Hiérarchie MSO et langages d'images bien proportionnées

Patrick GARDY

7 septembre 2013

Table des matières

1 Introduction **2**

2 Hiérarchie MSO et langages d'images **3**
2.1 Définitions 3
2.2 Logique MSO et langages de mots 5
2.3 Logique MSO et langages d'images 5
2.4 Choix de la structure logique 6

3 EMSO et reconnaissabilité **7**
3.1 EMSO=REC 8
3.2 Propriétés des langages reconnaissables 8
3.3 Degrés de liberté dans la notion de localité 10

4 Hiérarchie MSO sur les images quelconques **16**
4.1 Croissance des fonctions définissables dans Σ_k 17
4.2 Témoin de séparation 17
4.3 Spécificité du témoin de séparation 18

5 Divers résultats sur les images proportionnées **19**
5.1 Pliage d'images proportionnées 19
5.2 Rembourrage de langages d'images 26

6 Conclusion **30**

7 Bibliographie **31**

8 Appendice **31**

1 Introduction

Depuis le théorème de Fagin, on sait que la classe de complexité NP est exactement la classe des problèmes exprimables par une formule ESO. Ce théorème a mis en évidence l'existence d'un lien entre les ressources nécessaires pour résoudre un problème et les ressources nécessaires pour le définir. Nous nous intéressons ici à la logique MSO et à son expressivité sur les langages d'images. MSO est une classe logique suffisamment large pour exprimer des problèmes NP complets (l'exemple le plus connu étant la 3-colorabilité).

De nombreux travaux ont tenté d'établir une classification des langages (d'images comme de graphes) en fonction de la formule MSO les définissant. Deux hiérarchies ressortent. La première, la hiérarchie MSO a pour fondation la forme prénexe des formules du second ordre et le nombre d'alternances des différents types de quantificateurs du second ordre monadique. La seconde est la hiérarchie close de Ajtai/Fagin/Stockmeyer [1] qui est une version similaire à la première hiérarchie et qui autorise les quantifications du premier ordre entre les quantifications du second ordre. Dans ce stage nous nous intéressons à la première notamment pour sa simplicité. Nous nous restreignons aussi à la classe des images (mots de dimension deux). Avec la naissance de cette classification est apparue une question : la hiérarchie est-elle stricte ? Peut-on simplifier une formule appartenant à un niveau élevé de cette hiérarchie en une formule équivalente qui appartient à un niveau moindre ?

Plusieurs résultats sur cette hiérarchie MSO sur langages d'images sont connus. De par les travaux de Thomas, Matz et Schweikardt [10],[13] il est avéré que cette hiérarchie est stricte. Leurs travaux répondent à la question de la légitimité de la hiérarchie MSO, cependant leurs preuves ont de nombreux inconvénients. Le principal problème étant que la séparation de deux niveaux de la hiérarchie est faite par un langage dont les images sont exactement celles dont la longueur est de taille une tour d'exponentielle de la largeur. Ainsi une image du langage séparant le premier du deuxième niveau de 3 pixels de large sera longue de $(3*2^3)$=24 pixels, entre le deuxième et le troisième niveau une image de 3 pixels de large aura $24*2^{24}$=402 653 184 pixels de longueur. On comprend vite que de telles images ne sont pas naturelles. Ceci a amené la question principale du stage : peut-on prouver que la hiérarchie MSO est stricte par un langage dont les images sont bien proportionnées (linéairement ou polynomialement) ?

Cette restriction des structures étudiées aux langages d'images a un avantage important : depuis les travaux de Giammaresi et al. [3], il est connu que le niveau le plus bas de la hiérarchie MSO sur les images (celui avec des quantificateurs existentiels uniquement) correspond exactement à la classe des langages reconnaissables. Ce résultat nous permet d'étudier la hiérarchie sous un aspect combinatoire. Il est, à bien des égards, une extension du théorème de Buchi sur les langages de mots aux langages d'images. On trouve grâce à cette vision combinatoire un langage bien proportionné qui sépare le premier niveau de la hiérarchie MSO des niveaux supérieurs et apporte un début de réponse à la question d'une hiérarchie stricte sur les langages bien proportionnés.

Dans ce mémoire, tout d'abord, il sera proposé un résumé des résultats logiques connus sur les formules MSO sur les mots ainsi que sur les images ; nous définirons les notions nécessaires à la compréhension de la problématique du stage. Ensuite, nous étudierons les résultats combinatoires connus. Certains résultats de cette partie ont été développés dans le cadre du stage (partie 3.3). Avec tous ces outils en main, nous aborderons les travaux de Matz,Schweikardt et Thomas qui ont amené au théorème affirmant que la hiérarchie MSO est stricte. Nous aborderons aussi les particularités de la preuve de ce théorème. Enfin nous étudierons les opérations de pliage et de rembourrage d'image et l'impact que de telles opérations ont sur le niveau d'appartenance d'un langage dans la hiérarchie MSO ; ce chapitre a été développé dans le cadre du stage.

2 Hiérarchie MSO et langages d'images

Dans cette section, on présente les divers définitions nécessaires à la compréhension du sujet du stage ainsi que les résultats connus sur la logique MSO. On justifiera aussi le choix de la structure utilisée pour représenter les images.

2.1 Définitions

Je suppose que le lecteur connait les logiques du premier ordre (FO), du second ordre (SO), du second ordre monadique (MSO) et existentielle monadique du second ordre (EMSO). De même je suppose connue la notion de structure, la structure associée à un mot, la notion de langage. Si ce n'est pas le cas il peut se référer au livre de L.Libkin ([7]) pour l'ensemble de ces notions.

Définition 1 (Grille (de dimension 2))**.**
Une grille est une structure $(R, S_1, S_2, min_1, min_2, Max_1, Max_2)$ *d'ensemble de base R, avec* S_1/S_2 *des fonctions unaires et* min_i/Max_i *des relations unaires avec :*

- *R=[1...n]*[1...m] pour n,m deux entiers supérieurs à 1.*
- S_1 *:*
  - *((i,j)↦(i+1,j)) pour i∈[1..n-1] et j∈[1..m]*
  - *(n,j)↦(1,j) pour j∈[1..m]*
- S_2 *:*
  - *((i,j)↦(i,j+1)) pour i∈[1..n] et j∈[1..m-1]*
  - *(i,m)↦(i,1) pour i∈[1..n]*
- Max_1 *est composé des éléments de forme (n,j) pour j∈[1..m]*
- Max_2 *est composé des éléments de forme (i,m) pour i∈[1..n]*
- min_1 *est composé des éléments de forme (1,j) pour j∈[1..m]*
- min_2 *est composé des éléments de forme (i,1) pour i∈[1..n]*

Notons que l'on peut facilement définir des grilles de dimension supérieure à 2 : pour d dimensions, l'ensemble de base est un produit à d composantes et on ajoute 3d relations/fonctions (S_i, min_i, max_i pour i$\leq$d) qui chacune représente le successeur, les éléments minimum ou maximum sur la i^{eme} dimension.

On peut alors définir une image de manière informelle sur un alphabet Γ : une image de taille (n*m) sur un alphabet Γ est une fonction [1..n]*[1..m]$\rightarrow\Gamma$. Il existe plusieurs possibilités pour coder ceci par une structure logique. Ci-dessous, on donne une définition du codage par pixels des images, ce codage sera celui retenu pour l'ensemble du rapport. Dans la partie 2.4, nous présentons d'autres codages possibles et leur influence sur les résultats ainsi que les raisons qui nous ont amené à choisir la structure sur pixels.

Définition 2 (Images (de dimension 2))**.**
Soit Γ un alphabet. Une image p sur Γ est une structure $(R, (S_i)_{i=1,2}, (Q_s)_{s\in\Gamma}, (min_i)_{i=1,2}, (Max_i)_{i=1,2})$ telle que :

- $(R, S_1, S_2, min_1, min_2, Max_1, Max_2)$ *forme une grille.*
- $(Q_s)_{s\in\Gamma}$ *sont des relations unaires qui forment une partition du domaine R de la structure.*

Une image codée par pixels n'est jamais qu'une grille coloriée. Là encore, on peut généraliser ces définitions à des objets à d dimensions, une image d-dimensionnelle est un grille d-dimensionnelle coloriée.

On donne ensuite une définition des langages polynomialement/linéairement proportionnés, ce sont ces langages qui nous intéressent dans ce stage. Ces notions sont les deux choix possibles pour parler de langage (d'images) bien proportionné.

Définition 3 (Langage d'images polynomialement proportionné)**.** *Un langage d'images est dit polynomialement proportionné lorsqu'il existe un polynôme P(x) tel que toutes ses images sont de taille n*m avec m∈[n,P(n)] ou n∈[m,P(m)].*

Définition 4 (Langage d'images linéairement proportionné)**.** *Un langage d'images est dit linéairement proportionné lorsqu'il existe un entier c≥1 tel que toutes ses images sont de taille n*m avec m∈[n,c.n] ou n∈[m,c.m].*

Dans la logique du second ordre, une place importante est faite au nombre d'alternance de quantificateurs universels et existentiels. On verra dans la section 3 que les langages d'images EMSO-définissables ne sont pas clos par négation. Pour différencier deux formules MSO en fonction de leurs capacités descriptives respectives, on a établi une hiérarchie basée sur le nombre d'alternances de quantificateurs du second ordre.

Définition 5 (Hiérarchie MSO)**.**
On définit la hiérarchie par alternance de quantificateurs de MSO par induction comme suit :

- Σ_1=*EMSO*
- Π_1=$\{F | F = \neg G,\ G \in \Sigma_1\}$
- Σ_k=$\{F | F = \exists X_1 \ldots \exists X_n G,\ G \in \Pi_{k-1}\}$
- Π_k=$\{F | F = \neg G,\ G \in \Sigma_k\}$

On dira d'un langage qu'il est dans Σ_k s'il est défini par une formule Σ_k. On a les propriétés suivantes :

Proposition 1. *L'union/intersection de deux langages Σ_k est Σ_k.*

Démonstration. On ne donne qu'une idée de la preuve :
Soit F et G deux formules Σ_k définissant deux langages A et B $\in \Sigma_k$, alors F$\wedge$G définit A$\cap$B ; on suppose F et G sans variable communes (si besoin, il suffit de renommer ces variables). Or F$\wedge$G est équivalente à une formule Σ_k, il suffit de rentrer l'une dans l'autre comme dans l'exemple ci-dessous.

F = $\exists X_1 \exists X_2 \forall X_3 H$ et G = $\exists Y_1 \forall Y_2 \forall Y_3 H'$ avec H/H' $\in$ FO.
Alors F$\wedge$G est équivalent à $\exists X_1 \exists X_2 \exists Y_1 \forall X_3 \forall Y_2 \forall Y_3 \ H \wedge H'$ □

2.2 Logique MSO et langages de mots

Dans cette partie, nous faisons un court état de l'art de la logique MSO sur les langages de mots. Ceux-ci sont intéressants puisque les images ne sont qu'une généralisation des mots sur plusieurs dimensions et certains résultats développés dans le cadre des mots passent aux images.

On commence par le théorème de Buchi qui donne une caractérisation simple et facilement manipulable des langages de mots définissables dans MSO :

Théorème 1 (Buchi)**.**
Un langage sur les mots est régulier (reconnu par automate fini) si et seulement si il est définissable dans MSO.

La preuve de ce théorème a mis en évidence la conséquence que, sur la classe des mots, tout énoncé MSO est équivalent à un énoncé EMSO. Thomas dans [14] montre que l'on peut faire mieux, on peut retrouver les grandes lignes de la preuve dans [9] :

Théorème 2. *Sur les mots, tout énoncé MSO est équivalent à un énoncé EMSO avec une seule quantification existentielle du second ordre.*

2.3 Logique MSO et langages d'images

On expose ici quelques résultats sur la logique MSO sur la classe des images (simplification des formules, forme normale).
On retrouve la preuve du résultat ci-dessous dans [8].

Théorème 3.
Tout énoncé sur EMSO sur les images est équivalent à un énoncé EMSO avec une seule quantification existentielle du second ordre.

On a d'autres résultats de ce type dans [5] / [4] :

Théorème 4.
EMSO sur les images = EMSO($\forall$ 1)
Avec EMSO($\forall$ 1) : ensemble des énoncés EMSO dont les quantifications de la partie du premier ordre sont réduites à une seule quantification universelle.

On peut combiner les deux résultats, en partant d'une formule EMSO, on la transforme en une formule EMSO($\forall$1) par le procédé décrit dans [5] puis on la transforme en une formule EMSO($\forall$1) avec une seule quantification existentielle du second ordre comme dans la preuve du théorème 3. Il faut juste effectuer les transformations dans le bon ordre.

Théorème 5. *Tout énoncé sur EMSO sur les images est équivalent à un énoncé EMSO($\forall$1) avec une seule quantification existentielle du second ordre.*

2.4 Choix de la structure logique

La partie précédente ayant présenté les résultats logiques sur les images codées par pixel, nous pouvons évoquer maintenant d'autres codages possibles et leurs avantages/inconvénients. Ces différences dans le codage ont deux influences : une sur les résultats logiques et l'autre dans le cadre d'une implémentation informatique de la notion d'image : certains codages apparaissent comme plus naturels que d'autres. Ci-dessous deux autres codages possibles :

Définition 6 (Otto-grille, Otto-image)**.**
Une Otto-grille est une structure (E,R,C) où E=[1..n][1..m] est l'ensemble de base (où n,m sont 2 entiers fixés) et R et C sont deux relations binaires telles que :*

- *((i,j),(k,l))$\in$C ssi j=l, c'est-à-dire si (i,j) et (k,l) sont sur la même colonne,*
- *((i,j),(k,l))$\in$R ssi i=k, c'est-à-dire si (i,j) et (k,l) sont sur la même ligne.*

On définit les Otto-images comme des Otto-grilles coloriées.

Il est important de noter que les relations C et R sont des relations binaires globales et non locales au sens où elles donnent des informations sur des pixels arbitrairement éloignés (dans le respect de la taille de l'image considérée). Pour ces raisons, il est possible d'argumenter sur le fait que les Otto-images ne sont pas vraiment des images au sens de la localité : d'un point de vue logique, ajouter une relation globale/non locale dans la structure semble trop expressif.

La notion d'Otto-image peut être appropriée pour une implémentation informatique : en effet compte tenu de la manière dont sont représentés les tableaux/grilles en informatique, il n'est pas irréaliste de prétendre pouvoir savoir si 2 éléments d'un tableau sont sur la même ligne/colonne. Ce codage se démarque aussi du codage par pixel d'un point de vue logique, comme prouvé dans [11] :

Proposition 2. *Dans EMSO, sur la classe des Otto-images, il existe une hiérarchie stricte basée sur le nombre de quantificateurs existentiels autorisés : il existe un langage d'Otto-images définissable par une formule de type $\exists X_1, ..\exists X_{k+1}$ F (où F $\in$ FO) mais pas par une formule de type $\exists X_1, ..\exists X_k$ G (où G $\in$ FO) où X_i sont des variables du second ordre.*

Ce résultat n'est pas vrai sur la classe des images codées par pixel (cf théorème 3 partie 2.3).

Un autre codage est celui par coordonnées :

Définition 7 ("Coordinate" structures)**.**
Une image sur un alphabet Γ *encodé par coordonnées est une structure ([1..n],*$(R_s)_{s\in\Gamma}$*,<,min,Max,S) telle que :*

- *L'ensemble de base est [1..n].*
- *Pour* $s \in \Gamma$*,* R_s *est une relation binaire telle que (a,b)* $\in R_s$ *dit que le pixel de coordonnée (a,b) porte la couleur s.*
- $<$ *est une relation d'arité 2 représentant l'ordre classique sur [1..n].*
- *min=*$\{1\}$ *et Max=*$\{n\}$
- *S est la fonction successeur sur [1..n].*

Notons que cette définition peut s'étendre aux images de dimension supérieure à deux. De plus, le problème d'avoir la largeur égale à la longueur peut être contourné en ajoutant une lettre de remplissage à l'alphabet.
Comme montré dans [5] (théorème 10.6), un tel codage ne correspond pas à une étude par les aspects de reconnaissabilité puisque les langages reconnaissables d'images d-dimensionnelles (pour d $\geq$2) ne correspondent pas exactement à une classe logique (ces aspects sont développés dans le cadre du codage par pixels dans la section 3). Ceci en fait la principale raison pour laquelle, dans ce stage, ce codage n'est pas adopté pour représenter les images.

À l'inverse, ce codage présente un intérêt pour une étude de la complexité descriptive des automates cellulaires non déterministes à une ou plusieurs dimensions (cf [5] et [4]).

3 EMSO et reconnaissabilité

Pour traiter ces questions logiques, il est possible d'adopter un point de vue combinatoire grâce au théorème de Giammarresi et Restivo [3] ; pour cela plusieurs définitions sont nécessaires :
Soit p une image n*m sur un alphabet Σ, on appelle image bordée de p, noté $\hat{p}$, l'image (n+2)*(m+2) sur $\Sigma \cup \{\#\}$ obtenue en entourant p du symbole $\#$.

$$\text{p=}\begin{pmatrix} p_{1,1} & \cdots & p_{1,m} \\ \vdots & & \vdots \\ p_{n,1} & \cdots & p_{m,m} \end{pmatrix} \mapsto \hat{p} = \begin{pmatrix} \# & \# & \# & \# & \# \\ \# & p_{1,1} & \cdots & p_{1,m} & \# \\ \# & \vdots & & \vdots & \# \\ \# & p_{n,1} & \cdots & p_{m,m} & \# \\ \# & \# & \# & \# & \# \end{pmatrix}$$

Pour travailler sur l'aspect combinatoire des langages d'images, il est nécessaire d'avoir une manière combinatoire de reconnaitre les bords d'une image (et non un moyen logique comme les relations min/Max). Pour cela, on entoure l'image d'un symbole spécifique avant de travailler dessus. Pour un langage L, on définit $\hat{L}$ par $\{ \hat{p} \mid p \in L \}$.

Définition 8 (Langage a*b-local)**.**
Pour des entiers a,b fixés. Un langage L sur un alphabet Σ *est a*b local s'il existe un*

*ensemble Δ de motifs rectangulaires de taille a*b sur $\Sigma \cup \{\#\}$ tel que les images de $\hat{L}$ sont exactement celles dont toutes les sous-images de taille a*b sont dans Δ.*
*On parlera de langage k-local en lieu de k*k-local.*

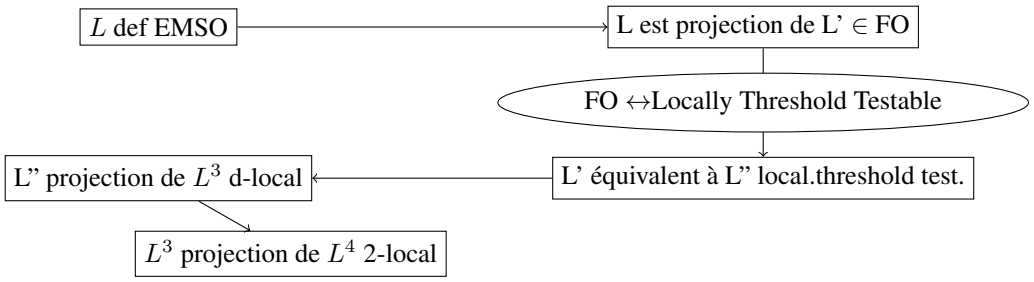
Définition 9 (Langage reconnaissable = REC)**.**
*Un langage L défini sur un alphabet Σ est reconnaissable s'il existe un langage 2*2-local M défini sur un alphabet $\Sigma * \Gamma$ tel que L=$\{\alpha(p)|p \in M\}$ où $\alpha : \Sigma * \Gamma \rightarrow \Sigma$ est la projection sur la première coordonnée.*

3.1 EMSO=REC

Théorème 6 ([3])**.** *Pour tout langage d'images L : L$\in$ REC ssi L$\in$ EMSO.*

Le théorème ci-dessus est semblable au théorème de Buchi mais appliqué aux langages d'images : sur les mots, un langage est reconnaissable s'il est projection d'un langage 1*2 local au sens défini avant, ie s'il existe un ensemble Δ tel que le langage est définit comme l'ensemble des mots dont tout sous-mot de taille 2 est dans Δ. Il est facile de voir que la classe des langages de mots reconnaissables est exactement la classe des langages réguliers.

Démonstration. Un sens de la preuve est du codage : on crée une formule EMSO qui code à la fois la projection et le système de tuiles avant projection.
L'autre sens peut être résumé à travers la figure ci-dessous :



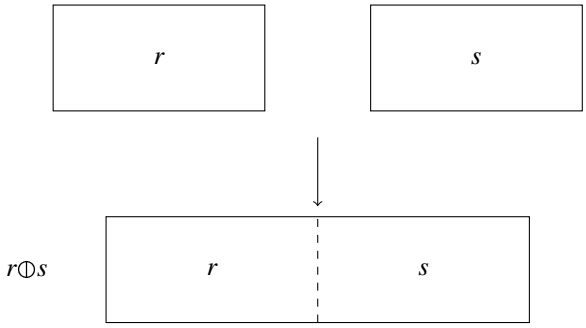
Donc L projection d'un langage local et donc $\in$REC

La notion de "Locally Threshold Testable" peut être trouvée dans [3]. Une étude des différents types de localité est proposée à la fin de cette section et reviendra sur la notion de langage local. Notons que ce résultat se généralise aux images de dimensions quelconques (cf [5]), la preuve à l'inverse de celle ci-dessus aborde un point de vue plus logique que combinatoire. Ce résultat est le fondement d'une étude basée sur des arguments de reconnaissabilité. □

3.2 Propriétés des langages reconnaissables

On présente ici plusieurs résultats sur les langages reconnaissables, des preuves combinatoires sont données dans [3] et [12].

Définition 10 ($\oplus$: concaténation par colonnes d'images)**.**
*Soit r,s deux images sur un même alphabet Σ, de taille n*m et n*m' (donc avec le même nombre de lignes). On définit r$\oplus$s comme la concaténation des images r et s en une image n*(m+m').*



De même, on définit $\ominus$ comme la concaténation par lignes de deux images qui ont le même nombre de colonnes. On peut alors définir la concaténation par colonnes ou par lignes de langages :

Définition 11 ($\oplus$: concaténation par colonnes de langages)**.**
$L_1 \oplus L_2$=$\{$ *x*$\oplus$*y* $\mid$ *x*$\in L_1$, *y*$\in L_2$ *et x,y ont le même nombre de lignes*$\}$

On peut de même définir la concaténation de deux langages par lignes. Alors on a les résultats suivants :

Proposition 3.

- *Soit A et B deux langages reconnaissables, alors A*$\ominus$*B et A*$\oplus$*B sont reconnaissables.*
- *Soit L un langage reconnaissable, alors* $L^{*\oplus}$=$\bigcup_{i\in\mathbb{N}}$ *L*$\oplus$*L...*$\oplus$*L (i fois) est aussi reconnaissable.*
- *De même,* $L^{*\ominus}$=$\bigcup_{i\in\mathbb{N}}$ *L*$\ominus$*L...*$\ominus$*L (i fois) est aussi reconnaissable.*

La famille des langages reconnaissables est aussi close par union et intersection, mais pas par complémentation (théorèmes 7.4 et 7.5 dans [12]) :

Proposition 4. *Soit A et B deux langages reconnaissables, alors A*$\cup$*B et A*$\cap$*B sont reconnaissables.*

Proposition 5. *Il existe un langage L reconnaissable tel que son complémentaire* $\overline{L}$ *n'est pas reconnaissable.*

Soit le langage suivant MIROIR=$\{p \ominus p \parallel$ p est une image carrée sur Σ $\}$ sur un alphabet quelconque avec au moins deux éléments Σ. L est son complémentaire. Montrons donc que MIROIR n'est pas reconnaissable mais est le complémentaire d'un reconnaissable. S'il existe un langage local M sur un alphabet $\Sigma * \Gamma$ et une projection α telle que α(M)=MIROIR, alors pour deux images sur $\Sigma * \Gamma$ p,q n*n, différentes, de même contour et telles que $\alpha(p) \neq \alpha(q)$, $p \ominus q$ serait dans M et $\alpha(p \ominus q)$ serait

dans MIROIR, or c'est impossible. C'est un argument de comptage basé sur le fait que asymptotiquement il y a plus d'images n*n sur Σ que de contour d'images n*n sur $\Sigma * \Gamma$. Pour la preuve que le complémentaire de MIROIR est reconnaissable, voir [12]

Ce dernier résultat en combinaison avec le fait que REC=EMSO a pour implication $\Sigma_1 \neq \Pi_1$. Il est à noter que le langage MIROIR obtenu est polynomialement proportionné (même linéairement proportionné) et que l'on peut le modifier de sorte d'obtenir un langage carré (développé par la suite). En rappelant que $\Pi_1 \subseteq \Sigma_2$, on obtient un langage séparateur qui est dans Σ_2 mais pas dans Σ_1 et de taille voulue. On dispose ainsi d'une séparation effective entre le premier et le deuxième niveau de la hiérarchie MSO sur les langages d'images polynomialement proportionnées.

3.3 Degrés de liberté dans la notion de localité

Puisque les langages Σ_1 sont ceux qui sont projections de langages locaux, il se pose une question sur la souplesse de la définition de la localité. Il existe de nombreuses définitions possibles pour aborder la notion de localité en combinatoire :
La première est celle par motifs interdits : un langage est dit local par motif interdits si il existe un ensemble Δ finit de motifs (éventuellement de différentes tailles) tel que les images du langage sont exactement celles qui ne possèdent aucun des motifs de Δ. Borchert (Lemme 5.1 [2]) a montré que ces langages sont exactement ceux qui peuvent être décrits par une formule du premier ordre avec un seul quantificateur universel, appelée formule FO($\forall$ 1).

Cependant la notion de motif interdit n'est pas toujours pratique au sens où on peut interdire en même temps un motif 2*3, un motif 5*7 et un motif en escalier ou formant un cercle composé de 50 cases, de plus une telle vision n'est pas très constructive. Pour définir la localité, il est plus naturel de fixer une taille a*b et ensuite de dire quels sont les rectangles autorisés de taille a*b et ainsi créer une image par juxtaposition de conditions locales. Cette version est à la fois constructive et met en évidence les formes de calcul simple qu'autorise tel ou tel langage (ces calculs sont moins apparents si l'on adopte la version de localité par motifs interdits). Latteux et Simplot dans [6] ont étudié la localité par langage dominos-local, ie avec des rectangles de taille 1*2 et 2*1 (définition ci-dessous). Giammaressi et Restivo ont choisit des langages locaux par rectangles 2*2 [12]. Une localité par dominos 2*1 et 1*2 peut être retranscrite dans une localité 2*2, l'inverse n'est pas vrai (ce point est rappelé au point 1 du théorème ci dessous). Je présente dans le théorème ci dessous certains résultats sur les autres tailles (le 1er point est tiré de l'article de Latteux et Simplot [6], le reste à été développé dans le cadre du stage).

Définition 12 (Langage domino-local)**.** *Un langage d'images L sur Γ est domino-local s'il existe deux ensembles de dominos Δ_{2*1} et Δ_{1*2} sur $\Gamma \cup \{\#\}$ tels qu'une image p est dans L si et seulement si $\hat{p}$ vérifie :*

- *Toutes les sous-images 2*1 sont dans Δ_{2*1}.*
- *Toutes les sous-images 1*2 sont dans Δ_{1*2}.*

Je présente dans le théorème ci dessous certains résultats sur les autres tailles (le 1er point est tiré de l'article de Latteux et Simplot [6], le reste à été développé dans le cadre du stage).

Théorème 7.

1. *Tout langage domino-local est 2-local et il existe un langage 2-local qui n'est pas domino-local.*
2. *Pour tout k$\geq$ 2 : il existe un langage k-local qui n'est pas k+1 local.*
3. *Pour tout k$\geq$ 2, si on restreint la classe des images aux images dont les deux cotés sont de taille $\geq$ k+1, tout langage k-local est k+1 local.*
4. *Pour tout k$\geq$ 3 , il existe un langage k+1 local qui n'est pas k-local.*
5. *Pour tout k$\geq$ 2 , tout langage k-local est définissable dans FO($\forall$1).*
6. *$\bigcup_{k\geq 2}$ k-local $\subsetneq$ FO($\forall$ 1). De plus, en ajoutant/supprimant un nombre fini d'images, tout langage définissable dans FO($\forall$ 1) peut être transformé en un langage qui est $\bigcup_{k\geq 2}$ k*k-local.*

Démonstration. **Point 1 : Tout langage domino-local est 2-local et il existe un langage 2-local qui n'est pas domino-local.**

La première partie de ce point est montré dans les articles de Giammaressi et Restivo [12] et de Latteux et Simplot [6]. La construction du pavage est facile et expliquée ci-dessous :
Soit L un langage domino-local et Δ_{2*1} et Δ_{1*2} les deux ensembles de dominos considérés. On définit Δ_2 l'ensemble de motifs 2*2 suivant par :

$$\Delta_2 \ = \ \{\begin{pmatrix} a & b \\ c & d \end{pmatrix} \ | \ (a,b),(c,d) \in \Delta_{1*2} \ et \ (a,c),(b,d) \in \Delta_{2*1}\}$$

Soit L' le langage 2-local défini par Δ_2. Montrons que l'on a L = L' :

Soit p $\in$ L et une sous-image $\begin{matrix} a & b \\ c & d \end{matrix}$ 2*2 de son image bordée p̂ -cf intro de la section pour cette notation-. Puisque p $\in$ L,on a (a,b) et (c,d) $\in$ Δ_{1*2} et $(a,c),(b,d) \in$ Δ_{2*1}. Cela implique que $\begin{matrix} a & b \\ c & d \end{matrix}$ est dans Δ_2, donc que p̂ est accepté par Δ_2 et donc p $\in$ L'.

Inversement, pour p $\in$ L', toute sous-image 1*2 (a,b) de p̂ est issue d'un motif 2*2 $\begin{matrix} a & b \\ c & d \end{matrix}$ qui est dans Δ_2 et, par définition de Δ_2, (a,b) est dans Δ_{2*1}. De même pour les sous-images 2*1. Donc les sous-images 1*2 de p̂ appartiennent bien à Δ_{1*2} et les sous-images 2*1 à Δ_{2*1}. Ceci montre que p̂ est accepté par $\Delta_{1*2} \cup \Delta_{2*1}$ et donc que p $\in$ L.

Donc tout langage domino-local est 2-local. Définissons un langage 2-local qui n'est pas domino-local sur Σ=$\{0,1\}$:

Δ_2 = tous les motifs 2*2 possibles et structurellement corrects (vis à vis de $\#$) sur $\Sigma \cup \{\#\}$, sauf $\begin{matrix} 1 & 1 \\ 1 & 0 \end{matrix}$.

À supposer que L(Δ_2) est domino-local, l'ensemble des dominos horizontaux doit autoriser les motifs 1 1 et 1 0 (il est facile de créer une image où chacune de ces configurations se retrouve). De même, les motifs $\begin{matrix} 1 \\ 1 \end{matrix}$ et $\begin{matrix} 1 \\ 0 \end{matrix}$ sont autorisés verticalement. Dans ce cas, l'image $\begin{matrix} 1 & 1 \\ 1 & 0 \end{matrix}$ doit être acceptée par l'union des deux ensembles de dominos, cependant cette image n'est pas acceptée par Δ_2. On obtient une contradiction, ainsi $L(\Delta_2)$ n'est pas domino-local.

Point 2 : Pour tout k$\geq$2 il existe un langage k-local qui n'est pas k+1-local

Les langages dont certaines images sont complètement définies par un unique motif k*k ne sont pas k+1-local. Par exemple pour k=3, voici un motif qui engendre une image qui ne sera jamais accepté par un ensemble de motifs 4*4 : Δ = $\{ \begin{matrix} \# & \# & \# \\ \# & 1 & \# \\ \# & \# & \# \end{matrix} \}$

Ainsi la présence de l'image composée seulement d'un pixel 1 dans un langage empêche ce dernier d'être 4-local. Cependant, ces exceptions ne sont qu'en nombre fini,et on a une inclusion pour les images de longueur et de largeur $\geq$k+1 comme expliqué au point 3.

Point 3 : Sur les langages ne contenant pas d'image avec un bord de taille k, k-local $\rightarrow$ k+1 local

Soit L un langage k-local défini par Δ_k, on suppose que L ne contient pas d'image dont un des bords est de longueur $\leq$ k (pour ne pas retomber sur un contre-exemple comme au point 2).
Montrons qu'il existe Δ_{k+1} ensemble de motifs k+1*k+1 qui définit L. La construction exposée au point 1 peut être reprise ici : on autorise un motif k+1*k+1 si tous les sous-motifs k*k sont dans Δ_k. On note L' le langage déterminé par Δ_{k+1}.
Avec les hypothèses posées au début sur L, toute image de L admet au moins un motif k+1*k+1. La vérification de L=L' est similaire à celle du point 1.

Les images qui posent problème au point 2 (k-local mais non k+1-local) ne sont pas représentatives au sens où elles sont décrites par un unique motif.

J'ai pensé à une modification de la définition de k-local qui exclurait les motifs qui ont 2 bordures opposées (par exemple deux rotations de tuiles comme $\begin{matrix} \# & \# \\ \# & ? \end{matrix}$). Ce-

pendant dans ce cas, il existera quand même des langages k-local qui ne seront pas k+1 locaux (par exemple ceux dont des images possédent un des bords de longueur k+1) et on retombe avec le même type de contre-exemple. Ainsi le langage avec une seule image (celle ci-dessous) serait k-local mais pas k+1-local relativement à la nouvelle définition.

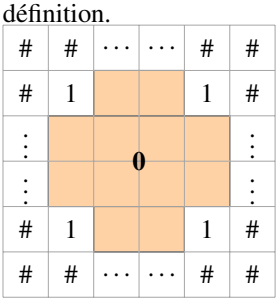


image k+1 * k+1 avec partie orange en 0.

Ce problème ne semble pouvoir être réglé par un changement de définition mais n'a pas d'importance dans l'étude asymptotique des langages locaux : le problème est cantonné aux images k*k.

Une conséquence des points 2 et 3 est que tout langage k-local peut être modifié par un nombre fini de suppressions d'images pour devenir k+1-local.

Point 4 : Pour tout k $\in \mathbb{N}$, il existe un langage k+1-local qui n'est pas k-local

L'idée est une généralisation d'un contre-exemple dans le contexte des mots. Premièrement, décrivons un langage de mots k-local qui n'est pas k-1 local.

On donne l'expression régulière qui définit le langage : L=$\{\ 0^k\ (10^k)^*\ \}$. Les motifs (mots de longueur k+1) qui décrivent le langage L sont évidents. Le langage L n'est pas k-local, tout système de motifs de taille k qui définirait L devrait accepter le motif 0^k, alors ce motif peut être répété un nombre arbitraire de fois d'affilée, ce qui n'est pas autorisé dans la définition de L.

La généralisation aux langages d'images peut être faite en créant une image dont la première ligne est dans L et le reste est à 0.

Point 5 : Pour tout k $\in \mathbb{N}$, tout langage k-local est FO($\forall$1) définissable

Je suppose pour plus de simplicité que l'on veut définir l'image avec les frontières #, dans le cas contraire il faut changer légèrement le procédé mais cela ne change pas le principe.

Le codage de la k-localité par FO($\forall$1) est légèrement fastidieux à cause des cases proches du bord inférieur et du bord droit qui ne peuvent être traitées de la même manière que les autres.

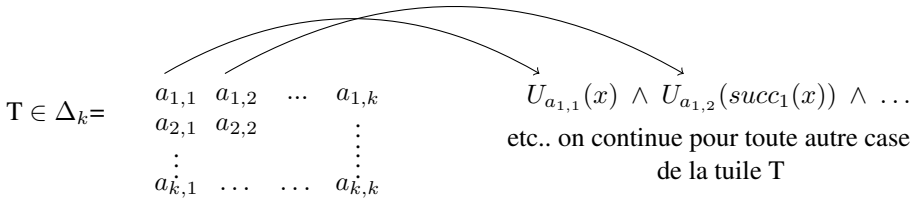
Soit L_k un langage k-local et Δ_k l'ensemble de motifs k*k qui le définit.

Soit F la formule qui code Δ_k :

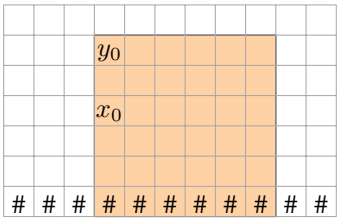
$$\forall x \left((\neg G_{Bas}(x) \wedge \neg G_{Droite}(x)) \rightarrow \bigvee_{T \in \Delta_k} G_T(x) \right)$$

avec G_{Bas}(x) qui nous dit si l'un des $succ_2^i$(x) est dans Max_2 pour i ∈ [0..k-1], autrement dit si x est à distance k-1 ou moins du bord inférieur (ie si le symbole # apparait autre part que sur la k-eme ligne/colonne). Similairement G_{Droite}(x) dit si x est à distance k-1 ou moins du bord droit de l'image.

Par ailleurs, G_T(x) décrit le comportement du motif T : G_T est une conjonction de formules atomiques sur la variable x qui chacune décrit le symbole d'une case du motif. Pour celà on considère la position en haut à gauche comme la position 'x' et on accède aux autres positions par les fonctions succ$_i$:



Il n'est pas nécessaire de décrire le comportement des pixels en bas ou à droite -à moins de k cases du bord- (sur le dessin ci-dessous, x_0 par exemple) car elles font partie du voisinage inférieur droit d'un point y_0 qui est suffisamment loin du bord ; et la formule F décrit le comportement du voisinnage inferieur droit de y_0.



Orange : voisinage inférieur droit de y_0

x_0 est déterminé par y_0 et les tuiles de bord inférieur qu'autorise Δ_k

Reste à vérifier que F décrit exactement le langage décrit par Δ_k :

Notons L_k le langage k-local défini par Δ_k et L' le langage défini par la formule F décrite ci-dessus.
Soit p∈ L_k, montrons que p∈ L'. Pour tout pixel (i,j) à distance ≥ k du bord droit et du bord inférieur, le motif k*k dont (i,j) est le pixel en haut à droite est dans Δ_k (par définition d'un langage local). On note T_0 ce motif. Par construction de F, (i,j) vérifie G_{T_0}. Par conséquent, p vérifie F. Ainsi $L_k \subseteq L'$.

Soit p ∈ L', montrons que p ∈ L_k. Supposons le contraire : dans ce cas il existe P un pavé k*k dans l'image p, qui n'est pas dans Δ_k (ce pavé n'est pas un motif de Δ_k). On note x_0=(i,j) le pixel haut gauche du pavé, x_0 est à distance ≥k des bords inférieur et droit puisqu'il est le premier pixel d'un pavé k*k, cependant il ne vérifie aucune des

formules G_T (pour T$\in \Delta_k$) -par hypothèse -, par conséquent p ne satisfait pas F, ce qui est une contradiction avec p$\in L'$.

Point 6 : $\bigcup_{k\in\mathbb{N}}$ Langages k-local $\subset$ FO($\forall$ 1). De plus cette inclusion est stricte.

L'inclusion découle du point 5.
Il reste à montrer que l'inclusion est stricte. Il est clairement possible de définir une image fixée avec une formule FO($\forall$ 1). De même, il est possible d'interdire certaines images ou tailles d'images (par exemple interdire les images 5*5) par une formule de FO($\forall$1). Cela permet de définir des langages locaux avec exceptions (ajout d'images particulières, interdiction d'images de certaines tailles, ajout de toutes les images de certaines tailles etc). Cette possibilité de gérer les exceptions dans FO n'est pas possible à travers $\bigcup_{k\in\mathbb{N}}$ Langages k-local . On peut considérer par exemple :

Soit p une image k*k quelconque décrite par une formule de type $\forall$x H avec H sans quantificateur. La formule H est une disjonction de formules $(H'_i)_{i\in k*k}$ qui chacune décrit l'image p en la balayant et en commencant par un pixel fixé. Soit L_0 le langage k+1 local mais pas k local défini au point 4 et $\forall$xF(x) la formule FO($\forall$1) qui décrit L_0. Soit L_1=$\{p\} \cup L_0$, il est facile de vérifer que L_1 est définit par la formule $\forall$x (F(x)$\vee$H), ainsi L_1 est définissable dans FO($\forall$ 1). Notons qu'une image ne peut vérifier pour certains pixels H et pour d'autres F et ne pas être dans L_1 : si une image q a un de ses pixels qui vérifie H, elle vérifie une des sous-formules H'_{i_0} et donc q=p et q$\in L_1$.
Pour tout k$\in \mathbb{N}$, L_1 n'est pas k-local : L_1 contient une image k*k donc pour tout i$>$0, L_1 ne peut pas être k+i local ; de plus, pour tout i$\leq$k, il ne peut être i local puisqu'il contient L_0 (l'argument est le même qu'au point 4). Donc $L_1 \notin \bigcup_{k\in\mathbb{N}}$ Langages k-local.

L'impossibilité de gérer certaines exceptions (d'images ou de tailles) est une des raisons (peut-être en existe-il d'autres) qui sépare FO($\forall$ 1) des langages locaux définis par motifs de taille fixé.

Remarque : il est important de noter que le point 6 n'est pas en contradiction avec les résultat de l'introduction de cette partie. Il ne faut pas voir $\bigcup_{k\in\mathbb{N}}$ Langages k-local comme une version équivalente des langages locaux par motifs interdits (définis dans l'introduction de cette partie) puisque dans un cas, on choisit une taille puis on choisit les motifs dans cette taille alors que dans l'autre cas les motifs peuvent varier en taille. La logique FO($\forall$1) permet de décrire dans une même formule des comportements sur des voisinages de diverses tailles, de même que pour les langages locaux par motifs interdits ; à l'inverse $\bigcup_{k\in\mathbb{N}}$ Langages k-local fixe une taille de voisinage et ensuite décrit les comportement autorisés pour des voisinages de cette taille donnée.
Il est à noter que là aussi seul un nombre fini d'images pose problème pour qu'un langage k-local puisse être définissable dans FO($\forall$1). C'est le même principe qu'au point 2. □

Puisque l'on sait que REC=Σ_1, on obtient le théorème suivant.

Théorème 8. *Les langages issus de la projection d'un langage domino-local sont exac-*

tement ceux issus de la projection d'un langage k-local ($\forall k$) et exactement ceux issus de la projection d'un langage FO($\forall 1$) définissable.

Démonstration. Comme précisé dans la première partie de cette section, EMSO=REC. De plus Latteux et Simplot ont montré dans [6] que proj-dominos-local = proj-2-local. Un langage obtenu par projection d'un langage FO($\forall$ 1) est définissable dans EMSO, on sait de plus que EMSO=EMSO($\forall 1$)=REC. Or tout langage k-local est FO($\forall$ 1) définissable, donc tout langage obtenu par projection d'un langage k-local est projection d'un langage FO($\forall$ 1) définissable, donc EMSO définissable et est donc reconnaissable (projection d'un langage 2-local). □

Ainsi pour un langage L $\in$ EMSO, on peut supposer qu'il est issu de la projection d'un langage k-local et ce pour tout k. Cela permet de choisir sur quelle localité nous souhaitons travailler lorsqu'on s'intéresse au langage "parent" local (ie : avant projection) d'un langage EMSO, cela aura un intérêt notamment dans la section 5 sur le pliage et le rembourrage des langages d'images.

J'ai choisi de travailler sur des motifs de formats carrés pour plus de simplicité, notons que l'on peut aussi travailler de manière similaire sur des formats rectangulaires et obtenir les mêmes résultats : "Etre projection d'un langage local par pavé 2*2 est équivalent à être projection d'un langage local par pavés 2*2.c -c$\geq$ 1-", on peut par exemple montrer qu'un langage 2*2.c local est à un nombre fini d'images près, 2.c*2.c local et ensuite utiliser le résultat ci-dessus. La seule condition sur le format de rectangle choisi est que longueur et largeur doivent être plus grandes que 1. De même on peut travailler avec 2 ensembles de rectangles (comme pour les dominos 1*2 et 2*1) pourvu qu'ils ne soient pas de type 1*a et 1*b ou de type a*1 et b*1. Là encore la projection écrase toutes ces nuances.

4 Hiérarchie MSO sur les images quelconques

Comme dit en introduction, on sait déjà prouver que la hiérarchie MSO sur les langages d'images est stricte. On rappelle ici les éléments principaux de la preuve et on expliquera pourquoi cette preuve, bien qu'ingénieuse, n'est pas complètement satisfaisante et ce sous plusieurs aspects.

La preuve que la hiérarchie Σ_k pour k$\geq$1 de MSO sur langages d'images quelconques est stricte est due à Matz et Thomas [10] et a été reprise par N.Schweikardt [13]. Ils se sont concentrés sur un type de langage particulier : les langages définissant des fonctions et sur le taux de croissance des fonctions autorisées à un certain niveau de la hiérarchie MSO. Dans un premier temps ils montrent que pour Σ_k, le taux de croissance de telles fonctions est borné par une tour d'exponentielles dont la hauteur dépend de k, ensuite ils montrent que l'on peut définir une fonction dans Σ_{2*k+3} dont le taux de croissance est plus grand que la borne trouvée (nous remplacerons ce deuxième point par un résultat plus précis de N.Schweikardt [13]).
Certaines définitions sont nécessaires :

Définition 13.
*On dit d'un langage d'images L d'alphabet Σ qu'il définit une fonction f : n$\mapsto$ f(n) lorsque pour toute image p sur Σ, on a : p est dans L si et seulement si p est de taille n*f(n).*

Si L est définissable dans Σ_k, on dira que la fonction f est définissable dans Σ_k. Dans le cas où k=1 on parlera de fonction reconnaissable.

Par exemple l'ensemble des images carrées sur un alphabet quelconque représente la fonction identité. Notons que l'alphabet Σ sur lequel est défini le langage n'a pas d'importance et que le contenu d'une image n'a pas d'influence sur l'appartenance au langage, seule la taille de l'image importe. On définira donc souvent les langages représentant des fonctions sur un alphabet singleton (à une seule lettre). Dans le reste de cette section, tout langage représentant une fonction sera défini sur un alphabet singleton. On dira qu'une fonction est définissable dans Σ_k pour k$\geq$1 lorsque le langage qui la représente est définissable dans Σ_k.

4.1 Croissance des fonctions définissables dans Σ_k

On définit d'abord deux familles de fonctions, ensuite on borne le taux de croissance des fonctions Σ_k définissables :

Définition 14.
$f_1(i) = 2^i$, $f_{n+1}(i) = f_n(i)\ 2^{f_n(i)}$
$s_0(i) = i$, $s_{n+1}(i) = 2^{s_n(i)}$

Théorème 9 ([10])**.**
Toute fonction f(x) Σ_n-définissable est $s_n(O(x))$

On trouve deux preuves de ce théorème, l'une (dans [10]) basée sur les automates et les langages de mots (bien que le résultat soit sur les images), l'autre (dans [13]) basée sur les jeux d'Ehrenfeucht-Fraissé. Ce résultat n'est pas surprenant compte tenu des propriétés des langages reconnaissables : c'est une généralisation aux niveaux supérieurs de la hiérarchie MSO du corollaire 10.1 de [12] du lemme d'itération des langages reconnaissables (9.1, [12]). On rappelle ici le corollaire :

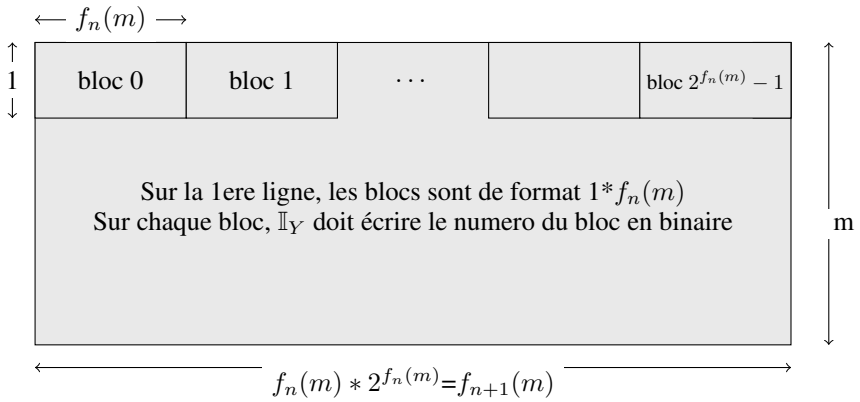
Corollaire 1. *Toute fonction qui croît plus rapidement que toute fonction exponentielle ne peut être reconnaissable (=Σ_1 définissable).*

4.2 Témoin de séparation

Théorème 10.
$\forall n \in \mathbb{N}$, *la fonction x*$\mapsto$ $f_n(x)$ *est Σ_n-définissable.*

On donne l'idée principale de la preuve qui se fait par induction, pour la preuve détaillée se référer au théorème 8 de [13].
L'étape initiale peut être faite avec le point de vue logique comme le point de vue combinatoire (cf corollaire 10.1 [12]). Pour l'étape d'induction, on a $f_n(m)*2^{f_n(m)}$=$f_{n+1}(m)$,

on va décomposer la première ligne de l'image $m * f_{n+1}(m)$ en blocs de taille $f_n(m)$ grâce à des quantifications du second ordre (noté Y sur le dessin ci-dessous). La formule qui force les blocs à être de bonne taille pour f_{n+1} est basée sur celle qui force les blocs à être de bonne taille pour f_n et augmente d'un niveau dans la hiérarchie.



Plusieurs points sont important dans cette démonstration : l'alphabet sur lequel est définie f_{n+1} est unaire, le découpage en blocs est fait par des quantifications du second ordre.

4.3 Spécificité du témoin de séparation

Plusieurs remarques sont à faire sur ce résultat :

– La borne supérieure du taux de croissance des fonctions Σ_k définissables est basée sur plusieurs points essentiels :

1. Pour un n$\in \mathbb{N}$ donné, un langage qui définit une fonction f n'a qu'une image de largeur n : l'image n*f(n).
2. Au delà de la borne donnée, on dispose de l'équivalent d'un lemme de la pompe sur les images n*m avec m plus grand que la borne donnée.
3. Les résultats obtenus sont sur les langages d'images qui définissent des fonctions, donc sur un alphabet unaire. Le contenu d'une image n'a pas d'importance sur son appartenance au langage. En particulier il n'existe pas d'entiers a,b tel que l'on puisse trouver deux images p et q de taille a*b de sorte que p$\in$L et q$\notin$L.

– On remarque aussi que dans ces papiers, rien n'est dit sur les langages qui codent des fonctions inférieures asymptotiquement à la borne donnée au point 4.2. En particulier rien n'est dit sur les fonctions "bien proportionnées" : fonctions linéaires ou polynomiales. La raison est simple : pour tout polynome p fixé, "être de taille n*p(n)" est Σ_1 et Π_1 : cf [12] pour la partie Σ_1, pour la partie Π_1 il suffit de remarquer qu'être de taille n*m m<p(n) est Σ_1 et de même pour être de taille n*m avec m>p(n) donc que le complémentaire des images n*p(n) est Σ_1 et donc "être de taille n*p(n)" est Π_1. Ainsi on ne peut obtenir une séparation

de deux niveaux de la hiérarchie par des langages d'images polynomialement proportionnées définissant des fonctions.

Pour les raison ci-dessus, il n'est pas possible de faire une adaptation simple de ce résultat aux langages d'images bien proportionnées. De plus, une question sous jacente à la problématique du stage est de séparer deux niveaux de la hiérarchie MSO par un langage sur un alphabet binaire et où pour une taille d'image arbitrairement grande, il existe à la fois des images dans et en dehors du langage, de sorte que l'appartenance au langage soit déterminée par le contenu de l'image.

5 Divers résultats sur les images proportionnées

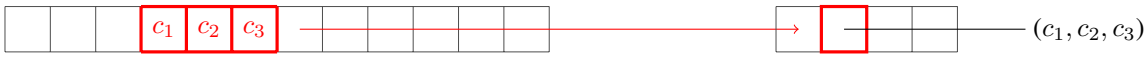
Dans cette section, nous allons nous intéresser aux langages d'images linéairement proportionnés. Dans un premier temps, nous allons montrer que l'on peut plier les langages sans changer leurs niveaux dans la hierarchie MSO. Ensuite, nous montrerons que l'on peut harmoniser la taille d'un langage linéairement proportionné au sens où les images seront n*c.n et non plus n*m avec n≤m≤c.n ou m≤n≤c.m.

La notion de pliage de langage de mots n'est pas nouvelle. Dans cette partie il s'agit de l'étendre aux images ainsi que de vérifier qu'une telle opération n'influe pas sur le niveau de définissabilité du langage plié. La deuxième partie est là dans un souci d'harmonisation des langages, néanmoins son impact sur les problématiques du stage est plus limité que pour le pliage. J'ai abordé la question sous les deux aspects : logique et reconnaissabilité. Cette dernière approche semble plus naturelle, et a donc été choisie pour ce rapport.

5.1 Pliage d'images proportionnées

A. Pliage d'un mot c.n par un mot n

L'idée est de replier les mots de longueur c.n en mots de longueur n en suivant le principe d'un accordéon. Une partie 1*c d'un mot w va être transformée en 1 lettre (partie 1*1), pour cela nous allons enrichir l'alphabet de départ.



mot 3*4, ici c=3 mot 4

Soit Σ un alphabet sur lequel est défini un langage L (dont les mots sont de longueur c.n -pour c un entier fixé-). On va lui associer un nouveau langage (dont les mots seront de taille n) sur un nouvel alphabet. On définit le nouvel alphabet comme Σ^c. On définit le c-plié p d'un mot q de longueur c.n comme suit :

p = $p_1 \quad \dots \quad p_n \quad \leftrightarrow$ q = $q_1^1 \quad \dots \quad q_1^c \quad \dots \quad q_n^1 \quad \dots \quad q_n^c$ avec p_i=$(q_i^1, \dots, q_i^c)$.

On définit maintenant le c-replié d'un langage L de mots dont les mots sont de taille c.n par plié(L)={p' | p' c-plié de p, p∈L}.

B. Pliage d'une image n*c.n par un une image n*n

L'idée se généralise sans problème aux images :

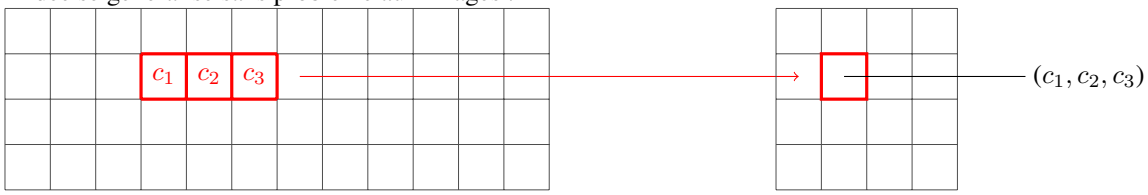


Image 4*(3*4), ici c=3 Image 4*4

$$\text{p} = \begin{matrix} p_{1,1} & \cdots & p_{1,n} \\ \vdots & \vdots & \vdots \\ p_{n,1} & \cdots & p_{n,n} \end{matrix} \in \text{L'} \leftrightarrow \text{q} = \begin{matrix} q^1_{1,1} & \cdots & q^c_{1,1} & \cdots & q^1_{1,n} & \cdots & q^c_{1,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ q^1_{n,1} & \cdots & q^c_{n,1} & \cdots & q^1_{n,n} & \cdots & q^c_{n,n} \end{matrix} \in L$$

avec q une image n*c.n et $p_{i,j}$=$(q^1_{i,j}, \ \ldots \ , q^c_{i,j})$.

On remarque que pour un alphabet Σ donné, toute image n*c.n sur Σ se plie sur Σ^c et que toute image n*n sur Σ^c est le replié d'une image n*c.n sur Σ. On a une correspondance bijective entre les images n*n sur Σ^c et les images n*c.n sur Σ. On peut donc parler d'une opération de dépliage : soit L' un langage sur Σ^c, on note déplié(L') le langage L sur Σ qui vérifie plié(L)=L'.

C. Localité et pliage sur les mots

Dans cette section, on ne s'intéresse qu'aux langages de mots. Sur les mots, un langage est local lorsqu'il est défini par un ensemble de dominos 1*2 (restriction simple de la notion de 2*2 localité sur les mots vus comme des images à une dimension).

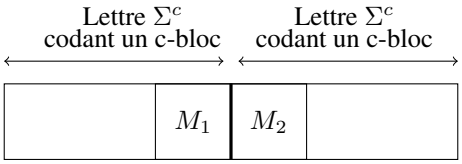
Lemme 1. *Soit B un langage local, alors B'=plié(B) est local.*

Démonstration. Soit B défini sur Σ local et B'=plié(B). Soit Δ un ensemble de dominos 1*2 qui définit B. On définit un ensemble Δ' de dominos 1*2 sur $\Sigma^c \cup \{\#\}$ comme suit :

- Cas des dominos centraux : $(r_1, \ldots, r_c)(s_1, \ldots, s_c) \in \Delta' \leftrightarrow (r_i, r_{i+1}) \in \Delta$ et $(s_i, s_{i+1}) \in \Delta$ et $(r_c, s_1) \in \Delta$ et r_i,$s_i \neq \#$.
- Cas des dominos de bord gauche : $(\#, (r_1, .., r_c)) \in \Delta' \leftrightarrow (\#, r_1) \in \Delta$ et $(r_i, r_{i+1}) \in \Delta$.
- Cas des dominos de bord droit : $((r_1, .., r_c), \#) \in \Delta' \leftrightarrow (r_c, \#) \in \Delta$ et $(r_i, r_{i+1}) \in \Delta$.

Soit A le langage reconnu par Δ', montrons que A=B'. Soit p un mot de B et p' son plié ($\in B'$) et soit x,y deux lettres consécutives de p', sans perte de généralité on suppose que x,y=$(x_1,..,x_c),(y_1,..,y_c)$. Alors $x_1,...,y_c$ est un sous-mot de p et donc tout sous-mot de longueur 2 de $x_1,...,y_c$ est dans Δ et donc (x,y) est dans Δ'. Si x est la première lettre du mot p' alors de même $(\#,x) \in \Delta'$. Idem si y est la dernière lettre de p' : $(y,\#) \in \Delta'$. Donc B'$\subset$A.
Soit q$\in$A, montrons que q$\in$B'. On suppose q=$x_1,..,x_n$=$(x_1^1,..,x_1^c),..,(x_n^1,..,x_n^c)$, soit r=$x_1^1,..,x_1^c,..,x_n^1,..,x_n^c$ alors r$\in$B de par la définition de Δ'. Alors plié(r) = r' = $((x_1^1,..,x_1^c),..,(x_n^1,..,x_n^c))$ = q, donc q$\in B'$ et A$\subset B'$. □

On ne peut pas avoir la réciproque, un système local sur un alphabet Σ ne prend pas en compte la position d'un pixel, or ceci peut être fait de manière artificielle sur l'alphabet Σ^c. Le système local sur Σ^c peut autorisé un motif de petite taille (en jouant sur le fait d'être sur un alphabet produit) sur les intersection des c-blocs mais pas ailleurs.



Bloc 1*2 sur Σ^c accepté par B'

M_1/M_2 motifs 1*2 sur Σ sur l'intersection des 2 cases codant 2 c-blocs

En assumant de plus que le motif M_1M_2 n'est pas possible autrement que sur le cas dessiné dessus. Un système local qui coderait B devrait accepter la tuile M_1M_2 et alors cette tuile pourrait être utilisée ailleurs que sur l'intersection de deux c-blocs.

Cependant on peut obtenir un résultat satisfaisant sous certains aspects :

Lemme 2. *Soit B un langage tel que ses images sont de longueur un multiple de c, et B'=plié(B) tel que B' est local. Alors B est reconnaissable.*

Démonstration. On rappelle que le choix de localité n'influe pas sur la reconnaissabilité (cf partie sur le degré de liberté de la notion de localité), le fait d'être sur les mots ne change pas cela.
Soit B défini sur Σ et B'=plié(B) local. Soit Δ' un ensemble de dominos 1*2 qui définit B'. On va créer Δ ensemble de dominos 1*2.c sur $(\Sigma * [1..c]) \cup \{\#\}$. L'idée est de marquer les positions des éléments du c-uplet qui compose une lettre de Σ^c, ceci permet de garder en mémoire le début d'une partie codée 1*c. Δ est composé de :

- Cas des cases centrales 1er partie : $(x_1*1, x_2*2, .., x_c*c, y_1*1, y_2*2, .., y_c*c) \in \Delta$ si $(x_1,..,x_c),(y_1,..,y_c) \in \Delta'$.
- Cas des cases centrales 2eme partie : $(x_j*j, ..x_c*c, y_1*1, ..., y_c*c, z_1*1, ..z_{j-1}*j-1) \in \Delta$ si j$\neq$1. Notons que les $x_i/y_i/z_i$ peuvent être n'importe quelle lettre de l'alphabet, de fait ce type de case est controlé par les cases centrales de 1er type (cf juste dessus) et en particulier les lettres de types x*1 qui sont autorisées.

– Cas des bords gauches : $(\#, x_1 * 1, ...x_c * c, y_1 * 1, ..y_{c-1} * c - 1) \in \Delta$ si $(\#, (x_1, .., x_c)) \in \Delta'$. Là encore les y_i peuvent être quelconques, et leur présence ou absence dans un mot est controlée par les cases centrales de type 1.
– Cas des bords droits : même idée que pour les cases de bord gauche, $(x_2 * 2, ...x_c * c, y_1 * 1, ..y_c * c, \#) \in \Delta$ si $((y_1, .., y_c), \#) \in \Delta'$.

On note A le langage reconnu par Δ et D la projection de A sur la première coordonnée. On remarque qu'une image de A a sa dernière lettre avec la deuxième composante qui est c. Montrons que B=D.
Soit p=$x_1...x_n \in$D et p_a un antécédent de p dans A par la projection sur la première coordonnée. Alors $p_a = x_1 * 1, x_2 * 2, .., x_{c-1} * c - 1, x_c * c, ..x_n * c$ et par définition de Δ', on a $(x_1, ..x_{c-1}), .., (..x_n) \in$B' et $x_1, ...x_n \in$ B. Donc D$\subset$B.
Inversement soit q=$y_1...y_n \in$ B, alors n est un multiple de c et q'=plié(q)=$(y_1, .., y_c), .., (y_{n-c+1}, .., y_n)$.
Dans ce cas $y_1 * 1, ..., y_n * c$ est dans A et $y_1, .., y_n$=q est dans D. Donc B$\subset$D.
B=D et par définition D est la projection du langage local A, donc B est reconnaissable.
□

D. Localité et pliage sur les images

Dans cette section, on ne s'intéresse qu'aux langages d'images. On va généraliser les résultats de la section précédente. Encore une fois, on fera appel à différents types de localité.

Lemme 3. *Soit B un langage d'images local, alors B'=plié(B) est local.*

Démonstration. Soit B défini sur Σ et local et B'=plié(B). Soit Δ_h et Δ_v deux ensembles de dominos 1*2 et 2*1 respectivement qui définissent B. On définit Δ'_h ensemble de dominos 1*2 sur $\Sigma^c \cup \{\#\}$ en suivant la construction dans la preuve du théorème similaire sur les langages de mots, auquel on ajoute $(\#, \#)$ -pour des raisons de définition des bordures qui ne sont pas importantes dans le cadre des mots-. On définit Δ'_v comme suit :

– Cases centrales : $\begin{matrix}(x_{1,1}, .., x_{1,c}) \\ (x_{2,1}, .., x_{2,c})\end{matrix} \in \Delta'_v$ si $(x_{j,i}, x_{j,i+1}) \in \Delta_h$ pour j=1,2 et tout i et $(x_{1,i}, x_{2,i}) \in \Delta_v$ pour tout i

– Bord supérieur : $\begin{matrix}\# \\ (x_1, .., x_c)\end{matrix} \in \Delta'_v$ si $(x_i, x_{i+1}) \in \Delta_h$ et $(\#, x_i) \in \Delta_v$ pour tout i.

– Bord inférieur : $\begin{matrix}(x_1, .., x_c) \\ \#\end{matrix} \in \Delta'_v$ si $(x_i, x_{i+1}) \in \Delta_h$ et $(x_i, \#) \in \Delta_v$ pour tout i.

– Bordures : $\begin{matrix}\# \\ \#\end{matrix}$. La présence de ces dominos est due aux bordures.

Soit A le langage local défini par Δ'_v et Δ'_h. Montrons que A=B' : Soit p'$\in$A, notons que p' est défini sur Σ^c, et soit p le déplié de p'. alors tout sous-mot 1*2 de p est dans Δ_h par construction de Δ'_h. De même pour les sous-mots 2*1 de p. Ainsi p est dans B et donc p' est dans B'. A$\subseteq$B'.
Soit p'$\in$B', et soit p le déplié de p' (p $\in B$). Puisque p $\in$B, toute ligne de p est acceptée

par Δ_h, et donc toute ligne pliée de p (ie ligne de p') est acceptée par Δ'_h par construction de Δ'_h. Ainsi tout sous-mot 1*2 de p' est dans Δ'_h. Même idée, tout sous-mot 2*1 de p' est dans Δ'_v et donc p' est reconnu par Δ'_h et Δ'_v donc est dans A. B'⊆A. □

Lemme 4. *Soit B un langage, et B'=plié(B) tel que B' est local. Alors B est reconnaissable.*

Démonstration. Soit B défini sur Σ et B'=plié(B) local. Soit Δ'_h et Δ'_v les ensembles de dominos respectivement 1*2 et 2*1 qui définissent B'. On crée deux ensembles de dominos respectivement 1*2.c (Δ_h) et 2*c (Δ_v) qui vont définir un langage local. L'ensemble Δ_h est défini comme dans le cadre des langages de mots (cf sous partie "localité et pliage sur les mots") à partir de Δ'_h. On définit Δ_v comme composé de :

- Cases centrales de type 1 : $\begin{matrix} x_1*1,..,x_c*c \\ y_1*1,..,y_c*c \end{matrix} \in \Delta_v$ si $\begin{matrix} (x_1,..,x_c) \\ (y_1,..,y_c) \end{matrix} \in \Delta'_v$
- Cases centrales de type 2 : $\begin{matrix} x_j*j,..,x_c*c,u_1*1,..,u_{j-1}*j-1 \\ y_j*j,..,y_c*c,v_1*1,..,v_{j-1}*j-1 \end{matrix} \in \Delta_v$ si j≠1.
  Le comportement de ces cases est controlé par les cases de type 1.

- Bord supérieur de type 1 : $\begin{matrix} \#,..,\# \\ x_1*1,..x_c*c \end{matrix}$ si (#,$(x_1,..,x_c)$)$\in \Delta_v$
- Bord supérieur de type 2 : $\begin{matrix} \#,\ \ldots,\# \\ x_j*j,..,x_c*c,y_1*1,..,y_{j-1}*j-1 \end{matrix}$
  Là encore le comportement est controlé par les cases de type 1.

- Bord inférieur : de même que pour les bords supérieur.
- Coins : on donne seulement le coin supérieur gauche, les autres sont sur le même principe. $\begin{matrix} \#,..,\# \\ \#,x_1*1,..x_{c-1}*(c-1) \end{matrix}$ si il existe z $\in \Sigma$ tel que (#,$(x_1,..,x_{c-1},z)$)$\in$ Δ_h et $\begin{matrix} \# \\ (x_1,..,x_{c-1},z) \end{matrix} \in \Delta_v$.

Soit A le langage local reconnu par $\Delta_h \cup \Delta_v$, A est donc défini sur Σ*[1..c]. Soit D la projection de A sur Σ. Montrons que D=B. Soit p, p_a et p' plié de p comme suit :

$$p=\begin{matrix} x_{1,1},..,x_{1,n*c} \\ \ldots\ \ \ldots\ \ \ldots \\ x_{m,1},..,x_{m,n*c} \end{matrix} ; p_a=\begin{matrix} x_{1,1}*1,..,x_{1,n*c}*c \\ \ldots\ \ \ldots\ \ \ldots \\ x_{m,1}*1,..,x_{m,n*c}*c \end{matrix} ; p'=\begin{matrix} (x_{1,1},..,x_{1,c}),..,(x_{1,n.c-c},..,x_{1,n.c}) \\ \ldots\ \ \ldots\ \ \ldots \\ (x_{m,1},..,x_{m,c}),..,(x_{m,n.c-c},..,x_{m,n*c}) \end{matrix}$$

Si on supppose que p∈A, alors p_a est un antécédent de p avant projection, et donc p_a ∈A. Alors p̂' est accepté par $\Delta'_h \cup \delta'_v$ et donc p'∈ B'. Or puisque le plié de p : p' est dans B', p∈B. D⊆B.
Si on suppose que p∈B, alors p'∈B' et p_a ∈A. Dans ce cas p est projection de p_a qui est dans A, donc p est dans D. B⊆D.
B=D et par définition D est la projection du langage local A, donc B est reconnaissable. □

E. Commutativité des opérations de pliage et de projection

Les opérations de pliage et de projection commutent, d'un point de vue intuitif, l'opération

de pliage ne perd pas d'information donc l'effectuer avant ou après projection n'a pas d'influence.

Lemme 5. *Soit p une image n*c.n sur un alphabet $\Sigma * \Gamma$, plié(p) son plié n*n sur $(\Sigma * \Gamma)^c$. Soit α défini sur $\Sigma * \Gamma$ comme la projection sur la première coordonnée. Soit α' défini sur $(\Sigma * \Gamma)^c$ par α' : $(a_1 * b_1, .., a_c * b_c) \mapsto (a_1, .., a_c)$.*
Alors α'(plié(p))= plié(α(p)).

Démonstration. Il suffit décrire p,α(p), plié(p), plié(α(p)) et α'(plié(p)).

$$\text{Soit p=} \begin{matrix} a_{1,1} * b_{1,1}, .., a_{1,n*c} * b_{1,n*c} \\ \vdots \quad \vdots \quad \vdots \\ a_{n,1} * b_{n,1}, .., a_{n,n*c} * b_{n,n*c} \end{matrix}$$

$$\text{alors } \alpha\text{(p)=} \begin{matrix} a_{1,1}, .., a_{1,n*c} \\ \vdots \quad \vdots \quad \vdots \\ a_{n,1}, .., a_{n,n*c} \end{matrix}$$

$$\text{et plié(p)=} \begin{matrix} (a_{1,1} * b_{1,1}, .., a_{1,c} * b_{1,c}), .., (a_{n*c-c+1} * b_{n*c-c+1}, .., a_{1,n*c} * b_{1,n*c}) \\ \vdots \quad \vdots \quad \vdots \\ (a_{n,1} * b_{n,1}, .., a_{n,c} * b_{n,c}), .., (a_{n,n*c-c+1} * b_{n,n*c-c+1}, .., a_{n,n*c} * b_{n,n*c}) \end{matrix}$$

$$\text{On a alors } \alpha'\text{(plié(p))=plié}(\alpha(p))\text{=} \begin{matrix} (a_{1,1}, .., a_{1,c}), .., (a_{n*c-c+1}, .., a_{1,n*c}) \\ \vdots \quad \vdots \quad \vdots \\ (a_{n,1}, .., a_{n,c}), .., (a_{n,n*c-c+1}, .., a_{n,n*c}) \end{matrix}$$ □

F. Pliage et hiérarchie MSO

Lemme 6. *Supposons que pour tout langage d'images L de format n*c.n, on ait L$\in \Sigma_k$ $\leftrightarrow$ plié(L)$\in \Sigma_k$. Alors pour tout langage d'images M de format n*c.n, on a M$\in \Pi_k \leftrightarrow$ plié(M)$\in \Pi_k$.*

La difficulté vient du fait que le complémentaire d'un langage L d'images n*c.n n'est pas un langage n*c.n mais c'est l'ensemble des images qui ne sont pas n*c.n plus celles qui le sont mais qui ne sont pas dans L. Ainsi la preuve bien que peu compliquée n'est pas triviale. Notons que "être de taille n*c.n", "de taille n*m (m<c.n)" ou "de taille n*m (m>c.n)" sont 3 propriétés qui sont à la fois Σ_1 et Π_1 (cf Appendice).

Rappelons aussi que les classes de langages Σ_k et Π_k sont closes par intersection et union (comme dit à la section 2.1).

Démonstration. Premier sens du théorème.
Soit L un langage d'images n*c.n. On note $\overline{L}_c$ l'ensemble des images n*c.n qui ne sont pas dans L et N_c l'ensemble des images de tailles n*c.n. Notons que N_1 est exactement les images carrées. On a :

L$\in \Pi_k \rightarrow \overline{L} \in \Sigma_k \rightarrow \overline{L} \cap N_c = \overline{L}_c \in \Sigma_k \rightarrow plie(\overline{L}_c) \in \Sigma_k \rightarrow \overline{plie(\overline{L}_c)} \in \Pi_k \rightarrow \overline{plie(\overline{L}_c)} \cap N_1 \in \Pi_k$

$\overline{plie(\overline{L}_c)}$ est formé des images de tailles différentes de n*n et des images n*n qui ne sont pas les repliées d'images de $\overline{L}_c$. Et donc $\overline{plie(\overline{L}_c)} \cap N_1$ est composé des images qui ne sont pas repliées d'images n*c.n de $\overline{L}_c$, c'est à dire les images repliées d'images de L. Donc $\overline{plie(\overline{L}_c)} \cap N_1$=plié(L= et plié(L)$\in \Pi_k$.

Réciproquement, on part d'un langage L' plié d'un langage L.
L'$\in \Pi_k \rightarrow \overline{L'} \in \Sigma_k \rightarrow \overline{L'} \cap N_1 \in \Sigma_k \rightarrow deplie(\overline{L'} \cap N_1) \in \Sigma_k \rightarrow \overline{deplie(\overline{L'} \cap N_1)} \in \Pi_k \rightarrow \overline{deplie(\overline{L'} \cap N_1)} \cap N_c \in \Pi_k$

$\overline{L'} \cap N_1$ est composé des images carrées qui ne sont pas dans L'. $\overline{deplie(\overline{L'} \cap N_1)}$ est composé des images de tailles différentes de n*c.n et des images n*n dépliées d'images qui sont dans L'. Donc $\overline{deplie(\overline{L'} \cap N_1)} \cap N_c$ est composé des images dépliés d'images de L' ie L et ainsi L$\in \Pi_k$. □

Avec le lemme ci-dessus, il devient possible de conclure que le pliage (et de manière équivalente, dépliage) n'influe ni sur la Σ_k définissabilité, ni sur la Π_k définissabilité.

Théorème 11. *Soit L un langage d'images de format n*c.n, alors :*

1. *L*$\in \Sigma_k \leftrightarrow$ *plié(L)*$\in \Sigma_k$.
2. *L*$\in \Pi_k \leftrightarrow$ *plié(L)*$\in \Pi_k$.

Démonstration. On procède par induction sur k.

Étape initiale
Soit L$\in \Sigma_1$=reconnaissable, donc L est projection d'un langage local A. De plus plié(L) = plié((projection(A)) = projection(plié(A)) de par la commutativité de la projection et du pliage. Or puisque A est local, plié(A) est local et donc plié(L) est projection d'un langage local, donc est reconnaissable=Σ_1.
Inversement si plié(L)$\in \Sigma_1$, plié(L) est projection d'un langage local B et L= déplié(projection(B)) = projection(déplié(B)). Or B est local, donc déplié(B) est reconnaissable et donc L est projection d'un langage reconnaissable et est lui-même reconnaissable.
Donc L$\in \Sigma_1 \leftrightarrow$ plié(L)$\in \Sigma_1$.

Avec le lemme ci-dessus, il est trivial de voir que L$\in \Pi_1 \leftrightarrow$ plié(L)$\in \Pi_1$.

Étape d'induction
Pour k$\geq$1, l'hypothèse d'induction est :

1. L$\in \Sigma_k \leftrightarrow$ plié(L)$\in \Sigma_k$.
2. L$\in \Pi_k \leftrightarrow$ plié(L)$\in \Pi_k$.

Soit L$\in \Sigma_{k+1}$, donc L est projection d'un langage Π_k A. De plus plié(L) = plié((projection(A)) = projection(plié(A)) de par la commutativité de la projection et du pliage. Or puisque

A est Π_k, plié(A) est aussi Π_k par hypothèse d'induction. Et donc plié(L) est projection d'un langageΠ_k, donc est Σ_{k+1}.
Inversement si plié(L)$\in \Sigma_k$, plié(L) est projection d'un langage B$\in \Pi_k$ et L = déplié(projection(B)) = projection(déplié(B)). Or B est Π_k, donc déplié(B) est Π_k par hypothèse d'induction.
Ainsi L est projection d'un langage Π_k et est Σ_{k+1}.

Donc L$\in \Sigma_{k+1} \leftrightarrow$ plié(L)$\in \Sigma_{k+1}$.
Avec le lemme ci-dessus, il est trivial de voir que L$\in \Pi_k \leftrightarrow$ plié(L)$\in \Pi_k$.

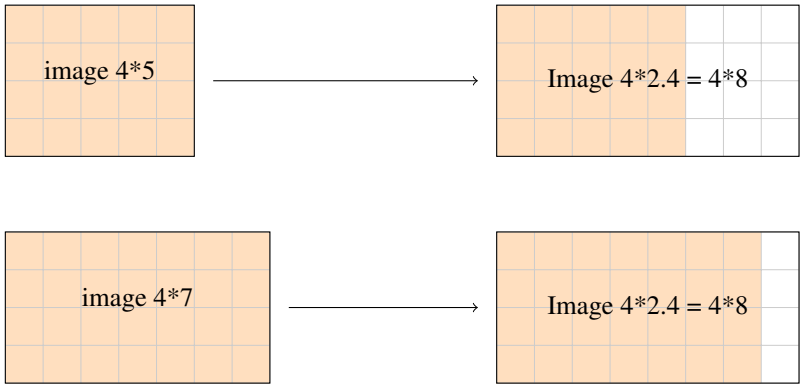
Ceci conclut l'induction et la preuve du théorème. □

Une conséquence évidente est que la hiérarchie MSO sur images carrées est équivalente à la hiérarchie sur langages d'images n*c.n (pour un entier c). Si l'une s'effondre, l'autre aussi.

5.2 Rembourrage de langages d'images

Dans toute cette partie, on s'intéresse à un langage linéairement proportionné dont les images sont de type n*m avec m∈[n,c.n] -pour c entier fixé minimal-. On va harmoniser la taille de tels langages. Le cas des langages linéairement proportionnés d'images n*m avec n$\in [m, c.m]$ est similaire. La notion de rembourrage sera relative à l'entier c.

A. Rembourrage d'images
L'idée est simple : à une image n*m du langage considéré, on associe une image n*c.n : on ajoute un ensemble qui va définir "l'ombre" de l'image et ajouter des pixels derrière. On va rembourrer l'image pour lui donner la forme rectangulaire voulue.



images correspondantes n*2.n

Dans l'image n*2.n la partie orange forme l'image n*m (la sous-image en orange forme l'image de gauche qui est 'parente' de cette image). La partie rembourrée (blanche) est remplie avec un nouveau symbole ξ qui n'est pas dans l'alphabet de départ. On note qu'une image sur Σ a une seule image rembourrée (de par la taille de celle-ci) et que

deux images différentes ont des homologues différents. Ainsi l'application qui à une image associe son homologue rembourré est injective.
Pour un langage d'image L, on note L' le langage modifié issu de L dont on a rembourré les images.

B. Localité et rembourage

Lemme 7. *Soit L un langage d'images linéairement proportionnées de facteur c et L' son homologue rembourré (n*c.n). Alors : L est local → L' est reconnaissable.*

Démonstration. Soit L un langage local défini sur un alphabet Σ. L est décrit par Δ, ensemble de pavés 2*2 autorisés. Alors on définit Δ'' ensemble de pavés 2*2 sur $\Sigma \cup \{\xi\} \cup \{\#\}$ comme Δ auquel on a :

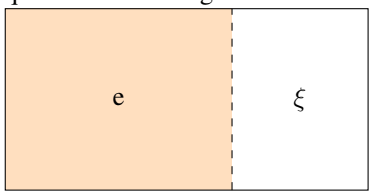
- Retiré les pavés de type $\begin{matrix} a & \# \\ b & \# \end{matrix}$ (a,b quelconques dans $\Sigma \cup \{\#\}$).
- Ajouté $\begin{matrix} a & \xi \\ b & \xi \end{matrix}$ si on a $\begin{matrix} a & \# \\ b & \# \end{matrix} \in \Delta$
- Ajouté $\begin{matrix} \xi & \xi \\ \xi & \xi \end{matrix}$; $\begin{matrix} \# & \# \\ \xi & \xi \end{matrix}$; $\begin{matrix} \# & \# \\ \xi & \# \end{matrix}$; $\begin{matrix} \xi & \# \\ \xi & \# \end{matrix}$; $\begin{matrix} \xi & \# \\ \# & \# \end{matrix}$; $\begin{matrix} \xi & \xi \\ \# & \# \end{matrix}$

On note E_1 le langage local défini par Δ'' (sur $\Sigma \cup \{\xi\}$).

On sait que "être de taille n*c.n" est une propriété reconnaissable, soit N_c le langage composé des images n*c.n sur $\Sigma \cup \{\xi\}$. On note B l'intersection des langages E_1 et N_c. Notons que B est reconnaissable puisque intersection d'un langage local et d'un langage reconnaissable. Montrons que B=remb(L).

Soit a' $\in$ remb(L). Soit a son homologue non rembourré dans L. Alors a' est de taille n*c.n par définition de remb(L) et l'image bordée de a' (â') est reconnu par Δ'' (puisque â est reconnu par Δ et de par la définition de Δ''). Donc a'$\in$B. Ainsi remb(L)$\subseteq$B.

Soit a'$\in$B. Alors â' est reconnu par Δ'' et puisque Δ'' ne comporte pas les tuiles $\begin{matrix} \xi & a \\ \xi & b \end{matrix}$ pour a/b$\in \Sigma$, on en déduit que a' est la concaténation de deux parties, l'une (notée e) qui forme une image sur Σ et l'autre définie sur $\{\xi\}$:



Alors e est une image qui munie des bordures (#) -ê- sera acceptée par Δ puisque â' est accepté par Δ''. Donc e $\in$ L et puisque a' est le rembourré de e, a'$\in$remb(L). Ainsi B$\subseteq$remb(L)

On conclut que B=remb(L) et comme B est reconnaissable, remb(L) aussi. □

On peut obtenir un résultat dans le sens inverse :

Lemme 8. *Soit L un langage d'images linéairement proportionnées de facteur c et L' son homologue rembourré (n*c.n). Alors : L' est local → L est local.*

Démonstration. Soit L un langage sur Σ et remb(L) son homologue rembourré sur $\Sigma \cup \{\xi\}$ tel que remb(L) est local. Soit Δ' l'ensemble de tuiles 2*2 qui accepte remb(L). Notons que puisque toutes les images acceptées par Δ' sont rembourrées d'un langage sur Σ, on peut supposer que Δ' ne contient pas de tuiles de type :
$\begin{matrix} \xi & a \\ \xi & b \end{matrix}$ avec a,b$\in \Sigma$.
On construit Δ à partir de Δ' en :

- Supprimant tout pavé qui a ξ.
- Ajoutant $\begin{matrix} a & \# \\ b & \# \end{matrix}$ si $\begin{matrix} a & \xi \\ b & \xi \end{matrix} \in \Delta'$.

On note A le langage local accepté par Δ. Montrons que A=L.
Soit a$\in$A et a' son homologue rembouré. â' est accepté par Δ' puisque â est accepté par Δ. Alors a'$\in$remb(L) et a$\in$L. Donc A$\subseteq$L
Soit a$\in$L et a' son homologue dans remb(L), alors â' est accepté par Δ' et par construction de Δ, â est accepté par Δ et donc a$\in$A. L$\subseteq$A.

Ainsi A=L et puisque A est local, L est local. □

C. Commutativité des opérations de rembourrage et de projection

Lemme 9. *Soit p une image d'un langage linairement proportionné de facteur c sur un alphabet $\Sigma * \Gamma$, remb(p) son homologue rembourré sur $(\Sigma * \Gamma) \cup \{\xi\}$. Soit α défini sur $\Sigma * \Gamma$ comme la projection sur la première coordonnée. Soit α' défini sur $(\Sigma * \Gamma) \cup \{\xi\}$ par :*

- *(a*b) $\mapsto$ a si (a,b) $\in (\Sigma * \Gamma) \cup \{\xi\}$.*
- *$\xi \mapsto \xi$.*

Alors remb(α(p))=α'(remb(p)).

Démonstration. Sur le même principe que pour la commutativité du pliage et de la projection, il suffit d'écrire l'image p, remb(α(p)) et α'(remb(p)). On s'apercoit que remb(α(p)) = α'(remb(p)). □

D. Rembourrage et hiérarchie MSO

Lemme 10. *Supposons que pour tout langage d'images linéairement proportionné L, on ait : L*$\in \Sigma_k \leftrightarrow$ *remb(L)*$\in \Sigma_k$*. Alors pour tout langage d'images M, on a M*$\in \Pi_k \leftrightarrow$ *remb(M)*$\in \Pi_k$.

Démonstration. La preuve est exactement la même que pour le lemme dans la partie "Pliage et hiérarchie MSO". L'opération pliage ou rembourrage n'intervient pas directement, elle est cachée dans l'hypothèse du lemme (pour tout langage d'images L : L$\in \Sigma_k \leftrightarrow$ remb(L)$\in \Sigma_k$). □

On est maintenant en mesure d'énoncer le résultat attendu :

Théorème 12. *Soit L un langage d'images, alors :*

1. *L*$\in \Sigma_k \leftrightarrow$ *remb(L)*$\in \Sigma_k$.
2. *L*$\in \Pi_k \leftrightarrow$ *remb(L)*$\in \Pi_k$.

Démonstration. On procède par induction sur k.

Étape initiale
L'étape initiale est exactement la même que pour la preuve sur les langages pliés. On obtient que pour tout L, L$\in \Sigma_1 \leftrightarrow$ remb(L)$\in \Sigma_1$ et L$\in \Pi_1 \leftrightarrow$ remb(L)$\in \Pi_1$.

Étape d'induction
Pour k$\geq$1, l'hypothèse d'induction est :

1. L$\in \Sigma_k \leftrightarrow$ plié(L)$\in \Sigma_k$.
2. L$\in \Pi_k \leftrightarrow$ plié(L)$\in \Pi_k$.

Pareil, en reprenant la preuve faite sur les langages pliés, on arrive à prouver l'étape d'induction au rang k+1.
On conclut alors l'induction et la preuve du théorème. □

Il est donc possible de modifier un langage linéairement proportionné en un langage dont les images sont exactement n*c.n. De manière générale, pour bon nombre d'opérations simples, il sera possible d'obtenir un résultat similaire. Il suffit que l'opération choisie commute avec la projection et que l'opération ne perde pas la notion de reconnaissabilité d'un langage.

On obtient ainsi une équivalence entre les hiéarchies MSO sur langages d' images carrées et la hiéarchie MSO linéairement proportionnée :

Proposition 6. *Un éffondrement de la hiéarchie MSO à un niveau k -* $\Sigma_k = \Sigma_{k+1}$*- sur langages d'images linéairement proportionnées entraine un éffondrement de la hiéarchie sur langages d'images carrés au même niveau.*

Démonstration. Soit L un langage linéairement proportionné sur un alphabet Γ qui sépare deux niveaux de la hiérarchie : L $\in \Sigma_{k+1}$, L$\notin \Sigma_k$. Puisque L est linéairement proportionné, il existe un entier c tel que les images de L sont n*m avec m$\in$[n,c.n] ou n*m avec n$\in$[m,c.m].
On décompose L en deux langages L_1 et L_2, L_1 a ses images de format n*m avec

m∈[n,c.n]et L_2 de format n*m avec n∈[m,c.m]. L_1 et L_2 sont aussi dans Σ_{k+1}, par exemple L_1 est l'intersection de L et du langage composé des images dont le nombre de ligne est plus petit que le nombre de colonne -ce langage est à la fois Σ_1 et Π_1-. Il n'est pas possible que les deux langages L_1 et L_2 soient tout deux Σ_k, sinon leur union (L) le serait aussi.
Supposons que L_1 ne soit pas Σ_k.Alors soit M le c-rembourré de L_1 sur $\Gamma \cup \{\xi\}$, on a M $\in \Sigma_{k+1}$, M$\notin \Sigma_k$. Soit N le c-plié de M, sur $(\Gamma \cup \{\xi\})^c$. Alors N $\in \Sigma_{k+1}$, N$\notin \Sigma_k$. N est un langage d'images carrées et on a un langage carré qui sépare les niveaux Σ_k et Σ_{k+1}. □

6 Conclusion

La structure particulière des langages EMSO(=Σ_1), comme projection de langage locaux, est la pierre de base de la hiérarchie MSO. On a vu que la localité choisie importe peu dès lors qu'apparait une projection, celle-ci gommant les différences. La projection est aussi une opération qui commute avec d'autres opérations simples, comme on l'a vu avec le pliage ou le rembourrage vers un format linéaire fixé. Ceci a pour conséquence qu'un langage peut être plié ou rembourré sans changer son niveau dans la hiérarchie et a fortiori que les hiérarchies MSO sur langages d'images carrées et sur langages linéairement proportionnés sont équivalentes. Il reste inconnu si les hiérarchies sur langages d'images polynomialement proportionnées et sur langages d'images carrées sont équivalentes.

Les raisonnements exposés dans ce rapport sont plus de nature combinatoire que logique. La raison étant que les arguments de non-Σ_k-définissabilité logique pour k$\geq$2 reposent sur des extensions des jeux d'Ehrenfeucht-Fraissé, et que pour ces extensions peu de stratégies gagnantes sont connues. Le point de vue combinatoire est à la fois plus intuitif et dispose d'arguments de non-définissabilité plus malléables pour les langages définissables par combinaisons booléennes de formules Σ_1 ; cependant ces arguments apparaissent difficiles à appliquer pour des niveaux plus élevés de la hiérarchie.

Dans l'espoir de répondre à la principale question du stage (les hiérarchies MSO sur langages d'images polynomialement ou linéairement proportionnées s'effondrent-elles ?) de manière satisfaisante, il faudrait trouver des arguments de non définissabilité basés sur le contenu des images du langage considéré.

D'un point de vu personnel, ce stage reste comme un premier contact avec la recherche et mon premier travail nécessitant une bibliographie allant au delà de 3 ou 4 articles. La majorité des résultats présentés dans ce rapport renforce l'idée acquise au cours de mes études : il est particulièrement difficile de produire un résultat qui ne s'appuie que de manière superficielle sur les résultats connus (et non de manière importante). Ainsi la borne du taux de croissance des fonctions Σ_k se base sur un lemme de la pompe sur les images, la preuve utilisant des méthodes de théorie des automates.

7 Bibliographie

Références

[1] Ajtai, Fagin, and Stockmeyer. The closure of monadic np. In *Journal of computer and system sciences*, pages 309–318, 1997.

[2] Borchert. Formal language characterizations of p, np, and pspace. *Journal of Automata, Languages and Combinatorics*, 13, 2008.

[3] Giammarresi, Restivo, Seibert, and Thomas. Monadic second order logic over rectangular pictures and recognizability by tiling systems. *Information and Computation*, 125 :32–45, 1996.

[4] Grandjean and Olive. Descriptive complexity for pictures languages. In *Computer Science Logic (CSL'12) - 26th International Workshop/21st Annual Conference of the EACSL*, volume 16 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 274–288, 2012.

[5] Grandjean and Olive. A logical approach to locality in pictures languages. *preprint submitted to Journal of Computer and System Sciences*, 2012.

[6] Latteux and Simplot. Recognizable picture languages and domino tiling. *Theoretical Computer Science*, 178, 1996.

[7] Libkin. *Elements of Finite Model Theory*. 2004.

[8] Matz. One quantifier will do in existential monadic second order logic over pictures. *Lecture Notes in Computer Science*, 1450, 1998.

[9] Matz and Schweikardt. Expressive power of monadic logics on words, trees, pictures, and graphs. 2008.

[10] Matz and Thomas. The monadic quantifier alternation hierarchy over graphs is infinite. pages 236–244, 1997.

[11] Otto. A note on the number of monadic quantifiers in monadic sigma1,1. *Information Processing Letters*, 25, 1995.

[12] Rozenberg and Salomaa. *Handbook of formal languages*, chapter Two-dimensional langages. 1997.

[13] Schweikardt. The monadic quantifier alternation hierarchy over grids and pictures. pages 236–244, 1997.

[14] Thomas. Classifying regular events in symbolic logic. *Journal of Computer and System Sciences*, 25, 1982.

8 Appendice

Être de taille n*c.n", "de taille n*m avec n≤m≤c.n" ou "de taille n*m avec (c-1).n≤m≤c.n" -pour c entier≥1 fixé- sont 3 propriétés reconnaissables.

On ne donne qu'un schéma de preuve : Etant donné un alphabet Σ quelconque, l'idée est de construire un ensemble de domino sur un alphabet $\Sigma * \Gamma$ codant c diagonales sur la composante Γ et en laissant la composante Σ libre (comme dans le dessin ci-dessous). Il faut ensuite vérifier que la dernière diagonale existe et atterrit sur le coin droit/inférieur/supérieur -suivant la propriété que l'on veut vérifier-. Pour une preuve formelle, on peut s'inspirer de l'exemple 7.2 de [12] -ils montrent que "être carré" est local-.