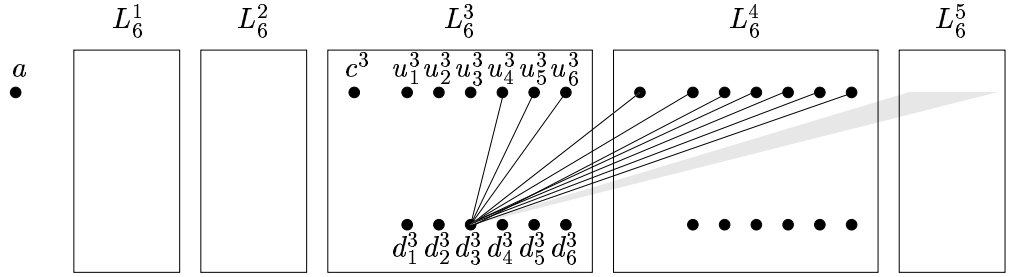

FIG. 4.11 – Un ordre de convexité pour S_3

Montrons que S_3 est de dimension 2.

S_3 est convexe avec l'ordre de convexité suivant (voir figure 4.11) :

$$a < c^1 < u_1^1 < u_2^1 < \cdots < u_6^1 < c^2 < u_1^2 < \cdots < u_6^2 < c^3 < \cdots < c^5 < u_1^5 < \cdots < u_6^5$$

S_3 est bien fermé puisque tous les points du bas ont u_6^5 comme borne de leur intervalle dans l'ordre de convexité (voir figure 4.11).

On peut donc réaliser B_3 en deux rectangles : T_3 et S_3^- où S_3^- est S_3 orienté dans le sens opposé. C'est à dire $S_3^- = (U_3 \cup \{a\}, D_3, P_{S_3}^-)$ avec

$$\forall (x,y) \in (D_3 \cup U_3 \cup \{a\})^2, (x,y) \in P_{S_3}^- \iff (y,x) \in P_{S_3}$$

On a donc montré qu'il est parfois plus efficace d'utiliser le prédicat c que le prédicat u dans les réalisations utilisant les ordres de dimension 2.

4.3 La famille $(B_n)_{n\geq 2}$

On va maintenant montrer :

Théorème 3 $\forall k \in \mathbb{N}, \exists \mathcal{P} \in \mathcal{O}, u(D_2)\text{-}epais(\mathcal{P}) \geq k\, c(D_2)\text{-}epais(\mathcal{P})$

Pour cela, on va construire une famille $(B_n)_{n\in\mathbb{N}}$ de bipartis telle que :
$\forall n \in \mathbb{N}, c(D_2)\text{-}epais(B_n) = 2$ et $u(D_2)\text{-}epais(B_n) = n$

Opération de composition sur les L_n

On introduit la famille de bipartis $(L_n)_{n\in\mathbb{N}^*}$, définie comme sur la figure 4.12.

Le point c est appelé point critique de L_n.

On définit l'opération suivante (notée multiplicativement) :

$$\begin{array}{ccc} \mathcal{L} \times \mathcal{B} & \to & \mathcal{B} \\ (L_n, B) & & L_n\, B \end{array}$$

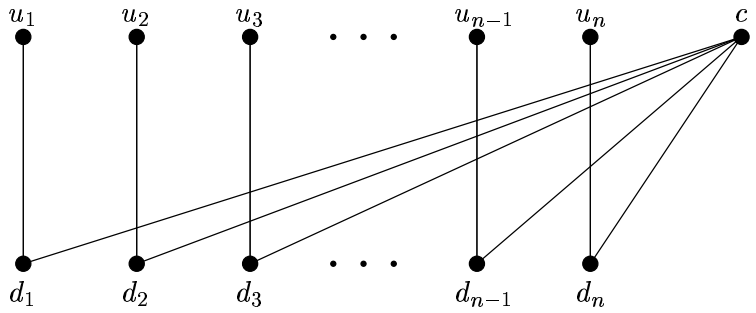

FIG. 4.12 – L_n

où $L_n\,B$ est défini comme suit :

$L_n = (D_n, U_n \cup \{c\}, Q_n)$
avec $D_n = \{d_i | 1 \leq i \leq n\}$ et $U_n = \{u_i | 1 \leq i \leq n\}$

$B = (G, H, Q)$
avec $G = \{g_i | 1 \leq i \leq p\}$ et $H = \{h_i | 1 \leq i \leq q\}$

$L_n\,B = \mathcal{P} = (X, Y \cup \{c\}, P)$
avec $X = \{x_{ij} | 1 \leq i \leq n \text{ et } \ 1 \leq j \leq p\}$ et $Y = \{y_{ij} | 1 \leq i \leq n \text{ et } \ 1 \leq j \leq q\}$
et P est défini ainsi :

- $\forall x \in X, (x, c) \in P$
- $(x_{ij}, y_{kl}) \in P$ si et seulement si $\ i = k$ et $(g_j, h_l) \in Q$

Cette opération ressemble beaucoup au produit cartésien d'ensemble partiellement ordonnés, c'est en fait une restriction du produit cartésien à un sous ensemble de points, celui qui est pertinent dans la preuve qui suit.

ATTENTION :
On utilise la notation $\prod_{i=1}^{n} L_{a_i}$ mais dans le sens inverse au sens habituel :

$$\prod_{i=1}^{n} L_{a_i} = L_{a_n}\ L_{a_{n-1}}\ \cdots\ L_{a_1}$$

Construction de la famille $(B_n)_{n \in \mathbb{N} \setminus \{0,1\}}$

On pose $\forall n \in \mathbb{N} \setminus \{0,1\}, B_n = \prod_{i=1}^{n-1} L_{a_i^n}$
Les a_i^n sont définis par récurrence de la manière suivante :

$a_1^2 = 3$

$$\forall n \in \mathbb{N} \setminus \{0,1,2\}, \begin{cases} a_1^n = (n-1)a_1^{n-1} \\ \forall i \in [\![2, n-1]\!], a_i^n = a_i^{n-1} \prod_{j=1}^{i-1} a_j^n \\ a_{n-1}^n = 2n-1 \end{cases}$$

$u(D_2)$- $epais(B_n)$

Théorème 4 $\forall n \in \mathbb{N} \setminus \{0,1\}, u(D_2)$-$epais(B_n) = n$

B_n n'est pas l'union de $n-1$ ordres de dimension 2

Lemme 1 *Soit* $n \in \mathbb{N} \setminus \{0,1,2\}$
On pose $B'_n = \prod_{i=1}^{n-2} L_{a_i^n} = (D', H', P')$.
Soit $\hat{D} \subseteq D'$ *avec* $|\hat{D}| \geq \frac{|D'|}{n-1}$.

Alors, $B'_n[\hat{D} \cup H']$ *contient une restriction à* B_{n-1}.

Preuve :
Pour $p \in [\![1, n-2]\!]$ on pose :

$$\begin{aligned} \mathcal{P}(p) \quad = \quad & '' \text{ En notant } \ A_p = \prod_{i=1}^{p} L_{a_i^n} = (D_p, H_p, P_p) \text{ avec} \\ & H_p = U_p \cup \{c_p\} \text{ où } \ c_p \text{ est le point critique de } \ L_{a_p^n}, \\ & \hat{D}_p \subseteq D_p \text{ et } \ |\hat{D}_p| \geq \tfrac{|D_p|}{n-1} \Longrightarrow A_p[\hat{D}_p \cup H_p] \text{ contient une} \\ & \text{restriction à } \prod_{i=1}^{p} L_{a_i^{n-1}} \ '' \end{aligned}$$

Nous allons montrer par récurrence que : $\forall p \in [\![1, n-2]\!]$, $\mathcal{P}(p)$ est vrai.

- $A_1 = L_{a_1^n} = L_{(n-1)a_1^{n-1}}$. Si $\hat{D}_1 \subseteq D_1$ et $|\hat{D}_1| \geq \frac{|D_1|}{n-1}$, alors $|\hat{D}_1| \geq a_1^{n-1}$ et en évidence $L_{a_1^n}[\hat{D}_1 \cup H_1]$ contient une restriction à $L_{a_1^{n-1}}$.
  $\mathcal{P}(1)$ est vraie.
- Supposons que $\mathcal{P}(q)$ est vraie pour $q \in [\![1, n-3]\!]$.

  Soit $\hat{D}_{q+1} \subseteq D_{q+1}$ avec $|\hat{D}_{q+1}| \geq \frac{|D_{q+1}|}{n-1}$.
  $A_{q+1} = L_{a_{q+1}^n} A_q = L_{a_{q+1}^{n-1} \prod_{j=1}^{q} a_j^n} A_q$

  On a $D_{q+1} = \{x_{kl} | 1 \leq k \leq a_{q+1}^n \text{ et } \ 1 \leq l \leq |D_q|\}$
  et $U_{q+1} = \{y_{kl} | 1 \leq k \leq a_{q+1}^n \text{ et } \ 1 \leq l \leq |D_q|\}$.
  Pour $1 \leq j \leq a_{q+1}^n$ on note $X_j = \{x_{jl} \in D_{q+1} | 1 \leq l \leq |D_q|\}$ et $Y_j = \{y_{jl} \in U_{q+1} | 1 \leq l \leq |D_q|\}$.

  Remarquons que $\forall j \in [\![1, a_{q+1}^n]\!], |X_j| = |D_q| = \prod_{j=1}^{q} a_j^n$ et que $|D_{q+1}| = a_{q+1}^n |D_q|$

  On pose $J = \left\{ j \in [\![1, a_{q+1}^n]\!] \mid |X_j \cap \hat{D}_{q+1}| \geq \frac{|D_q|}{n-1} \right\}$

Montrons $|J| \geq a_{q+1}^{n-1}$:

Si $i \notin J$, $|X_i \cap \hat{D}_{q+1}| < \frac{|D_q|}{n-1}$. Comme $\frac{|D_q|}{n-1}$ est entier (car $|D_q| = \prod_{j=1}^{q} a_j^n$ et $a_1^n = (n-1)a_1^n - 1$), $|X_i \cap \hat{D}_{q+1}| \leq \frac{|D_q|}{n-1} - 1$. D'où :

$$|\hat{D}_{q+1}| \leq |D_q|\,|J| + \left(\frac{|D_q|}{n-1} - 1\right)\left(a_{q+1}^n - |J|\right)$$

Or $|\hat{D}_{q+1}| \geq \frac{|D_{q+1}|}{n-1}$, d'où

$$\frac{|D_{q+1}|}{n-1} \leq |J|\,\left(|D_q| - \frac{|D_q|}{n-1} + 1\right) + a_{q+1}^n\,\left(\frac{|D_q|}{n-1} - 1\right)$$

or $|D_{q+1}| = a_{q+1}^n\,|D_q|$, d'où

$$a_{q+1}^n\,\left(\frac{|D_q|}{n-1} - \frac{|D_q|}{n-1} + 1\right) \leq |J|\,\left(|D_q| - \left(\frac{|D_q|}{n-1} - 1\right)\right)$$

De plus, $\forall q \in [\![1, n-3]\!]$, $\frac{|D_q|}{n-1} \geq 1$, d'où $0 < \left(|D_q| - \left(\frac{|D_q|}{n-1} - 1\right)\right) \leq |D_q|$. Comme $a_{q+1}^n = a_{q+1}^{n-1}\,|D_q|$, on obtient :

$$a_{q+1}^{n-1}\,|D_q| \leq |J|\,|D_q|$$

Et finalement,

$$|J| \geq a_{q+1}^{n-1}$$

Soit $J' \subseteq J$ tel que $|J'| = a_{q+1}^{n-1}$.

Pour $j \in J'$, $A_{q+1}[X_j \cup Y_j] \cong A_q$, de plus $|X_j \cap \hat{D}_{q+1}| \geq \frac{|D_q|}{n-1}$, donc, par hypothèse de récurrence, $A_{q+1}[(X_j \cap \widehat{D}_{q+1}) \cup Y_j]$ contient une restriction à $\prod_{i=1}^{q} L_{a_i^{n-1}}$. On note $A_{q+1}[V_j \cup W_j]$ cette restriction, avec $V_j \subseteq X_j \cap \widehat{D}_{q+1}$ et $W_j \subseteq Y_j$. Avec ces notations :

$$A_{q+1}[\bigcup_{j\in J'} V_j \cup \bigcup_{j\in J'} W_j \cup \{c_q + 1\}] \cong L_{a_{q+1}^{n-1}} \prod_{i=1}^{q} L_{a_i^{n-1}} = \prod_{i=1}^{q+1} L_{a_i^{n-1}}$$

Comme $\bigcup_{j\in J'} V_j \subseteq \widehat{D}_{q+1}$ et $\bigcup_{j\in J'} W_j \cup \{c_{q+1}\} \subseteq H_{q+1}$, cette restriction est bien une restriction de $A_{q+1}[\widehat{D}_{q+1} \cup H_{q+1}]$. $\mathcal{P}(q+1)$ est vraie.

– On vient donc de montrer qu'en notant $A_p = \prod_{i=1}^{p} L_{a_i^n} = (D_p, H_p, P_p)$:

$\forall p \in [\![1, n-2]\!], \widehat{D}_p \subseteq D_p$ et $|\hat{D}_p| \geq \frac{|D_p|}{n-1} \Longrightarrow A_p[\widehat{D}_p \cup H_p]$ contient une restriction à $\prod_{i=1}^{p} L_{a_i^{n-1}}$

En particulier, pour $p = n-2$, comme $\prod_{i=1}^{n-2} L_{a_i^{n-1}} = B_{n-1}$ ceci montre le lemme 1.

Théorème 5 $\forall n \in \mathbb{N} \setminus \{0,1\}, u(D_2)\text{-}epais(B_n) \geq n$

Preuve :
Nous allons le montrer par récurrence sur n :

- Pour n=2, puisque $B_2 = L_3$ n'est pas convexe supérieur, il n'est donc pas de dimension 2 et $u(D_2)\text{-}epais(B_2) \geq 2$.
- Supposons que pour $n \in \mathbb{N} \setminus \{0,1,2\}$, $u(D_2)\text{-}epais(B_{n-1}) \geq n-1$ et montrons qu'alors $u(D_2)\text{-}epais(B_n) \geq n$.
  Nous allons le montrer par l'absurde :
  Supposons que B_n soit l'union de p ordres de dimension 2 avec $p \leq n-1$.
  Si c'est le cas, B_n est aussi l'union de $n-1$ ordres de dimension 2 : on ne considère que le cas $p = n-1$.

  On conserve les notations précédentes :
  $B_n = L_{2n-1}\, B'_n$
  $B_n = (D, H, P)$ avec $H = U \cup \{c\}$ où c est le point critique de L_{2n-1}.

  On a $D = \{x_{kl} | 1 \leq k \leq 2n-1 \text{ et } 1 \leq l \leq |B'_n|\}$
  et $U = \{y_{kl} | 1 \leq k \leq 2n-1 \text{ et } 1 \leq l \leq |B'_n|\}$.
  Pour $1 \leq j \leq 2n-1$ on note $X_j = \{x_{jl} \in D | 1 \leq l \leq |B'_n|\}$ et $Y_j = \{y_{jl} \in U | 1 \leq l \leq |B'_n|\}$.

  On note $(R_i)_{1 \leq i \leq n-1}$ une famille d'ordres de dimension 2 dont l'union est B_n. Remarquons que, nécessairement, $\forall i \in [\![1, n-1]\!], R_i \in \mathcal{OB}(D, H)$.

  Pour X un ensemble fini, Soit $\mathcal{F} \subseteq \mathcal{P}_f(\mathcal{O}(X))$, pour $\mathcal{P} \in \mathcal{F}$, on note $\mathcal{P} = (X, P_{\mathcal{P}})$. On définit $\bigcup_{\mathcal{P} \in \mathcal{F}} \mathcal{P} = (X, \bigcup_{\mathcal{P} \in \mathcal{F}} P_{\mathcal{P}})$.
  propriété 3 *Pour* $X' \subseteq X, (\bigcup_{\mathcal{P} \in \mathcal{F}} \mathcal{P})[X'] = \bigcup_{\mathcal{P} \in \mathcal{F}} \mathcal{P}[X']$
  Dans un ordre R_i de la famille, un point de D peut être de 3 sortes :

  1. type 1 $d <_{R_i} c$ et $\exists u \in U, d <_{R_i} u$

     Dans l'ordre total $\mathcal{T}$ réalisant la convexité de R_i, c a au plus 1 prédécesseur $u_p \in Y_p$ et 1 successeur $u_s \in Y_s$.
     Si d est de type 1, comme $d <_{R_i} u$ et $d <_{R_i} c$ et comme les successeurs de d dans R_i forment un intervalle dans $\mathcal{T}$ alors $d < u_p$ ou $d < u_s$ et par conséquent $d \in Y_p$ ou $d \in Y_s$.

Donc les points de type 1 dans R_i sont répartis dans au plus 2 X_j différents.

2. type 2 $d \not<_{R_i} c$
3. type 3 $d <_{R_i} c$ et $\nexists u \in U, d <_{R_i} u$

Comme dans un ordres de la famille $(R_i)_{1\leq i\leq n-1}$ il y a au plus 2 X_j contenant des points de type 1, alors il y a au plus $2n-2$ X_j contenant des points de type 1 dans au moins un des ordres de la famille.
D'où,

$$\exists r \in [\![1, 2n-1]\!], X_r \text{ contient uniquement des points de type 2 ou 3}$$

Pour $i \in [\![1, n-1]\!]$, on note $T_i = \{b \in X_r | d \text{ est de type 3 dans } R_i\}$
Soit $b \in X_r$, comme on doit coder $b <_{B_n} c$, il existe au moins un rectangle de la famille dans lequel b est de type 3.
Donc

$$\bigcup_{1\leq i\leq n-1} T_i = X_r$$

Et par conséquent,

$$\exists h \in [\![1, n-1]\!], |T_h| \geq \frac{|X_r|}{n-1} = \frac{|B'_n|}{n-1}$$

$B_n[X_r \cup Y_r] \cong B'_n$. Or $T_h \subseteq X_r$ et $|T_h| \geq \frac{|B'_n|}{n-1}$, d'où d'après le lemme 1, $B_n[T_h \cup Y_r]$ contient une restriction à B_{n-1}. Or, comme les points de T_h sont de type 3 dans R_h, $R_h[T_h \cup Y_r] = (T_r, Y_r, \varnothing)$. De plus,comme $B_n = \bigcup_{1\leq i\leq n-1} R_i$ et en utilisant la propriété 3, on obtient :

$$B_n[T_h \cup Y_r] = \bigcup_{1\leq i\leq n-1} R_i[T_h \cup Y_r]$$

D'où

$$B_n[T_3^h \cup Y_r] = \bigcup_{\substack{1\leq i\leq n-1 \\ i\neq h}} R_i[T_h \cup Y_r]$$

Finalement, $B_n[T_3^h \cup Y_r]$ est l'union de $n-2$ ordres de dimension 2 et contient une restriction à B_{n-1}. D'après la propriété 3, B_{n-1} est l'union de $n-2$ ordres de dimension 2. **Absurde**
B_n n'est pas l'union de $n-1$ ordres de dimension 2. $u(D_2\text{-}epais(B_n) \geq n$.
$\mathcal{P}(n)$ est vraie.

Ce qui achève la récurrence et la preuve du théorème 5.

B_n est l'union de n ordres de dimension 2

Pour $p \in [\![1, n-1]\!]$ on pose :

$$\mathcal{P}(p) \quad = \quad \text{“ } A_p = \prod_{i=1}^{p} L_{a_i^n} = (D_p, H_p, P_p) \text{ est l'union de } p+1 \text{ ordres de dimension 2 ”}$$

On notera encore $H_p = U_p \cup \{c_p\}$ où c_p est le point critique de $L_{a_p^n}$.

- $L_{a_1^n} = (D_1, U_1 \cup \{c_1\}, P)$ est l'union des 2 ordres :
  $R_1 = A[D \times U]$
  $R_2 = A[D \times \{c\}]$
  qui sont de dimension 2. $\mathcal{P}(1)$ est vraie.
- Supposons pour $1 \leq q \leq n-2$, que $\mathcal{P}(q)$ est vrai.
  $A_{q+1} = L_{a_{q+1}^n} A_q$

  On introduit les mêmes notations que précédemment :
  On a $D_{q+1} = \{x_{kl} | 1 \leq k \leq a_{q+1}^n \text{ et } 1 \leq l \leq |D_q|\}$
  et $U_{q+1} = \{y_{kl} | 1 \leq k \leq a_{q+1}^n \text{ et } 1 \leq l \leq |D_q|\}$.
  Pour $1 \leq j \leq a_{q+1}^n$ on note $X_j = \{x_{jl} \in D_{q+1} | 1 \leq l \leq |D_q|\}$ et $Y_j = \{y_{jl} \in U_{q+1} | 1 \leq l \leq |D_q|\}$.

  Comme, pour $1 \leq j \leq a_{q+1}^n$, $A_{q+1}[X_j \cup Y_j] \cong A_q$, alors, par hypothèse de récurrence, $A_{q+1}[X_j \cup Y_j]$ est l'union de $q+1$ ordres de dimension 2.
  Il en est donc de même pour $A_{q+1}(X_j \times Y_j) = (D_{q+1}, H_{q+1}, E_j)$ qui est l'union de la famille d'ordres de dimension 2 $(R_i^j)_{1 \leq i \leq q+1}$ où $R_i^j = (D_{q+1}, H_{q+1}, P_i^j)$. On a

$$A_{q+1}(X_j \times Y_j) = \bigcup_{1 \leq i \leq q+1} R_i^j$$

  Pour $1 \leq i \leq q+1$, on pose $R_i = (D_{q+1}, H_{q+1}, \bigcup_{1 \leq j \leq a_{q+1}^n} P_i^j)$. Comme les $(P_i^j)_{1 \leq j \leq a_{q+1}^n}$ sont à supports disjoints, R_i est de dimension 2. On a :

$$\bigcup_{1 \leq i \leq q+1} R_i = (D_{q+1}, H_{q+1}, \bigcup_{1 \leq i \leq q+1} \bigcup_{1 \leq j \leq a_{q+1}^n} P_i^j)$$

  Or $\bigcup_{1 \leq i \leq q+1} \bigcup_{1 \leq j \leq a_{q+1}^n} P_i^j = \bigcup_{1 \leq j \leq a_{q+1}^n} \bigcup_{1 \leq i \leq q+1} P_i^j = \bigcup_{1 \leq j \leq a_{q+1}^n} E_j$, d'où

$$\bigcup_{1 \leq i \leq q+1} R_i = (D_{q+1}, H_{q+1}, \bigcup_{1 \leq i \leq a_{q+1}^n} E_j) = A_{q+1}(D_{q+1} \times U_{q+1})$$

  Comme en évidence $R_{q+2} = A_{q+1}(D_{q+1} \times \{c_{q+1}\})$ est de dimension 2, on a :

$$A_{q+1} = A_{q+1}(D_{q+1} \times U_{q+1}) \cup A_{q+1}(D_{q+1} \times \{c_{q+1}\}) = \bigcup_{1 \leq i \leq q+1} R_i \cup R_{q+2} = \bigcup_{1 \leq i \leq q+2} R_i$$

  A_{q+1} est donc l'union de $q+2$ ordres de dimension 2. $\mathcal{P}(q+1)$ est vraie.
- On vient de montrer que :
  $\forall p \in [\![1, n-1]\!], A_p = \prod_{i=1}^{p} L_{a_i^n}$ est l'union de $p+1$ ordres de dimension 2.

En particulier pour $p = n-1$, ceci montre que B_n est l'union de n ordres de dimension 2.

Conclusion : $u(D_2)\text{-}epais(B_n) = n$

$c(D_2)$-$epais(B_n) = 2$

Théorème 6 $\forall n \in \mathbb{N}^*, \forall (a_i)_{1\leq i\leq n} \in (\mathbb{N}^*)^n, c(D_2)\text{-}epais\left(\prod_{i=1}^{n} L_{a_i}\right) = 2$

Preuve :
On montre d'abord par récurrence la propriété suivante pour $n \in \mathbb{N}^*$:

$\mathcal{P}(p)$ = "En notant $\prod_{i=1}^{n} L_{a_i} = (D, U, P)$ et $S = (D, U, P_s)$,
$\forall (a_i)_{1\leq i\leq n} \in (\mathbb{N}^*)^n, \exists S \in \mathcal{OB}(D, U)$,
$P_s \cap P = \varnothing$
et il existe un ordre total $\mathcal{T}_s$ sur les éléments de U tel que les successeurs dans S des éléments de D sont des intervalles dans $\mathcal{T}_s$ et tel que tous les intervalles non vides ont l'élément maximal de $\mathcal{T}_s$ comme borne
et $R = (D, U, P \cup P_s)$ est tel qu' il existe un ordre total $\mathcal{T}$ sur les éléments de U tel que les successeurs dans R des éléments de D sont des intervalles dans $\mathcal{T}$ et tel que tous les intervalles non vides ont l'élément maximal de $\mathcal{T}$ comme borne "

- $\mathcal{P}(1)$ et $\mathcal{P}(2)$ sont vrais, voir précédemment.
- Supposons pour $n \in \mathbb{N}^*$ que $\mathcal{P}(n)$ est vrai Soit $(a_i)_{1\leq i\leq n+1} \in (\mathbb{N}^*)^{n+1}$, on note

$E_{n+1} = \prod_{i=1}^{n+1} L_{a_i} = (D, H, P)$ avec $H = U \cup \{c\}$ où c est le point critique de $L_{a_{n+1}}$

$$E_n = \prod_{i=1}^{n} L_{a_i} = (D', H', P')$$

On a donc

$$E_{n+1} = L_{a_{n+1}}\, E_n$$

On introduit les mêmes notations que précédemment :
On a $D = \{x_{kl} | 1 \leq k \leq a_{n+1} \text{ et } 1 \leq l \leq |D'|\}$
et $U = \{y_{kl} | 1 \leq k \leq a_{n+1} \text{ et } 1 \leq l \leq |D'|\}$.
Pour $1 \leq j \leq a_{n+1}$ on note $X_j = \{x_{jl} \in D | 1 \leq l \leq |D'|\}$ et $Y_j = \{y_{jl} \in U | 1 \leq l \leq |D'|\}$.
$F_j = (X_j, Y_j, P_j)$ avec $P_j = P[X_j \cup Y_j]$

D'après l'hypothèse de récurrrence , comme pour $1 \leq j \leq a_{n+1}$, $F_j \cong E_n$:
$\forall j \in [\![1, a_{n+1}]\!], \exists S_j \in \mathcal{OB}(X_j, Y_j), S_j = (X_j, Y_j, P_{S_j})$ et $P_{s_j} \cap P_j = \varnothing$
et il existe un ordre total $\mathcal{T}_{S_j}$ sur les éléments de Y_j tel que les successeurs dans S_j des éléments de X_j sont des intervalles dans $\mathcal{T}_{S_j}$ et tel que tous les intervalles non vides ont l'élément maximal de $\mathcal{T}_{S_j}$ comme borne
et $R_j = (X_j, Y_j, P_j \cup P_{S_j})$ est tel qu' il existe un ordre total $\mathcal{T}_j$ sur les éléments de Y_j tel que les successeurs dans R_j des éléments de X_j sont des intervalles dans $\mathcal{T}_j$ et tel que tous les intervalles non vides ont l'élément maximal de $\mathcal{T}_j$ comme borne

On construit $S = (D, H, P_S)$ ainsi :
Un point $d \in X_j$ est inférieur dans S à :
– tous les $y \in Y_j$ tels que $(x, y) \in P_{S_j}$
– tous les $y \in Y_i$ avec $i > j$
Comme $P_{S_j} \cap P_j = \varnothing$ et comme dans E_{n+1}, les successeurs de $d \in X_j$ sont dans Y_j, par construction de P_S, on a $P_S \cap P = \varnothing$.
On définit $\mathcal{T}_S$ ainsi :

$$\forall (x, y) \in H, \quad x < y \iff \begin{array}{l} x = c \\ ou \\ x \in Y_i \text{ et } y \in Y_j \text{ et } \ i < j \\ ou \\ (x, y) \in {Y_i}^2 \text{ et } x <_{\mathcal{T}_{S_i}} y \end{array}$$

Pour tout $d \in D$, les successeurs de d dans S sont bien des intervalles dans $\mathcal{T}_S$ et les intervalles non vides ont le maximum de $\mathcal{T}_{S_{a_{n+1}}}$ qui est aussi le maximum de $\mathcal{T}_S$ comme borne supérieure.

On définit $\mathcal{T}$ ainsi :

$$\forall (x, y) \in H, \quad x < y \iff \begin{array}{l} y = c \\ ou \\ x \in Y_i \text{ et } y \in Y_j \text{ et } \ i < j \\ ou \\ (x, y) \in {Y_i}^2 \text{ et } x <_{\mathcal{T}_i} y \end{array}$$

Pour tout $d \in D$, les successeurs de d dans E_{n+1} sont bien des intervalles dans $\mathcal{T}$ et les intervalles non vides ont c qui est le maximum de $\mathcal{T}$ comme borne supérieure.

Donc, P(n+1) est vrai.
Ce qui termine la récurrence.

Soient $n \in \mathbb{N}^*$ et $(a_i)_{1 \leq i \leq n} \in (\mathbb{N}^*)^n$, montrons que $c(D_2)\text{-}epais(\prod_{i=1}^{n} L_{a_i}) = 2$:
D'après la récurrence précédente, en notant $\mathcal{P} = \prod_{i=1}^{n} L_{a_i} = (D, H, P)$, $\exists S \in \mathcal{OB}(D, H)$ tel que $S = (D, H, P_S)$ et $P_S \cap P = \varnothing$ et S est biconvexe fermé et $R = (D, H, P \cup P_S)$ est biconvexe fermé.
On note $S^- = (H, D, P_S^-)$ avec P_S^- défini ainsi : $(x, y) \in P_S^- \iff (y, x) \in P_S$.

$dim(R) = dim(S^-) = 2$ et (R, S^-) est donc une $c(D_2)$-réalisation de $\mathcal{P}$.

$$c(D_2)\text{-}epais(\prod_{i=1}^{n} L_{a_i}) = 2$$

Corollaire 1 $\forall n \in \mathbb{N} \setminus \{0, 1\}, c(D_2)\text{-}epais(B_n) = 2$

En effet, $B_n = \prod_{i=1}^{n-1} L_{a_i^n}$.

conclusion

Nous avons vu en 3.2 que la $c(D_2)$-réalisation est plus expressive que la $i(D_1)$-réalisation. Nous venons de montrer qu'elle est aussi plus expressive que la $u(D_2)$-réalisation. Cela n'enlève pas pour autant son intérêt à la $c(D_2)$-réalisation. Tout d'abord, car comme le codage par $c(D_2)$-réalisation est une généralisation de la dimension il risque d'etre difficile à exploiter. De surcroît, Il a été difficile de construire une famille d'ordres qui aient des $c(D_2)$-épaisseurs croissantes. La famille que nous avons utilisé a une croissance plus qu'exponentielle du nombre de points dans les ordres. Cela peut laisser penser que les familles qui ont une $u(D_2)$-épaisseur bornée contiennent un grand nombre d'ordres. La $u(D_2)$-réalisation est donc une notion dont l'étude reste à faire et qui pourrait s'avérer prometteuse.

Chapitre 5

De nouvelles dimensions orientées vers le codage

Dans ce chapitre, nous présentons deux nouveaux codages. Le premier, s'appuyant sur le concept de dimension centrée que nous introduisons, est un codage ad'hoc pour la décomposition d'un ordre par ses points, fréquemment utilisée dans la littérature. Nous cherchons des sous ensembles de points de l'ordre dont les relations avec tous les points de l'ordre, y compris eux-mêmes, soient codables par deux entiers (voir dimension centrée introduite ci-après). Nous montrons qu'en faisant cela, on obtient des codes en $O(n)$ bits par étiquette dans le pire des cas.

Le deuxième codage est le codage le plus général que nous présentons. Son originalité réside dans le calcul récurrent que demande son interprétation. La motivation étant d'éviter par un décodage plus long la redondance d'information. Les possibilités de ce codage n'ont pas été vraiment approchée dans le stage, mais il semble digne d'intérêt, au moins par son caractère original.

5.1 Décomposition d'un ordre par ses points

5.1.1 Dimension centrée

Définition 13 *La dimension centrée d'un ordre partiel $\mathcal{P} = (X, P)$, notée $cdim(\mathcal{P})$, avec $|X| = n$, est le plus petit entier naturel k tel qu'il existe une partition de $X = Y \sqcup Y'$ et un ordre partiel $\mathcal{Q} = (X, Q)$ tel que $P = Q((Y' \times X) \cup (X \times Y'))$ et X se plonge dans $[\![1, n]\!]^k$ de la manière suivante :*

$$\begin{array}{rcl} X & \to & [\![1, n]\!]^k \\ x & \mapsto & v(x) = (v_1(x), \ldots, v_k(x)) \end{array}$$

avec $x <_{\mathcal{Q}} y$ si et seulement si $v(x) < v(y)$ c'est à dire :
$\forall i \in [\![1, k]\!]$, $v_i(x) \leq v_i(y)$ et $\exists j \in [\![1, k]\!]$, $v_j(x) < v_j(y)$

Cette définition est correcte car il existe toujours une partition et un plongement satisfiant la propriété. On peut prendre par exemple $Y = \{x\}$, où $x \in X$,

et $Y' = X \setminus \{x\}$. La propriété est alors satisfaite avec $\mathcal{Q} = \mathcal{P}$ et $k = dim(\mathcal{P})$. Ce qui montre aussi que $cdim(\mathcal{P}) \leq dim(\mathcal{P})$.

Remarque :
Nous emploierons le même vocabulaire pour la dimension centrée que pour la dimension :

Un ordre partiel $\mathcal{P}$ est dit de dimension centrée k s'il admet une partition et un plongement, selon la définition précédente, dans $[\![1, n]\!]^k$.
La dimension d'un ordre partiel $\mathcal{P}$ est le minimum des k tel que $\mathcal{P}$ est de dimension centrée k.

Pour $n \geq 1$, on note C_n l'ensemble des ordres partiels de dimension centrée n.

Le codage issu de la dimension centrée

A chaque élément x d'un ordre de dimension centrée k, on peut associer une étiquette $e(x) = (s(x), e_1(x), \ldots, e_k(x))$ avec $s(x) = 1$ si et seulement si $x \in Y$ et $e_1(x), \ldots, e_k(x)$ les k entiers du plongement dans $[\![1, n]\!]^k$.

Ainsi, $x < y$ si et seulement si :

$$s(x) \vee s(y) = 1 \quad \text{et} \quad (e_1(x), \ldots, e_k(y)) < (e_1(y), \ldots, e_k(y))$$

.

5.1.2 $c(C_2)$-réalisation d'un ordre partiel

Nous allons montrer que tout ordre partiel $\mathcal{P} = (X, P)$, avec $|X| = n$, admet une $c(C_2)$-réalisation de cardinal $\lceil \frac{n}{4} \rceil$.
Nous montrons, de plus, que chaque ordre de la réalisation peut être codé avec 8 bits, ce qui donne une représentation de l'ordre $\mathcal{P}$ par attribution d'étiquettes d'au plus $2n + 6$ bits à chaque élément.

Théorème 7 *Tout ordre partiel biparti $\mathcal{P} = (L, U, P)$, avec $|X| = n$, admet une $c(C_2)$-réalisation de cardinal inférieur à $\lceil \frac{n}{4} \rceil$, et dans lequel chaque ordre du réaliseur est codé avec des étiquettes d'au plus 4 bits.*

Y-indiscernabilité

Soit $\mathcal{P} = (X, P)$ un ordre partiel, soient $x \in X$ et $Y \subseteq X$. On note $D(x, Y) = \{y \in Y | y < x\}$ et $U(x, Y) = \{y \in Y | x < y\}$.

Deux éléments x et y de X sont dits indiscernables par rapport à Y si :

$$D(x, Y) = D(y, Y) \text{ et } U(x, Y) = U(y, Y)$$

La relation *être indiscernable par rapport à Y* est une relation d'équivalence sur les éléments de $X \setminus Y$, on peut donc naturellement partitionner $X \setminus Y$ en classes d'équivalence appelées classes d'Y-indiscernabilité.

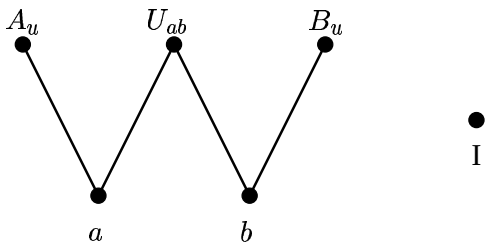

FIG. 5.1 – Classes d'$\{a, b\}$-indiscernabilité

Support d'un ordre

On appelle support d'un ordre $\mathcal{P} = (X, P)$ l'ensemble des éléments de X qui sont en relation avec un autre élément de l'ordre :

$$support(\mathcal{P}) = \{x \in X | \exists y \in X, x < y \; ou \; y < x\}$$

Preuve du théorème 7 :

L'information pertinente de l'ordre est contenue dans son support, la classe des éléments isolés étant facilement codable.

Pour des raisons de symétrie, on peut considérer $|L \cap support(\mathcal{P})| \leq |U \cap support(\mathcal{P})|$ dans la suite de la démonstration.

On prouve d'abord, par récurrence sur $k = |L \cap support(\mathcal{P})|$, une propriété plus forte que le théorème.L'hypothèse de récurrence est la suivante pour $p \geq 2$:

P(p) = " Tout ordre partiel biparti $\mathcal{P} = (L, U, P)$, avec $k = |L \cap support(\mathcal{P})| \leq p$, admet une réalisation à l'aide d'ordres de dimension empreinte 2, de cardinal inférieur à $\lceil \frac{k}{2} \rceil$, et dans lequel chaque ordre du réaliseur est codé en attribuant aux éléments des étiquettes d'au plus 4 bits "

- pour $p = 2$

  On note a, b les 2 éléments de $L \cap support(\mathcal{P})$. Si $k = 1$ on peut enlever b dans le codage suivant sans changer le reste.

  On regarde les classes d' $\{a, b\}$-indiscernabilité données par la figure 5.1.

  Par exemple, A_u représente l'ensemble des éléments qui sont supérieur à a et pas à b.

  Tous les éléments d'une classe se voient attribuer le même code.

  Les classes peuvent être vides sans que cela ne modifie le code des éléments des autres classes.

  L'étiquette associée à un élément x est la suivante :

$$s(x) = \begin{cases} 1 & \text{si x=a ou x=b} \\ 0 & sinon \end{cases}$$

  Les 2 entiers attribués à chaque élément sont donnés par le tableau :

3		A_u, U_{ab}
2		a, I
1	b, B_u, U_{ab}	B_u
0	a, A_u, I	b

On vient de construire un réaliseur de $\mathcal{P}$ de cardinal 1.
Or $1 \leq \lceil \frac{k}{2} \rceil \leq \lceil \frac{n}{4} \rceil$ car $k \leq \frac{n}{2}$. On remarque que ce codage demande 4 bits : 1 bit pour s(x), 1 bit pour le premier entier et 2 pour le second.

- pour $p \geq 3$
  Soit un ordre biparti $\mathcal{P} = (L, U, P)$ avec $k = p$.
  Soient a et b deux éléments de $L \cap support(\mathcal{P})$.
  Le cas $|L| = 2$ traité ci-dessus montre que $\mathcal{P}_r = \mathcal{P}((\{a,b\} \times X) \cup (X \times \{a,b\}))$ est un ordre de dimension centrée 2, codable avec 4 bits par élément.
  En utilisant l'hypothèse de récurrence, on obtient une réalisation de $\mathcal{P}((X \setminus \{a,b\}) \times (X \setminus \{a,b\}))$ de cardinal $q \leq \left\lceil \frac{|(L \setminus \{a,b\}) \cap support(\mathcal{P})|}{2} \right\rceil = \lceil \frac{k-2}{2} \rceil$. On note $\mathcal{F} = (\mathcal{P}_1, \ldots, \mathcal{P}_q)$ cette réalisation.
  $\mathcal{F}' = (\mathcal{P}_1, \ldots, \mathcal{P}_q, p_r)$ est donc une réalisation de $\mathcal{P}$ à l'aide d'ordres de dimension centrée 2 dans laquelle chque ordre est codé en attribuant aux éléments des étiquettes de 4 bits. De plus, $|\mathcal{F}'| \leq \lceil \frac{k-2}{2} \rceil + 1 = \lceil \frac{k}{2} \rceil$.

Ce qui termine la récurrence.

Comme $k \leq \frac{|support(\mathcal{P})|}{2}$, on a $\lceil \frac{k}{2} \rceil \leq \left\lceil \frac{|support|}{4} \right\rceil \leq \lceil \frac{n}{4} \rceil$.
Ce qui achève la preuve du théorème 7.

Théorème 8 *Tout ordre partiel $\mathcal{P} = (X, P)$, avec $|X| = n$, admet un réalisation à l'aide d'ordres de dimension centrée 2, de cardinal inférieur à $\lceil \frac{n}{4} \rceil$, et dans lequel chaque ordre de la réalisation est codé avec des étiquettes d'au plus 8 bits.*

Preuve :
On note k le cardinal du support de $\mathcal{P}$.
On prouve d'abord, par récurrence sur $k = |L \cap support(\mathcal{P})|$, une propriété plus forte que le théorème.L'hypothèse de récurrence est la suivante pour $p \geq 4$:

P(p) = " Tout ordre partiel $\mathcal{P} = (X, P)$, avec $k \leq p$, admet une réalisation à l'aide d'ordres de dimension centrée 2, de cardinal inférieur à $\lceil \frac{k}{4} \rceil$, et dans lequel chaque ordre de la réalisation est codé avec des étiquettes d'au plus 8 bits. "

- pour $p = 4$
  Un ordre $\mathcal{P}$ avec $k \leq 4$ est de dimension 2, donc a fortiori de dimension centrée 2 et est codable dans ce formalisme en attribuant à chaque élémentune étiquette de 8 bits.
- pour $p \geq 5$ Si l'ordre est biparti, la démonstration du théorème 7 donne le résultat.
  Si l'ordre n'est pas biparti, on va montrer qu'il existe 4 éléments a, b, c, d dont les relations avec les autres sont un ordre de dimension centrée 2, $\mathcal{P}((\times X) \cup (X \times \{a,b,c,d\}))$ avec les notations introduites, codable avec

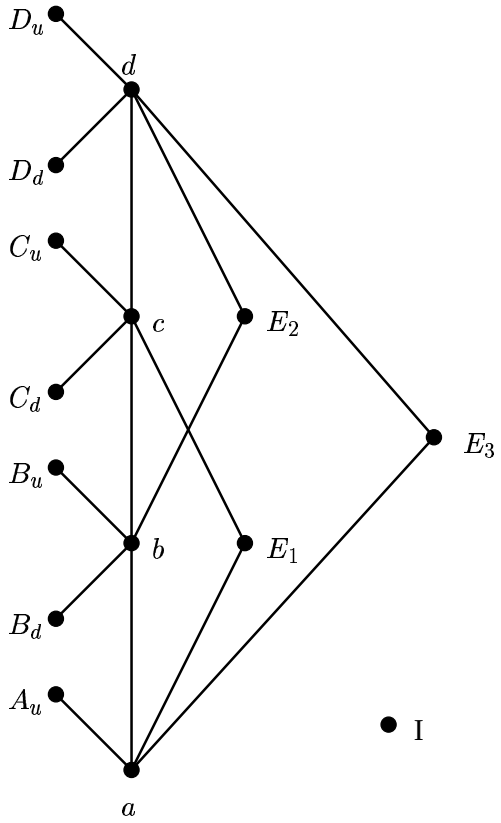

FIG. 5.2 – Classes d'indiscernabilité d'une chaîne à 4 éléments

des étiquettes de 8 bits par éléments.
Une fois montré cela, on conluera la récurrence comme suit.
En utilisant l'hypothèse de récurrence, on obtient une réalisation de $\mathcal{P}((X \setminus \{a,b,c,d\}) \times (X \setminus \{a,b,c,d\}))$ de cardinal $q \leq \left\lceil \frac{|(X\setminus\{a,b,c,d\})\cap support(\mathcal{P})|}{4} \right\rceil = \lceil \frac{k-4}{4} \rceil$. On note $\mathcal{F} = (\mathcal{P}_1, \ldots, \mathcal{P}_q)$ cette réalisation.
$\mathcal{F}' = (\mathcal{P}_1, \ldots, \mathcal{P}_q, p_r)$ est donc une réalisation de $\mathcal{P}$ à l'aide d'ordres de dimension centrée 2 dans laquelle chaque ordre est codé en attribuant aux éléments des étiquettes de 8 bits. De plus, $|\mathcal{F}'| \leq \lceil \frac{k-4}{4} \rceil + 1 = \lceil \frac{k}{4} \rceil$.
Il nous reste donc à montrer que pour $k = 5$, il existe 4 éléments du support de $\mathcal{P}$, a, b, c, d, tels que $\mathcal{P}((\times X) \cup (X \times \{a,b,c,d\}))$ est de dimension centrée 2. On fait une disjonction de cas pour trouver 4 tels éléments.

1. il existe une chaîne à 4 éléments

   S'il existe une chaîne à 4 éléments alors il existe une chaîne à 4 éléments $a < b < c < d$ avec a minimal dans $\mathcal{P}$ et a, b, c, d consécutifs dans le diagramme de Hasse.

   Le diagramme de Hasse des classes de $\{a,b,c,d\}$-indiscernabilité est donné par la figure 5.2.

   Des propriétés de la chaîne choisie, découle l'absence de classe d'éléments plus petit que a, ainsi que l'absence de classe d'éléments com-

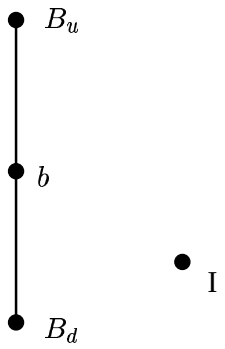

FIG. 5.3 – Classes de $\{b\}$-indiscernabilité

pris entre a et b, entre b et c et entre c et d.
Un code de $\mathcal{P}((\{a,b,c,d\} \times X) \cup (X \times \{a,b,c,d\}))$ possible est le suivant :

$$s(x) = \begin{cases} 1 & \text{si } x \in \{a,b,c,d\} \\ 0 & sinon \end{cases}$$

Les 2 entiers attribués à chaque élément sont donnés par le tableau :

7	D_u, I	A_u, B_u, C_u, D_u
6	d	d
5	C_u, D_d	E_2, E_3
4	c	c
3	B_u, C_d, E_2	E_1
2	b	b
1	A_u, B_d, E_1, E_3	a
0	a	B_d, C_d, D_d, I

2. il n'existe pas de chaine à 4 éléments
   Comme l'ordre n'est pas biparti, il existe une chaine à 3 éléments. Il existe donc une chaîne à 3 éléments $a < b < c$ tels que a est minimal dans $\mathcal{P}$ et a, b, c sont consécutifs dans le diagramme de Hasse.
   Pour le quatrième élément :

   (a) si tous les éléments du support de $\mathcal{P}$ sont comparables à b
   Alors, les classes de $\{b\}$-indiscernabilité sont réduites aux classes de la figure 5.3.
   et les éléments de B_u sont incomparables entre eux, de même pour ceux de B_d et de I. Cet ordre est de dimension 2, donc de dimension centrée 2.
   Un code de $\mathcal{P}$ possible est :
   $\forall x \in X, s(x) = 1$
   Les 2 entiers attribués à chaque élément sont donnés par le tableau :

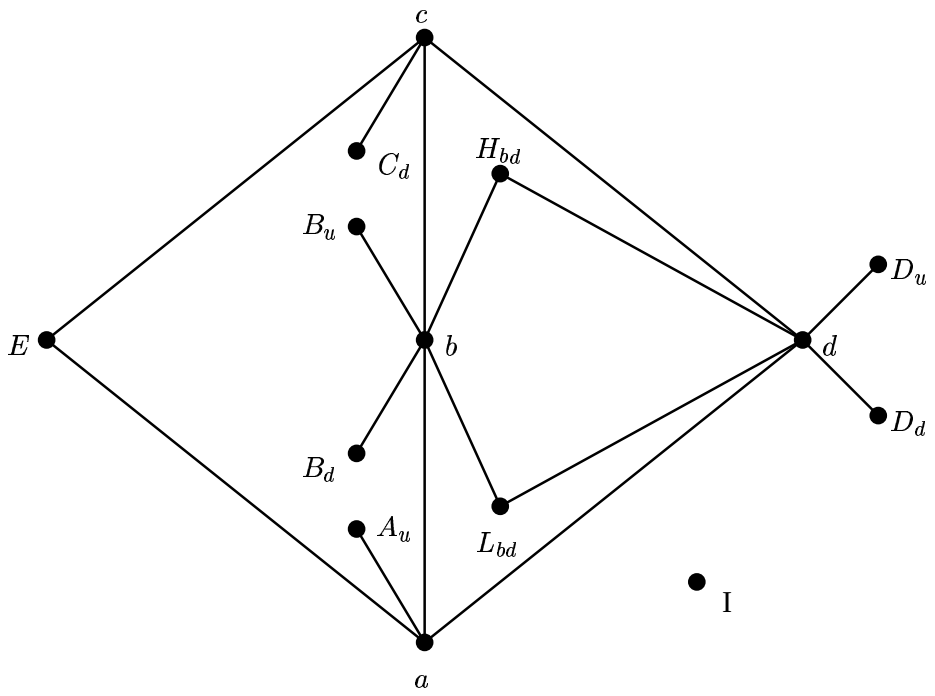

FIG. 5.4 – 1^{er} cas

2	B_u	
1	b	
0	B_d	b, B_u, B_d

(b) s'il existe un élément $d \in X$ tel que $d||b$

On distingue les 3 cas suivants :

1^{er} cas : $a < d < c$

Le diagramme de Hasse des classes d'$\{a, b, c, d\}$-indiscernabilité est donné par la figure 5.4.

Il n'y a pas d'éléments entre a et d ni entre c et d car sinon il y aurait une chaîne à 4 éléments. Pour les mêmes raisons, il n'y pas d'éléments supérieurs à c ni inférieurs à a. Comme a, b, c sont consécutifs, il n'y a pas d'éléments compris entre a et b ni entre b et c.

Un code de $\mathcal{P}\left((\{a, b, c, d\} \times X) \cup (X \times \{a, b, c, d\})\right)$ possible est le suivant :

$$s(x) = \begin{cases} 1 & \text{si } x \in \{a, b, c, d\} \\ 0 & sinon \end{cases}$$

Les 2 entiers attribués à chaque élément sont donnés par le tableau :

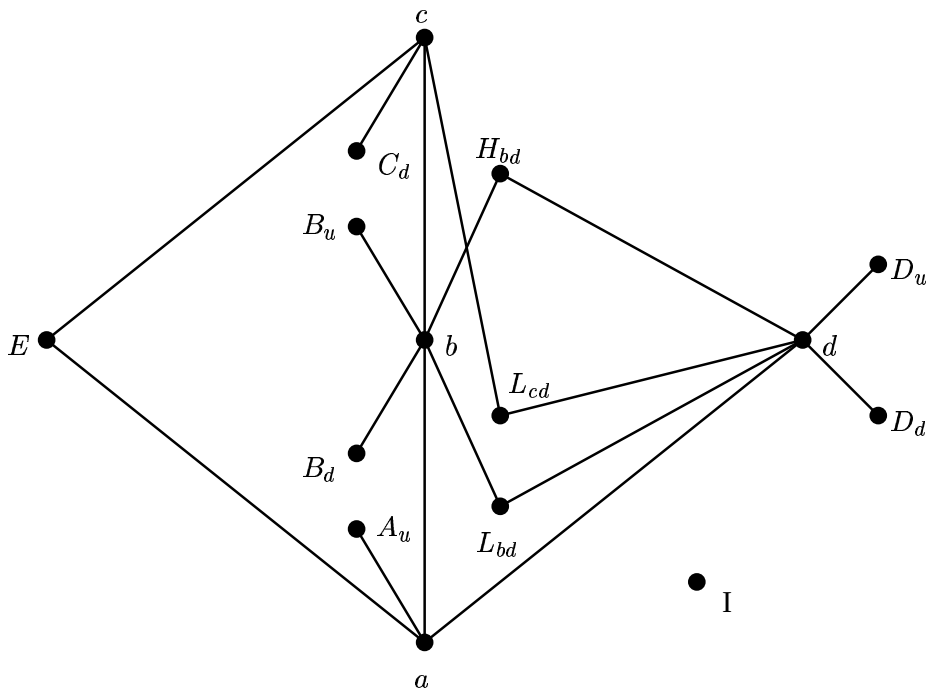

FIG. 5.5 – $2^{ème}$ cas

8	B_u	
7	c	
6	C_d, H_{bd}	A_u, D_u, H_{bd}, I
5	b	c
4	B_d, D_u, E	D
3	d	B_u, D_d, E
2	A_u, L_{bd}	b
1	a	a
0	D_d, I	B_d, C_d, L_{bd}

$2^{ème}$ cas $a < d$ et $d||c$

Le diagramme de Hasse des classes d'$\{a, b, c, d\}$-indiscernabilité est donné par la figure 5.5.

Un code de $\mathcal{P}((\{a, b, c, d\} \times X) \cup (X \times \{a, b, c, d\}))$ possible est le suivant :

$$s(x) = \begin{cases} 1 & \text{si } x \in \{a, b, c, d\} \\ 0 & sinon \end{cases}$$

Les 2 entiers attribués à chaque élément sont donnés par le tableau :

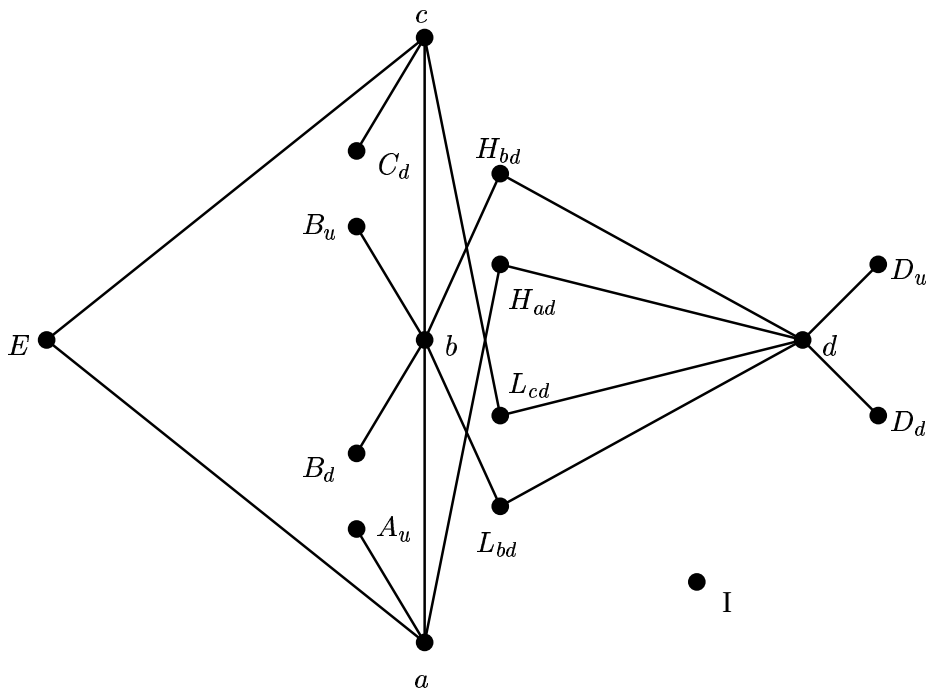

FIG. 5.6 – $3^{ème}$ cas

8	B_u	
7	c	A_u, D_u, H_{bd}, I
6	C_d, H_{bd}	d
5	b	D_d
4	B_d, D_u, E	c
3	d	B_u, L_{cd}, E
2	A_u, L_{bd}	b
1	a	a
0	D_d, L_{cd}, I	B_d, C_d, L_{bd}

$3^{ème}$ cas $a||d$ et $d||c$

Le diagramme de Hasse des classes d'$\{a, b, c, d\}$-indiscernabilité est donné par la figure 5.6.

Un code de $\mathcal{P}\left((\{a, b, c, d\} \times X) \cup (X \times \{a, b, c, d\})\right)$ possible est le suivant :

$$s(x) = \begin{cases} 1 & \text{si } x \in \{a, b, c, d\} \\ 0 & sinon \end{cases}$$

Les 2 entiers attribués à chaque élément sont donnés par le tableau :

8	B_u, A_u	
7	c	D_u, H_{ad}, H_{bd}, I
6	E, H_{bd}	d
5	b	D_d
4	H_{ad}	c
3	a	B_u, C_d, L_{cd}
2	B_d, C_d, D_u	b
1	d	A_u, B_d, E, L_{bd}
0	D_d, L_{bd}, L_{cd}, I	a

Ce qui termine la récurrence.

De plus, comme $k = |support(\mathcal{P}) \leq |X| = n$, on a $\lceil \frac{k}{4} \rceil \leq \lceil \frac{n}{4} \rceil$ et on a bien montré le théorème 8.

5.2 Codage par chaînes parenthésées

Pour un mot w de longueur fini sur un alphabet A et un élément $a \in A$, on note $|w|_a$ le nombre de a dans le mot w.

Une expression bien parenthésée est un mot fini w sur un alphabet A contenant les symboles (et) et tel que :

$$\text{pour tout } w' \text{ préfixe de } w, |w'|_(\geq |w'|_) \quad \text{et} \quad |w|_(= |w|_)$$

Exemple : $AC((X(D)F(C))Y)G(C)UI$ est bien parenthésée sur l'alphabet latin.

Si l'alphabet A ne contient que des relations binaires sur un meme ensemble sous-jacent X et les symboles (et), on parle d'expression bien parenthésée de relations binaires sur X.

Exemple : Soient R, S, T des relations binaires sur un meme ensemble X, $R(S(T)SR)R(T)$ est une expression bien parenthésée de relations binaires sur X.

Une expression bien parenthésée avec une parenthèse englobante est une expression bien parenthésée w qui commence par (et telle que pour tout préfixe w' différent de w, $|w'|_(> |w'|_)$.

Exemple : $(xd(sdf)fg(cv)h)$ est une expression bien parenthésée avec parenthèse englobante.

Parenthésage de relations binaires sur X

On appelle parenthésage de relations binaires sur X une expression bien parenthésée de relations binaires sur X avec une parenthèse englobante. Si les

relations binaires sont des ordres linéaires, on parle de parenthésage d'ordres linéaires sur X.

Soit un parenthésage w de relations binaires sur X, soit w' tel que $w = (w')$, alors il existe ununique découpage de $w' = w'_1\, w'_2 \cdots w'_n$ tel que $\forall i \in [\![1, n]\!], w'_i$ est une relation binaire ou un parenthésage de relations binaires. On appelle ce découpage le découpage propre de w.

Résultat d'un parenthésage de relations binaires sur X

On définit par induction la relation binaire résultat d'un parenthésage de relations binaires sur X :

- Le résultat d'un parenthésage w de relations binaires tel que $|w|_(= |w|_) = 1$ et contenant l'ensemble de relations binaires $\mathcal{R} = \{R_1 \cdots R_n\}$ est la relation binaire R_t sur X définie par :

$$\forall (x, y) \in X^2, xR_t y \iff \exists i \in [\![1, n]\!], xR_i y \text{ et } \nexists j \in [\![1, n]\!], yR_j x$$

  On note $R_t = (R_1 \cdots R_n)$.
- Soit w un parenthésage de relations binaires sur X dont le découpage propre est $w' = w'_1\, w'_2 \cdots w'_n$. On pose, pour tout $1 \leq i \leq n$, $R_i = w'_i$ si w'_i est une relation binaire, sinon R_i est la relation binaire sur X résultat du parenthésage w'_i de relations binaires sur X. Le résultat du parenthésage w est la relation binaire $R_t = (R_1 \cdots R_n)$ telle qu'elle est défini ci-dessus.

Exemple : Soit R, S, T des relations d'ordres sur X. Soit $(x, y) \in X^2$ tels que $x <_R y, y <_S x$ et $x||y$ dans T. Soit le parenthésage de relations binaires sur X $U = (((RS)T)(R(RS)))$. Comme $x < y$ dans R et $y < x$ dans S, alors $x||y$ dans (RS). Or $x||y$ dans T, d'où $x||y$ dans $((RS)T)$. $x < y$ dans $(R(RS)$. Et finalement $x < y$ dans $(((RS)T)(R(RS)))$.

Réalisation par chaînes parenthésées d'un ordre partiel

Un paranthésage w d'ordres linéaires sur X est une réalisation par chaînes parenthésées de $\mathcal{P} = (X, P)$ si et seulement si le résultat de w est P.

Conclusion

Dans ce rapport, nous avons proposé un formalisme général du codage par étiquettes. Nous avons également introduit un formalisme très précis dans lequel entre la plupart des variations de la notion de dimension qu'on rencontre dans la littérature. Ce formalisme, dit de la $\lambda(\mathcal{E})$-réalisation, nous a permis de mieux cerner où résident les différences entre les codages existants et de les classer dans une hiérarchie en fonction de leurs performances relatives. Nous avons constaté que pour les codages du haut de la hiérarchie, qui sont les généralisations de la notion de dimension, nous ne savions pas donner d'algorithme polynomial qui ait des performances meilleures qu'en $O(n)$ bits par étiquette, c'est à dire en $O(log\, n)$ bits, sur des familles importantes d'ordres. Pourtant, ces codages peuvent potentiellement fournir des codes en $O(log\, n)$ bits par étiquette sur les familles d'ordres dont la dimension est bornée. Ces remarques nous ont amenés à nous intéresser à une approche transversale à celle de la théorie de la dimension qui code un ordre par intersection d'ordres linéaires. Nous avons proposé un codage par union d'ordres de dimension 2, le codage par $u(D_2)$-réalisation. Nous avons montré que celui-ci est moins expressif qu'un codage mixte que nous avons également introduit, le codage par $c(D_2)$-réalisation. Cependant, la démonstration que nous donnons de ce résultat nous laisse penser que les performances du codage par union sont déjà très intéressantes, d'autant plus qu'il paraît difficile d'obtenir des algorithmes d'encodage efficaces pour le codage par $c(D_2)$-réalisation car celui-ci est une généralisation du codage issu de la théorie de la dimension. Les questions issues de ce travail sont :

- Le codage par $u(D_2)$-réalisation possède-t-il un pire des cas en $O(n)$?
- Si oui, peut-on trouver un algorithme polynomial qui garantisse ces performances ?
- Les problèmes de trouver la $u(D_2)$-épaisseur d'un ordre et une $u(D_2)$-réalisation de cet ordre sont-ils polynomiaux ?
- Sinon, quelles sont les garanties offertent par les algorithmes polynomiaux sur ces questions ?

Bibliographie

[1] G.-V. Jourdan, L'analyse d'exécutions réparties en utilisant la théorie de l'ordre, Thèse à l'Université de Rennes 1, 1995.

[2] C. Diehl, Analyse de la relation de causalité dans les exécutions réparties, Thèse à l'Université de Rennes 1, 1992.

[3] J. Fidge, Timestamps in message passing systems that preserve the partial ordering. In Proc. 11^{th} Australian Computer Science Conference, pages 55-66, february 1988.

[4] F. Mattern. Virtual time and global states of distributed systems. In Cosnard, Quinton, Raynal, and Robert, editors, Proc. Int. Workshop on Parallel and Distributed Algorithms Bonas, France, Oct. 1988, north Holland, 1989.

[5] Vijay K. Garg and Chakarat Skawratananond, String Realizers of Posets with Applications to Distributed Computing, http ://www.citeseer.nj.nec.com/garg01string.html.

[6] Neeraj Mittal and Vijay K. Garg, Rectangles are Better than Chains for Encoding Partially Ordered Sets, http ://www.utdallas.edu/ neerajm/.

[7] C. de la Higuera and L. Nourine, Drawing and Encoding Two Dimensional Posets, TCS Special Issue on Orders, Algorithmes and Applications", Vol 175 pp :293-308, 1997.

[8] David Kelly, William T. Trotter, Jr. Dimension theorie for ordered sets, I. Rival(ed), Ordered Sets, 171-211.

[9] American Mathematical Society Colloquium Publications, volume xxv, Third (New) Edition, 1967.

[10] B. Dushnik and E.W. Miller. Partially ordered sets, Amer. J. Math. 63, 600-610. MR3, p.73, 1941.

[11] Eric Thierry, Sur quelques intéractions entre structures de données et algorithmes efficaces, Thèse à l'université Montpellier II, 2001.

[12] Jong Youl Kim et Jeh Gwon Lee, Weak Dimension and Chain-weak Dimension of ordered Sets, 2002.

Résumé

Ce rapport présente une approche du codage des ordres partiels : celle du codage par étiquettes dont nous donnons un cadre formel général. Nous introduisons également un cadre plus précis, celui des réalisations paramétrées par un prédicat logique, ou $\lambda(\mathcal{E})$-réalisations, dans lequel nous plaçons quelques codages existants variant autour de la théorie de la dimension des ensembles ordonnés. Nous utilisons ce cadre pour comparer les codages entre eux et en construire une hiérarchie. Nous y plaçons trois nouveaux codages dont ceux par union, ou $u(D_2)$-réalisation, et par chaînes parenthésées. Nous démontrons quelques propriétés sur ces codages.

Mots clés : ordres partiels, codage, hiérarchie de codages, $\lambda(\mathcal{E})$-réalisation, codage par union, codage par chaînes parenthésées.

Abstract

This report presents a labelling approach for partially ordered sets encoding problem. We first introduce a general formal framework of the labelling encoding. A more precise framework specifically designed for some encoding methods using realizers parameterized by logic predicate, called $\lambda(\mathcal{E})$-realizers, has also been introduced. With this framework, a set of existing methods as variations of ordered sets dimension theory can be more clearly understood. The framework is then used to compare and build a hierarchy among some existing encoding methods. The hierarchy is enlarged by adding three new encoding methods of which those by union, called $u(D_2)$-realizers, and by parenthesised chains. We prove some properties of these encoding methods.

Key words : partially ordered sets, encoding, hierarchy of encoding methods, $\lambda(\mathcal{E})$-realizers, encoding by union, encoding with parenthesised chains.