USTL LIFL Laboratoire d'Informatique Fondamentale de Lille

N° d'ordre : 1650

THESE

Nouveau Régime

Présentée à

L'UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

Pour obtenir le titre de

DOCTEUR en INFORMATIQUE

par

Denis ROBILLIARD

LANGAGES DE FIGURES

Thèse soutenue le 15 Janvier 1996, devant la Commission d'Examen

Président	:	Mireille CLERBOUT	Université de LILLE I
Rapporteurs	:	Maurice NIVAT	Université de PARIS VII
	:	Patrice SÉÉBOLD	Université de PICARDIE
Examinateurs	:	Philippe AIGRAIN	Université de TOULOUSE III
	:	Michel LATTEUX	Université de LILLE I

Je remercie Mireille Clerbout pour avoir accepté de présider ce jury. Son accord me touche d'autant plus que je garde un excellent souvenir des cours qu'elle dispensait et auxquels j'ai assisté lors de la première année de mon cursus informatique à Lille I.

Je remercie Maurice Nivat et Patrice Séébold qui m'ont fait l'honneur de bien vouloir consacrer une partie de leur temps à rapporter cette thèse.

Je remercie Philippe Aigrain qui m'a fait l'honneur de participer à ce jury.

Je remercie Michel Latteux qui a encadré ces recherches avec une disponibilité jamais démentie. Je remercie Karine Slowinski, Bodonirina Ratoandromanana et David Simplot qui ont, à un moment ou à un autre, amené leur compétence et leur bonne humeur à notre groupe de travail.

Je remercie tous les membres du laboratoire, les amis et les parents qui ont, à leur manière, soutenu mes efforts et permis la réalisation de ces travaux.

Enfin, j'accorde une pensée toute particulière à l'ensemble de l'équipe du service de transplantation d'organes de l'hopital Calmette de Lille, dirigée, en 1992, par le Professeur Ribet. Sans leur savoir-faire, ce mémoire n'aurait pas été rédigé.

Table des matières

Chapitre 1

Introduction

1.1 À la recherche d'un formalisme adéquat

Parmi la multitude des problèmes qui questionnent l'intelligence humaine, il en est une certaine classe qui relève intrinsèquement du domaine de la géométrie. D'autres encore ont un énoncé qui s'appréhende plus facilement par l'intermédiaire de ce domaine. Les perspectives offertes par l'informatique, avec des possibilités nouvelles ouvertes à l'exploration, ont généré leur lot d'interrogations appartenant à ces classes. Un des problèmes clés que l'on y rencontre est celui de la reconnaissance des formes, dont les applications sont nombreuses. Un autre champ notable est constitué par la génération automatique ou semi-automatique de motifs, qui doivent répondre à des critères esthétiques ou mathématiques. Dans cette problématique, nous pouvons ranger la génération d'images fractales et de polyominos, ces derniers étant utilisés pour certaines expériences de physique statistique. Nous mentionnerons également les questions de pavages, qui ressortent aussi de l'analyse combinatoire.

Aborder les problèmes que nous venons d'énumérer renvoie à un dénominateur commun : la nécessité de disposer d'un "bon" formalisme pour les représenter (Fig. 1.1). La valeur en sera jaugée par son adéquation à décrire exactement le phénomène étudié, par le caractère pratique, accessible des descriptions, par la possibilité, plus ou moins grande, d'en extraire des connaissances et des relations. Sur ce dernier point le choix d'un formalisme n'est pas neutre, mais au contraire il oriente notre étude, en permettant par exemple l'accès à un corpus de savoir déjà bien établi, sur lequel nous obtenons de surcroît un nouveau point de vue.

1.2 Une solution simple et puissante : les mots

Les problèmes mentionnés plus haut s'expriment souvent de manière naturelle par une représentation en deux dimensions. L'expérience nous montre cependant que l'emploi d'un formalisme unidimensionnel, les mots, nous en permet aussi

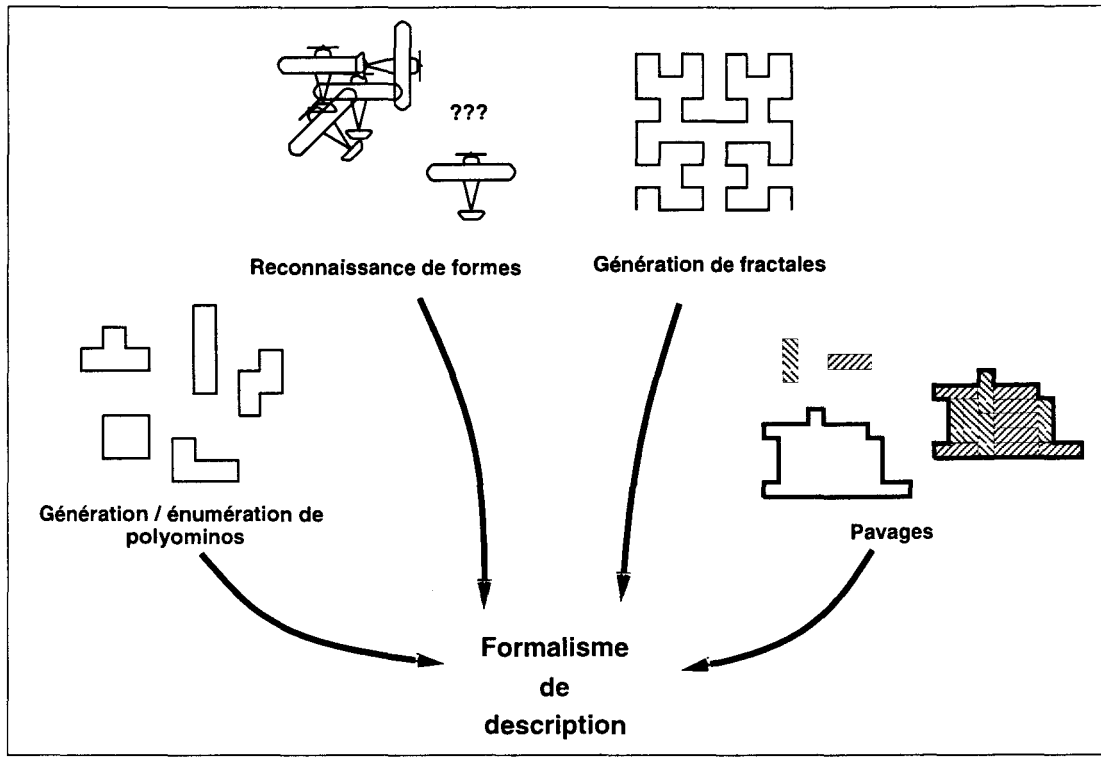


FIG. 1.1 - *Un formalisme commun pour différents problèmes.*

l'approche. Nous n'obtiendrons certes pas des solutions dans tous les cas, mais nous pourrons traiter néanmoins un domaine de portée non négligeable, avec un outil pertinent et efficace.

La génération de fractales en fait usage, entre autres pour modéliser la croissance des plantes à l'aide de L-systèmes. Dans ce contexte, proposé en 1968 par Lindenmayer [Lin68], et développé par la suite notamment par Prusinkiewicz [Pru86], on part d'un axiome, constitué d'un mot, auquel on applique des règles de réécriture, de manière itérative. On obtient un mot, que l'on épelle, chaque lettre déclenchant une action graphique sur l'écran ou la table traçante. Par exemple sur l'alphabet $\{X, [,], +, -\}$, on peut prendre pour axiome le mot "X", et pour seule règle de réécriture "$X \rightarrow X[+X]X[-X]X$". À chaque itération on réécrit la chaîne en substituant la partie droite de la règle à chaque occurrence de X (et non à une seule occurrence comme dans les systèmes de réécriture habituels). On obtient après une itération :

$$X[+X]X[-X]X.$$

Après deux itérations :

$$X[+X]X[-X]X[+X[+X]X[-X]X]X[+X]X[-X]X[-X[+X]X[-X]X]X[+X]X[-X]X.$$

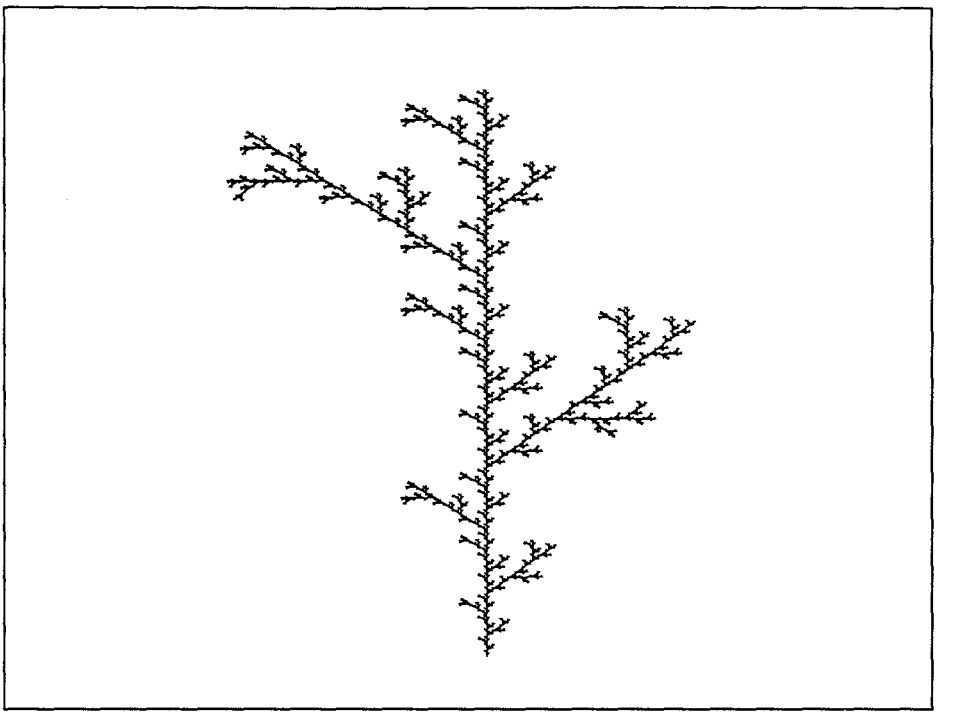


FIG. 1.2 - *Plante générée par un L-système.*

Le mot résultat peut alors être interprété comme une suite d'instructions du langage LOGO, avec, par exemple, la sémantique suivante :

> La tortue graphique est placée initialement à l'origine, pointant vers le nord, la lettre "X" la fait avancer d'un pas, la lettre "+" indique un pivotement dans le sens trigonométrique de $\pi/8$, la lettre "−" la même rotation mais en sens inverse, et les symboles "[" et "]" donnent l'ordre respectivement de mémoriser sur une pile la position et l'orientation de la tortue, et de dépiler cette information pour restaurer ces paramètres.

Au bout de six itérations on obtient l'image de la Fig. 1.2. Cet exemple intéressant est tiré de [PS88].

Un des problèmes que nous avons évoqués dans la première section, l'énumération de polyominos, trouve souvent sa solution par l'emploi de mots. Considérons un pavage régulier du plan par des carrés unitaires, un polyomino (voir [Gol65]) est alors une union finie de carrés, union qui doit être 4-connexe, *i.e.* la connexité est définie par les bords, mais pas par les coins (Fig. 1.3). L'énumération des polyominos d'une classe donnée (convexes, piles...) intéresse les mathématiciens, mais aussi les physiciens, car ces constructions peuvent modéliser certains objets physiques comme des chaînes polymères, ou bien certains phénomènes critiques

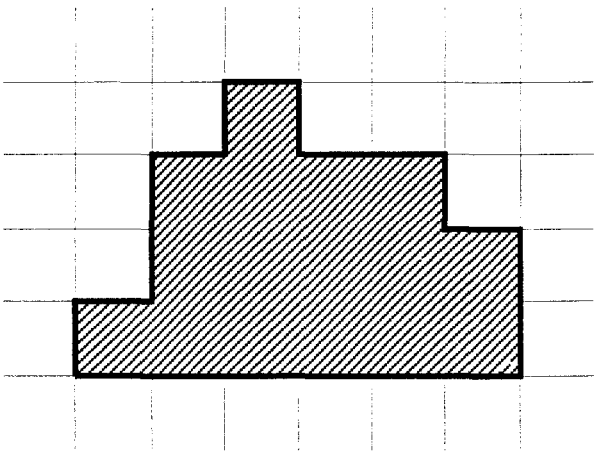


FIG. 1.3 - *Exemple de polyomino pile.*

comme la percolation (voir [PS89, Vie92, Del92]). Une des techniques employées consiste à suivre le schéma méthodologique proposé par Schützenberger [Sch62] : on décrit par exemple la frontière ou l'intérieur (*l'animal*) du polyomino par un mot de telle sorte que l'on puisse établir une bijection entre l'ensemble des polyominos de la classe étudiée et les mots d'un langage algébrique. La recherche d'une grammaire non-ambiguë pour laquelle on évalue une fonction génératrice permet alors l'énumération.

Dans un autre domaine, savoir si l'on peut paver le plan avec une pièce ou un jeu de pièces, est un problème qui nécessite l'analyse du contour des pièces en question. De même, si l'on s'intéresse au pavage d'une pièce finie, là encore le contour est une donnée fondamentale. Dans les deux cas, cette donnée se prête soit directement à une représentation sous la forme d'un mot, nous en reparlons un peu plus loin, soit à une interprétation très proche : on peut définir une structure de monoïde sur des courbes orientées, dont les propriétés ressemblent alors à celles de mots classiques (voir [BN91b]).

Enfin, pour la reconnaissance de formes, l'approche syntaxique, où un motif se décompose en sous-motifs qui tiennent lieu de primitives, a été proposée de longue date ([Fu74]). Dans cette approche, les grammaires de graphes, d'images ou de tableaux, qui fonctionnent avec un mécanisme génératif à deux dimensions, constituent un outil puissant pour la description ou création d'ensembles de motifs, au prix d'une certaine complexité ([ENRR87]). Plus simple, le formalisme des mots peut, là aussi, être envisagé dans bien des cas. En effet, il est souvent possible de parcourir séquentiellement l'ensemble d'un motif, en énumérant une à une les primitives que l'on rencontre : si l'on code chaque primitive par une lettre, un motif est alors décrit par un mot, et un ensemble de motifs par un langage de mots. On se met donc en situation de pouvoir utiliser les méthodes et les outils de la théorie des langages formels, et c'est l'approche que proposèrent Maurer,

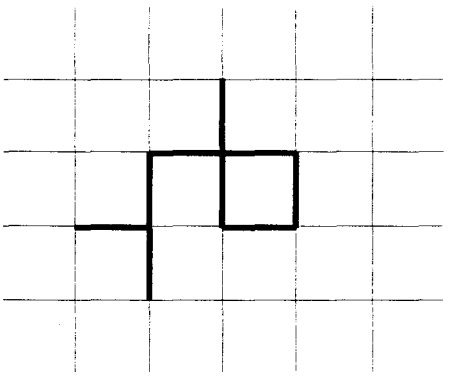


FIG. 1.4 - *Exemple de figure.*

Rozenberg et Welzl [MRW82], au début des années quatre-vingt.

1.3 Langages de figures et de mots de figures

C'est le modèle de Maurer *et al.* que nous adopterons, pour l'affiner par la suite. Il se caractérise toutefois par certaines contraintes. Tout d'abord les motifs ou *figures* sont dessinés sur le plan cartésien $\mathbb{Z}^2$, considéré comme une grille régulière à mailles carrées, et sont constitués d'un ensemble fini de segments de droite de longueur unitaire, soit verticaux, soit horizontaux, chacun d'eux joignant deux nœuds voisins de la grille. On peut se donner une idée intuitive de ce système en imaginant de tracer les figures sur un papier quadrillé, aucun trait ne pouvant s'écarter du quadrillage. La seconde obligation est que les figures doivent être connexes (voir Fig. 1.4).

On conçoit bien que l'un des retentissements de la contrainte posée par cette structure de grille est de réduire le champ des objets représentables dans ce modèle. Cependant on notera qu'il s'agit d'une limitation somme toute assez naturelle dans l'environnement informatique. En effet l'utilisation de la matrice de pixels d'un écran d'ordinateur pour la visualisation des images impose elle aussi une structure en forme de réseau à mailles quadrangulaires, de même que les motifs collectés par un numériseur se présentent habituellement eux aussi sous cette forme. Le choix d'un maillage suffisamment fin permet, d'autre part, d'obtenir une approximation de n'importe quelle image avec une bonne précision. Le gain escompté en se limitant à un domaine d'application restreint n'est autre qu'une meilleure efficacité. Ces critères matériels ne sont pas négligeables et ont d'ailleurs suffisamment d'impact pour constituer une des motivations du développement de la géométrie discrète. Nous estimons que si ce système ne vise pas à l'universalité, il n'en constitue pas moins un compromis intéressant qui autorise la représentation, par des moyens simples, d'une grande variété de formes, comme nous allons le voir par la suite.

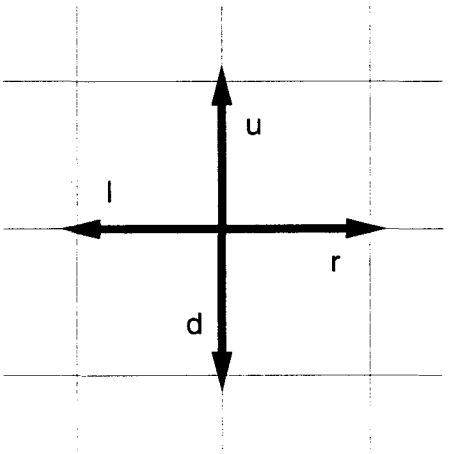


FIG. 1.5 - *Directions associées aux lettres de* Π.

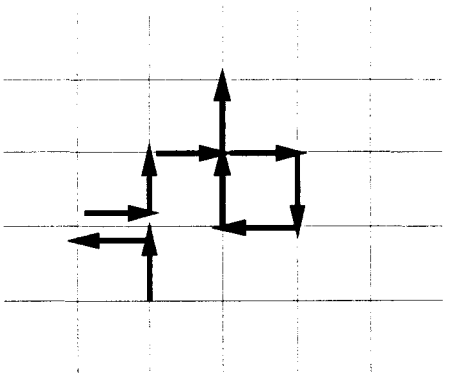


FIG. 1.6 - *Exemple de parcours et mot de figure.*

Pour dessiner les figures, dans ce modèle, nous allons tirer parti de leur connexité. On peut en effet imaginer de placer un crayon sur un des nœuds de la grille atteint par un des segments qui appartiennent à la figure, puis de déplacer le crayon de nœud en nœud, en suivant les segments qui appartiennent au motif, sans lever le crayon. À chaque nœud on se déplace soit vers la droite, soit vers la gauche, soit vers le haut, soit vers le bas. Si l'on associe une lettre de l'alphabet $\Pi = \{u, r, d, l\}$ à chacune de ces directions, u pour haut, r pour droite, d pour bas, l pour gauche (d'après l'anglais *up, right, down, left*), on peut construire un mot sur Π qu'on appelle *mot de figure*, voir Fig. 1.5 et Fig. 1.6. On notera qu'il est parfois nécessaire de repasser plusieurs fois sur un même segment. Notons toutefois que rien ne distingue un segment parcouru une fois d'un segment parcouru plusieurs fois, et qu'une même figure peut être décrite par différents mots : par exemple le motif formé d'un segment unitaire horizontal est représenté aussi bien par le mot r que par le mot l, ou même par chacun des mots du langage infini $r(lr)^*$. Par la suite, nous noterons $\mathrm{Fig}(\mu)$ la figure décrite par le mot μ de Π^*, et $\mathrm{Fig}(\mathcal{L})$ le langage des figures décrites par les mots du langage $\mathcal{L}$ inclus dans Π^*.

En fait, il est possible de trouver une trace antérieure de cette méthode de représentation dans une proposition de Freeman [Fre61], dont l'objectif consistait à stocker dans la mémoire d'un ordinateur des courbes continues tracées sur un maillage discret. Les déplacements en diagonale était pris en compte par l'utilisation d'un alphabet à huit symboles. Ces travaux précurseurs ne se situaient cependant pas dans le cadre de la théorie des langages, embryonnaire à l'époque.

Un intérêt, et non des moindres, de [MRW82] réside dans le caractère systématique de l'approche de la dualité figures et mots de figures. Les langages de figures y sont classés selon une hiérarchie réminiscente de celle de Chomsky, en posant qu'un langage de figures $\mathcal{G}$ est rationnel si et seulement s'il existe un langage rationnel de mots de figures $\mathcal{K}$ qui le décrit. C'est à dire que pour toute figure de $\mathcal{G}$, il doit exister un mot de $\mathcal{K}$ qui la décrit, et tout mot de $\mathcal{K}$ décrit une figure de $\mathcal{G}$. De manière similaire sont définis les langages de figures linéaires, algébriques, contexte-sensitifs et récursivement énumérables. L'étude démontre alors notamment que, contrairement à ce qui se passe dans le cas des mots, les familles des langages de figures contexte-sensitifs et récursivement énumérables sont les mêmes. On peut bien sûr s'attendre à ce que les langages les plus simples dans cette classification présentent des propriétés plus fortes, et il est intéressant de sonder leur pouvoir d'expression.

1.4 Un tour d'horizon

D'autres recherches sont venues approfondir ce domaine. Nous allons en donner une présentation succincte.

Problème	Description	Classe		
		Rat.	Lin.	Alg.
Appartenance	$\forall\mu, \forall\mathcal{K}, \mathrm{Fig}(\mu) \in \mathrm{Fig}(\mathcal{K})$?	NP-complet		
Intersection	$\forall\mathcal{K}, \forall\mathcal{L}, \mathrm{Fig}(\mathcal{K}) \cap \mathrm{Fig}(\mathcal{L}) = \emptyset$?	indécidable		
Équivalence	$\forall\mathcal{K}, \forall\mathcal{L}, \mathrm{Fig}(\mathcal{K}) = \mathrm{Fig}(\mathcal{L})$?	indécidable		
Inclusion	$\forall\mathcal{K}, \forall\mathcal{L}, \mathrm{Fig}(\mathcal{K}) \subset \mathrm{Fig}(\mathcal{L})$?	indécidable		
Sous-figure	$\forall\mu, \forall\mathcal{K}, \exists\nu \in \mathcal{K}, \mathrm{Fig}(\mu) \subset \mathrm{Fig}(\nu)$?	NP-complet		
Sur-figure	$\forall\mu, \forall\mathcal{K}, \exists\nu \in \mathcal{K}, \mathrm{Fig}(\mu) \supset \mathrm{Fig}(\nu)$?	temps polynomial		

FIG. 1.7 - *Quelques problèmes de décision (μ, ν des mots et $\mathcal{K}, \mathcal{L}$ des langages).*

1.4.1 Comparaisons des figures

Un certain nombre de questions sont inspirées par le thème de la reconnaissance des formes. C'est le cas du problème de l'appartenance (*membership problem*), où l'on cherche à savoir si une figure, donnée par un mot, appartient à un langage, défini par une grammaire de mots. Parmi les problèmes dérivés de celui-ci, on trouve celui de la sous-figure, de la sur-figure, de l'équivalence, de l'intersection et de l'inclusion. On se souviendra qu'une figure donnée (non vide) a toujours une infinité de descriptions, et c'est ce qui rend ces problèmes difficiles. Par exemple, $\mathrm{Fig}(u)$ appartient au langage $\mathrm{Fig}((ud)^*)$, bien que le mot u n'appartienne pas, lui, au langage $(ud)^*$. Le tableau en Fig. 1.7 récapitule les principaux résultats, montrés dans [MRW82], [SW85], [KS87], [HW88]. Pour les langages de figures contexte-sensitifs, on constate sans surprise que ces problèmes deviennent indécidables.

Comme on peut le voir, la majorité de ces problèmes sont difficiles ou inabordables. Pour pallier à cette situation, des classes plus restreintes de langages de figures ont été définies, dans l'espoir de résultats plus prometteurs et en prenant en compte le fait qu'il n'est pas rare que des problèmes réels puissent être modélisés par des langages plus contraints que les langages généraux de la hiérarchie de Chomsky. Nous en mentionnons trois, avec tout d'abord la classe des langages-fenêtres (*stripe-languages*), où toutes les figures doivent être comprises dans la portion de plan délimitée par deux droites parallèles de pente donnée, concept précisé plus exactement dans [SW85], où il est montré que l'on peut décider si un langage algébrique de mots de figures décrit un langage-fenêtre. Ensuite notons les langages à 3 directions, définis sur un sous-ensemble de l'alphabet Π ne comprenant que trois lettres, et les langages à retours bornés, où l'on fixe un naturel k tel qu'aucun mot du langage ne peut contenir plus de k occurrences d'une succession de lettres rl ou lr : on limite ainsi le nombre de changement de direction possibles sur l'axe horizontal ("retours en arrière"), ce qui fait de ce concept une extension de celui des langages à 3 directions. On trouvera les définitions précises

Problème	Classe		
	Rat.	Lin.	Alg.
Appartenance	temps linéaire	NP-complet	
Intersection	décidable	indécidable	
Équivalence	décidable	indécidable	
Inclusion	décidable	indécidable	

FIG. 1.8 - *Problèmes de décision dans les langages-fenêtres.*

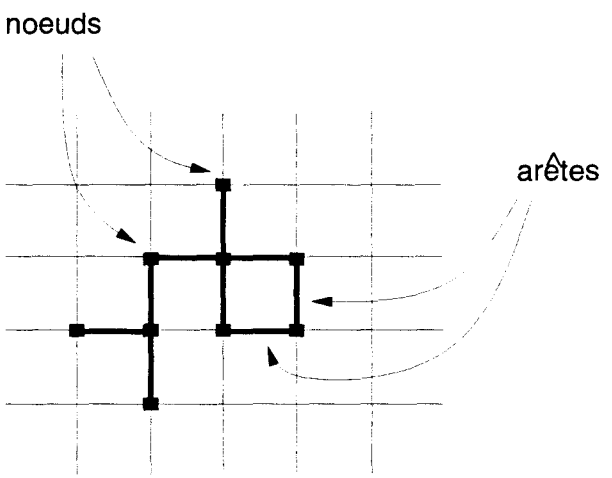


FIG. 1.9 - *Une figure vue comme un graphe.*

de ces deux dernières notions dans [Kim90] et [KS92]. Le tableau de la Fig. 1.8 reprend certaines des questions de la Fig. 1.7 dans le cas des langages-fenêtres.

1.4.2 Propriétés géométriques et graphiques

On peut aussi envisager un figure comme un graphe particulier, l'ensemble des arêtes étant l'ensemble des segments unitaires de la figure, celui des nœuds étant constitué par les points de $\mathbb{Z}^2$ aux intersections de la grille qui sont atteints par un des segments (voir Fig. 1.9). On cherche alors à savoir si un graphe est, par exemple, eulérien ou hamiltonien, ou bien encore s'il est un arbre ([Das88]). On cherche aussi la possibilité de mettre en évidence des propriétés géométriques telles que convexité et simplicité des figures ([Das89]), ou si elles représentent une famille de motifs particuliers comme les contours de polyominos quelconques et les courbes closes ([Bea91]), ou encore les polyominos convexes ([BLS92]). La forme selon laquelle les questions précédentes sont souvent posées se ramifie en

Problème	Classe		
	Rat.	Lin.	Alg.
graphe eulérien	¬ d		
graphe hamiltonien	¬ d		
arbre	¬ d		
courbe simple	d	¬ d	
courbe convexe	d	¬ d	
courbe close	d		
polyomino	¬ d		
polyomino convexe	d	¬ d	
d = décidable, ¬ d = indécidable.			

FIG. 1.10 - *Propriétés des graphes, propriétés géométriques.*

deux cas, comme suit :

> Soit $\mathcal{K}$ un langage de mots de figures d'une famille donnée, peut-on décider si :
>
> 1. il existe au moins une figure de Fig($\mathcal{K}$) qui vérifie la propriété,
> 2. toutes les figures de Fig($\mathcal{K}$) vérifient la propriété.

Une présentation des principaux résultats obtenus sur ces problèmes pour la sous-question 1, existentielle, est faite en figure 1.10. La sous-question 2 a été étudiée de manière moins exhaustive, on sait toutefois qu'elle est décidable pour les graphes eulériens, les courbes simples et les courbes closes, et ce dans les trois classes de langages rationnels, linéaires et algébriques.

1.4.3 Descriptions minimales

La multiplicité des descriptions associées à une figure donnée, mentionnée plus haut, est à l'origine d'une autre catégorie de travaux qui cherchent à obtenir des représentations de longueur minimale ou quasi-minimale. Obtenir un mot de longueur minimale est possible en temps polynomial comme l'a montré Brandenburg [Bra89], par des méthodes issues de la théorie des graphes. Une autre approche a été proposée par Séébold et Slowinski, qui donnent un système de réécriture permettant d'obtenir tous les mots décrivant une même figure. Ils en dérivent un algorithme qui permet lui aussi de construire un mot minimal, mais cette fois à l'aide de transformations sur les mots, dans [SS90, SS91], et on

trouve par ailleurs un résultat similaire dans [Gut89]. D'autres réductions, inspirées du système précédent, permettent d'obtenir des mots quasi-minimaux en temps quadratique ([BD93]).

On a aussi proposé, dès l'origine, différents systèmes qui réécrivent les mots, en modifiant éventuellement les figures, pour en conserver certaines propriétés, dans le but notamment de rechercher des formes normales ([MRW82, Hin89, SS94]). On peut notamment espérer obtenir ainsi des outils pouvant aider à la démonstration d'autres résultats sur les mots de figures.

1.4.4 De nouvelles sémantiques

Pour étendre le champ des objets décrits par les langages de mots de figures, une solution consiste à modifier l'alphabet et/ou la sémantique qui lui est associée.

On a proposé des systèmes pour la description de figures avec plusieurs couleurs de trait ([Hin86]), et l'emploi d'une couleur "invisible" pour la représentation des figures non connexes a bien sûr été l'un des premiers objectifs visé. On peut dans ce cas ajouter des symboles qui correspondent, dans la métaphore de la table traçante, à des instructions de lever et de baisser de crayon. Nous aborderons plus loin dans l'ouvrage quels peuvent être les problèmes sémantiques soulevés par cette innovation. Notons toutefois, au sujet des questions de décidabilité dont on a parlé plus tôt, que ce formalisme permet parfois d'apporter des réponses à des problèmes difficilement solubles dans le point de vue classique ([HW88]). Cela dit, d'une manière générale, les résultats sont plus faibles dans la sémantique avec lever de crayon (voir [Hin88, Slo90, Das91, DH93, Slo93]).

Une autre proposition vient de Gutbrod [Gut90, Gut91], et consiste à se munir, en plus des quatre lettres habituelles, de deux symboles pour empiler et dépiler la position courante du crayon. Le lecteur attentif aura remarqué que ce même raffinement nous a été utile lorsque nous avons donné plus haut un exemple de fractale générée par un L-système. Effectivement, le pouvoir d'expression des classes de langages ainsi définies se trouve augmenté, ce dont nous reparlerons.

1.4.5 Problèmes apparentés

Tel que nous l'avons présenté, le formalisme des mots de figures trouve aussi son emploi en vue d'autres d'objectifs.

Sans entrer dans les détails, nous mentionnons les problèmes de pavages des polyominos. L'une des méthodes employées fait appel à une représentation des polyominos par un mot décrivant leur contour. On peut ainsi caractériser certaines figures pour lesquelles un pavage utilisant un jeu de pièces donné est possible: voir notamment [BN91a, BN92, Rem92, AB94].

Par ailleurs la génération de mots de figures par des classes spécifiques de L-systèmes est abordée dans [DH91], avec comme point de mire la réalisation de structures hiérarchisées (fractales, circuits VLSI).

1.5 Contribution à l'étude des langages de figures

Dans cette section nous présentons un résumé des travaux exposés dans la suite de la thèse. Comme nous l'avons mentionné plus haut, nous nous sommes placés dans la lignée des recherches initiées par Maurer *et al.*. D'une manière générale, nous nous sommes souciés de mesurer le pouvoir d'expression de la sémantique classique, et de le comparer à celui de variantes proposées depuis. Nos travaux se sont principalement focalisés sur la famille des langages rationnels, car c'est la plus exploitable, les résultats d'indécidabilité y étant relativement moins nombreux. Entre autres, nous avons aussi souhaité partir du point de vue du graphiste, et donc définir des opérations à priori sur les figures, plutôt que directement sur les mots, afin de jauger si le formalisme étudié est à même de concrétiser une démarche intuitive.

Dans la première partie, nous définissons formellement les différents concepts dont nous faisons l'étude, en insistant sur la dualité qui rassemble et oppose tout à la fois d'une part les langages de figures et d'autre part les langages de mots de figures. La connaissance de ces notions de base est un préalable nécessaire pour aborder le reste de l'ouvrage, toutefois le lecteur familier de ce genre de travaux s'y trouvera en terrain connu.

Nous posons qu'une figure est la classe d'équivalence, modulo une translation dans le plan, d'un ensemble de segments unitaires de la grille, ensemble qui n'est pas forcément connexe. Nous introduisons alors les figures pointées, qui sont des figures munies de deux points distingués qui servent à les concaténer. Cette concaténation permet, entre autres, de définir une classe des rationnels directement sur les figures, classe dont on a par la suite l'exact équivalent sur les mots. Ensuite nous présentons la sémantique classique sur l'alphabet $\Pi = \{u, r, d, l\}$, évoquée précédemment dans cette introduction, puis la sémantique avec lettres dites "invisibles", sur l'aphabet $\Pi_i = \{u, r, d, l, u', r', d', l'\}$, où les lettres primées indiquent un mouvement crayon levé, sans tracé. Nous rappelons la notion de mot minimal, qui est un mot le plus court représentant une figure donnée, et celle de complexité descriptive, qui est le rapport entre la taille d'un mot minimal et celle de la figure qu'il représente. Cette dernière notion est donc un indicateur de la facilité avec laquelle on peut représenter une figure dans une sémantique donnée.

Dans la deuxième partie de la thèse, nous remarquons que l'usage des lettres invisibles est intéressant même lorsque l'on se limite au cas des figures connexes. Nous nous intéressons au problème de la recherche de mots minimaux sur cette sémantique.

Dans le cas des figures connexes, il existe déjà une solution à cette question, connue aussi sous le nom du problème du postier chinois. Dans le cas des figures non connexes nous montrons que ce problème est NP-dur, en ramenant à cette question une variante du problème du voyageur de commerce.

Ensuite nous étendons à la sémantique des lettres invisibles un résultat de complexité descriptive de Maurer *et al.*. Pour cela nous construisons un langage infini de la famille des langages EOL, telle que la complexité sur Π_i des figures de tout sous-ensemble infini de ce langage n'est bornée que par 2. Nous montrons aussi que, par contre, pour tout langage algébrique, il existe un sous-ensemble infini pour lequel cette complexité est borné par un nombre strictement plus petit que 2.

Par ailleurs, de nombreuses preuves d'indécidabilité, sur les langages linéaires et algébriques, font appel au problème de Post. Nous illustrons cette technique en démontrant qu'on ne peut pas décider de l'existence d'un mot de contour de polyomino convexe dans un langage linéaire. Pour aborder la classe des rationnels, d'autres auteurs ont recours à des constructions basées sur des machines de Turing. Nous exposons une méthode, toujours basée sur le problème de Post, qui nous permet de redémontrer trois résultats connus : l'indécidabilité de l'existence d'un mot décrivant un arbre ou un contour de polyomino dans un rationnel, ou une figure connexe dans un rationnel sur Π_i. Nous répondons par la même méthode à une question de Berstel, en montrant qu'on ne peut pas décider de l'existence d'un mot minimal dans un rationnel.

Dans la troisième partie, nous définissons, sur des figures non pointées, une opération de collage dite "superposition". La superposition de deux figures est l'ensemble des figures obtenues en dessinant les deux motifs l'un sur l'autre, avec comme seule contrainte que le résultat soit connexe. Nous montrons alors que cette opération ne préserve pas la rationnalité des langages sur Π.

Nous nous sommes intéressé à la sémantique des langages avec branchements, proposée par Gutbrod. Dans ces langages, définis sur $\Pi_b = \{u, r, d, l, \#, \&\}$, il est possible de mémoriser sur une pile la position du crayon à un moment donné afin de se replacer à cet endroit par la suite, grâce aux symboles $\{\#, \&\}$, qui codent respectivement l'empilement et le dépilement. Un langage dont tous les mots sont bien parenthésés sur les symboles $\{\#, \&\}$ est dit descriptif. Après avoir complété ce système en introduisant une notion de mot inverse, nous montrons que la superposition préserve la rationalité et la descriptivité de ces langages, et qu'elle n'augmente que de 1 la profondeur de la pile nécessaire à les implémenter. Nous montrons que l'itération de la superposition possède ces mêmes propriétés.

Dans la quatrième et dernière partie, nous proposons une sémantique nouvelle, sur l'alphabet $\Pi_p = \{u, r, d, l\}$, où les lettres codent toujours les même déplacement, mais, au lieu de tracer des segments, allument le carré unitaire ou "pixel" situé à droite de l'arête parcourue. Une figure devient donc un ensemble de pixels. Ce système nous semble plus à même de prendre en compte les préoccupations des graphistes, et nous permet d'aborder différemment la riche problématique des polyominos et des pavages.

Dans cette optique, nous étudions un collage sur les polyominos pointés, plus simple que celui envisagé dans la partie précédente. Notre opération est définie

comme étant la concaténation de deux polyominos pointés, et elle n'est licite que s'il n'y a pas chevauchement des deux motifs, ce qui nous rapproche des pavages. Nous appelons "carrés unitaires" les quatre polyominos pointés unitaires dessinés chacun par une des quatre lettres de notre alphabet, et polyomino "prolongeable" un polyomino pointé auquel on peut coller un carré unitaire. Que pouvons nous dire du collage des carrés unitaires?

Nous avons restreints nos recherches aux polyominos dits "optimaux" qui sont ceux de complexité descriptive égale à 1. Nous démontrons que l'on ne peut pas obtenir tous les polyominos optimaux par collage répété des carrés unitaires, ni non plus tous les polyominos optimaux prolongeables. Nous montrons que, par contre, tous les polyominos optimaux prolongeables, et qui sont sans trous, peuvent être obtenus par collage itéré des carrés unitaires.

Ce dernier résultat conclut notre ouvrage, et nous semble ouvrir la voie à d'autres questions intéressantes tant sur ce collage que sur les langages à pixels.

Première partie

Notions fondamentales

Chapitre 1

Notations

Dans la suite du mémoire, nous nous efforcerons d'utiliser un jeu de notations uniforme, afin de faciliter la lecture. Nous en présentons les conventions ici. Lorsqu'il sera nécessaire de déroger à ces habitudes, nous l'indiquerons à notre lecteur.

1.1 Références essentielles

Nous supposons par ailleurs que ce dernier connaît les bases de la théorie des langages formels. À défaut, nous l'aiguillons vers quelques ouvrages qui en présentent les notions fondamentales: [Eil74, Har78, Ber79, HU79, Aut87, Lee90a]. Nous rappelons plus bas quelques définitions usuelles.

Nous utiliserons par moments des notions très élémentaires de théorie des graphes. S'il était nécessaire, le lecteur peut se référer à [GM84, Lee90a].

1.2 Conventions typographiques

L'ensemble des naturels est noté $\mathbb{N}$, celui des relatifs $\mathbb{Z}$, celui des réels $\mathbb{R}$. Un ensemble est représenté en général par une majuscule calligraphiée (ex: $\mathcal{A}$), mais les alphabets sont, quant à eux, notés par une lettre grecque majuscule (ex: Π). Le cardinal d'un ensemble $\mathcal{A}$ est $\mathrm{Card}(\mathcal{A})$. L'ensemble des parties d'un ensemble $\mathcal{A}$ est $2^{\mathcal{A}}$. La différence ensembliste entre $\mathcal{A}$ et $\mathcal{B}$ est écrite $\mathcal{A} \setminus \mathcal{B}$. Le monoïde libre engendré par l'alphabet Π est identifié par Π^*. Une lettre générique dans un alphabet est indiquée par une lettre minuscule (ex: $a \in \Pi$). Pour les indices sont utilisées des lettres minuscules, de préférence entre i et n. Les chaînes finies sur un alphabet, appelées *mots* sont notées en general par des lettres grecques minuscules (ex: $\mu \in \Pi^*$).

1.3 Vocabulaire de base

La plupart des définitions de cette section portent sur des mots ou des langages de mots sur l'alphabet Γ.

Le mot vide est noté ε. La longueur d'un mot μ est indiqué par $|\mu|$. Le nombre d'occurrences de la lettre a dans μ est noté $|\mu|_a$. L'ensemble des facteurs (respectivement gauche, droit) d'un mot μ est $\mathrm{Fac}(\mu) = \{\omega \in \Gamma^* \mid \mu = \mu_1\omega\mu_2, \text{ et } \mu_1, \mu_2 \in \Gamma^*\}$ (respectivement $\mathrm{FacG}(\mu) = \{\omega \in \Gamma^* \mid \mu = \omega\mu_2, \text{ et } \mu_2 \in \Gamma^*\}$, $\mathrm{FacD}(\mu) = \{\omega \in \Gamma^* \mid \mu = \mu_1\omega, \text{ et } \mu_1 \in \Gamma^*\}$). Le *miroir* d'un mot μ, noté $\underline{\mu}$, est défini récursivement par: $\underline{\mu a} = a\underline{\mu}$ avec $a \in \Gamma$, et $\underline{\varepsilon} = \varepsilon$. Soient les langages $\mathcal{A}$ et $\mathcal{B}$, inclus dans Γ^*, nous notons $\mathcal{B}^{-1}\mathcal{A} = \{\omega \in \Gamma^* \mid \exists \mu \in \mathcal{A}, \exists \nu \in \mathcal{B}, \mu = \nu\omega\}$ (respectivement $\mathcal{A}\mathcal{B}^{-1} = \{\omega \in \Gamma^* \mid \exists \mu \in \mathcal{A}, \exists \nu \in \mathcal{B}, \mu = \omega\nu\}$ le quotient à gauche (respectivement à droite) de $\mathcal{A}$ par $\mathcal{B}$. Soit un automate fini $R =< \Gamma, Q, D, F, \delta >$, on note $R_{qq'}$ le langage reconnu par l'automate déduit de R en prenant l'état q comme unique état de départ, et l'état q' comme unique état final.

Chapitre 2

Langages de figures

Dans ce chapitre nous allons définir formellement et précisément les motifs graphiques qui constituent la base des deux premières parties de notre étude. Nous nous intéressons à des dessins qui sont des agrégats de segments horizontaux et verticaux, non orientés, et tracés sur une grille à mailles carrées. On retrouve ici les notions développées dans [MRW82], avec toutefois comme différence que nous avons autorisé les dessins non connexes.

2.1 Segments unitaires

Nous donnons les formalismes correspondant à la notion de distance que nous utilisons. Cette dernière est connue sous le nom de distance rectilinéaire ou distance de Manhattan ou encore "city-block distance". Puis nous définissons les notions de segments unitaires dont sont faits nos dessins, puis celles relatives à la connexité, qui nous permettent de considérer comme une entité indépendante tout objet intuitivement "en un seul morceau".

Définition 2.1.1 *La* distance *entre deux points de* $\mathbb{Z}^2$, $i = (x_i, y_i)$ *et* $j = (x_j, y_j)$, *est donnée par:* $Dist(i,j) = |x_i - x_j| + |y_i - y_j|$. *Elle est encore appelée* distance rectilinéaire.

Définition 2.1.2 *Soient* $i = (x_i, y_i)$ *et* $j = (x_j, y_j)$ *deux points de* $\mathbb{Z}^2$, *tels que* $Dist(i,j) = 1$. *On appelle* segment unitaire *le segment de droite* $[i,j]$, *noté* $Seg(i,j)$. *Un segment unitaire n'est pas orienté:* $Seg(i,j) = Seg(j,i)$. *S'il n'y a pas d'ambiguïté, nous dirons simplement* segment *au lieu de segment unitaire.*

Définition 2.1.3 *L'armature d'un segment* $Seg(i,j)$ *est :* $Arm(Seg(i,j)) = \{i,j\}$. *C'est à dire l'ensemble des points extrémités du segment. L'armature d'un ensemble de segments est l'union des armatures des segments de l'ensemble.*

On notera que deux ensembles de segments différents peuvent avoir la même armature (voir Fig. 2.1).

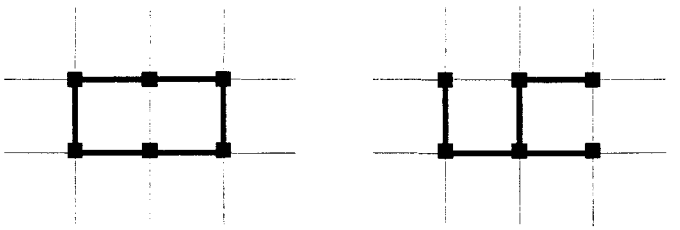


FIG. 2.1 - *Ensemble de segments différents ayant même armature.*

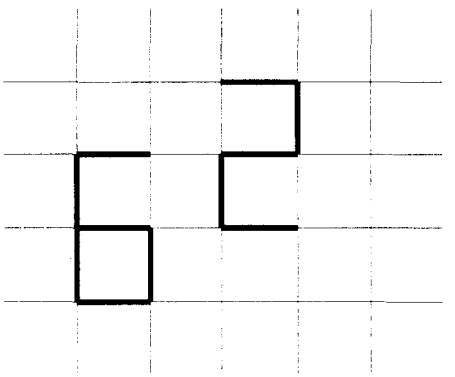


FIG. 2.2 - *Ensemble de segments avec deux composantes connexes.*

Définition 2.1.4 *Un ensemble de segments e est* connexe *si et seulement si quels que soient s et t deux segments de e, il existe une suite de segments* $\{s_0, \ldots, s_n\}$ *dans e tels que :* $\forall i \in [0, n-1]$, $Card(Arm(s_i) \cap Arm(s_{i+1})) = 1$ *et* $s = s_0$, $t = s_n$.

C'est à dire que si l'on prend une paire de segments quelconques de l'ensemble, alors on peut aller de l'un à l'autre en empruntant une suite de segments de l'ensemble tels que chacun d'eux est contigu par une extrémité avec le suivant.

Définition 2.1.5 *Une* composante connexe *d'un ensemble de segments e est un ensemble de segments f tel que :*

- $f \neq \emptyset$
- $f \subset e$
- *f connexe*
- $\forall p \in e \setminus f \Rightarrow f \cup \{p\}$ *non connexe*

2.2 Dessins et dessins pointés

Nous pouvons maintenant présenter la notion de *dessin*, dont nous dériverons par la suite celle de *figure*, qui sera l'objet fondamental de notre étude. Pour pouvoir coller des dessins, nous définissons aussi le *dessin pointé*, c'est à dire un dessin muni de deux points distingués qui servent d'ancrage pour concaténer. C'est une manière classique de procéder, que l'on retrouve aussi bien dans les travaux spécifiques aux langages de figures que dans ceux qui ont trait aux grammaires de graphes. Notons que les points distingués ne sont pas obligatoirement situés sur l'armature du dessin, notre définition ayant été choisie pour pouvoir obtenir des objets non connexes par concaténation d'objets plus simples.

Définition 2.2.1 *Un* dessin *est un ensemble fini de segments. On note $\mathcal{D}$ l'ensemble des dessins. Le dessin vide est l'ensemble vide, noté d_ε.*

Définition 2.2.2 *Un* dessin pointé *est un triplet (e, d, a), avec e un dessin et d, a des points de $\mathbb{Z}^2$. On note $\mathcal{D}'$ l'ensemble des dessins pointés. Un dessin pointé (d_ε, d, a) est dit vide si et seulement si $d = a$.*

Notons que dans le cas d'un dessin pointé (d_ε, d, a) où $d \neq a$, nous considérons donc avoir affaire à un dessin pointé non vide. Cette définition du dessin pointé vide se justifie par le souhait d'en faire un élément neutre vis à vis de l'opération de concaténation présentée plus bas.

Nous verrons un peu plus loin que les points distingués, ajoutés au concept de dessin pour obtenir celui de dessin pointé, sont destinés à jouer un rôle lors de la concaténation. Ce rôle n'étant pas similaire pour chacun des deux points, nous aurons parfois besoin d'échanger ces deux points l'un pour l'autre. Cela nous amène à définir la notion de l'inverse d'un dessin pointé.

Définition 2.2.3 *L'*inverse *d'un dessin pointé $f = (e, d, a)$ est le dessin pointé noté $\overline{f} = (e, a, d)$.*

Définition 2.2.4 *La* base *d'un dessin pointé $f = (e, d, a)$ est : $Base(f) = e$. C'est à dire que l'on oublie les points de départ et d'arrivée. De même, soit $\mathcal{A}$ un ensemble de dessins pointés, alors on note : $Base(\mathcal{A}) = \{Base(f) \mid f \in \mathcal{A}\}$.*

Exemple : les deux dessins pointés de la Fig. 2.3 ont pour base le dessin de la même figure.

Définition 2.2.5 *Le* poids *d'un dessin e, noté $|e|$, est le cardinal de e. Le poids d'un dessin pointé f est le poids de la base de f.*

Exemple : Chaque dessin de la Fig. 2.3 a pour poids 7.

Définition 2.2.6 *La* concaténation *de deux dessins pointés $f = (e, d, a)$ et $f' = (e', d', a')$, notée $f.f'$, est définie si et seulement si $a = d'$. On a alors : $f.f' = (e \cup e', d, a')$.*

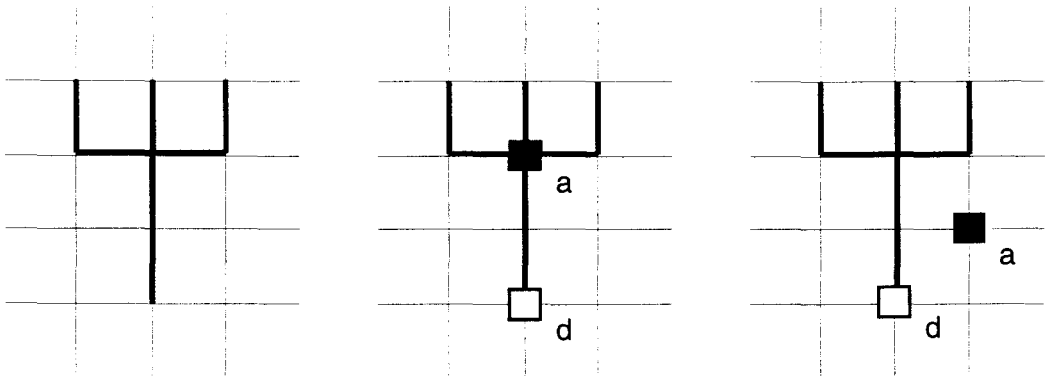


FIG. 2.3 - *Un dessin et deux dessins pointés.*

Définition 2.2.7 *Un* dessin pointé connexe *est un triplet* (e, d, a)*, avec* e *un dessin connexe et* d*,* a *des points de* $\mathbb{Z}^2$*, tels que si* $e = d_\epsilon$ *alors* $d = a$*, sinon* d *et* a *appartiennent à l'armature de* e*.*

On notera que cette définition de la connexité est cohérente avec ce que l'on en attend : la concaténation de dessins pointés connexes est toujours un dessin pointé connexe et le dessin pointé vide est connexe. Exemple : parmi les dessins de la Fig. 2.3, seul le dessin pointé situé à droite n'est pas connexe.

2.3 Figures et figures pointées

En général nous nous intéressons plus à la forme d'un dessin qu'à l'endroit précis du plan où il est dessiné. Par conséquent nous introduisons la notion de *figure* pour désigner une classe d'équivalence de dessins modulo une translation dans le plan.

Définition 2.3.1 *Soit* $(x, y) \in \mathbb{Z}^2$*, on définit la translation* $\tau_{x,y}$ *sur les entiers par :*

$$\tau_{x,y} : \begin{cases} \mathbb{Z}^2 & \longrightarrow & \mathbb{Z}^2 \\ (i,j) & \longmapsto & (i+x, j+y) \end{cases}$$

On définit la translation $t_{x,y}$ *sur les dessins par :*

$$t_{x,y} : \begin{cases} \mathcal{D} & \longrightarrow & \mathcal{D} \\ d & \longmapsto & \{Seg(\tau_{x,y}(i), \tau_{x,y}(j)) \mid Seg(i,j) \in d\} \end{cases}$$

On définit la translation $t'_{x,y}$ *sur les dessins pointés par :*

$$t'_{x,y} : \begin{cases} \mathcal{D}' & \longrightarrow & \mathcal{D}' \\ \{e,d,a\} & \longmapsto & \{t_{x,y}(e), \tau_{x,y}(d), \tau_{x,y}(a)\} \end{cases}$$

On définit la relation d'équivalence $\perp$ *entre dessins par :*

$$\forall a, b \in \mathcal{D}, a \perp b \Leftrightarrow \exists (x,y) \in \mathbb{Z}^2, a = t_{x,y}(b)$$

On définit la relation d'équivalence $\top$ *entre dessins pointés par:*

$$\forall a, b \in \mathcal{D}', a \top b \Leftrightarrow \exists (x, y) \in \mathbb{Z}^2, a = t'_{x,y}(b)$$

Définition 2.3.2 *Une* figure *est la classe d'équivalence d'un dessin e pour la relation* $\perp$. *Une* figure pointée, *encore appelée* p-figure *est la classe d'équivalence d'un dessin pointé f pour la relation* $\top$. *La base d'une p-figure* $f = [e]_\top$ *est la figure* $Base(f) = [Base(e)]_\perp$.

Nous pouvons remarquer que quels que soient $a, b \in \mathcal{D}'$, si $a \top b$ alors Base(a) $\perp$ Base(b).

Définition 2.3.3 *On note* $\mathcal{F}$ *l'ensemble des figures et* $\mathcal{F}'$ *l'ensemble des p-figures. La figure et la p-figure vides seront notées respectivement* f_ϵ *et* f'_ϵ.

On remarquera qu'étant donnée la définition d'un dessin pointé vide, il n'y a effectivement qu'une seule classe d'équivalence pour la p-figure vide f'_ϵ.

Lorsqu'il est clair qu'une propriété sur un dessin est invariante par translation, nous dirons que la figure dont ce dessin est un représentant possède la propriété en question : par exemple si e est un dessin connexe et $g = [e]_\perp$, alors nous dirons que g est une figure connexe.

Par souci pratique de faciliter la lecture, nous confondrons souvent à l'usage la classe d'équivalence avec un de ses représentants. Pour les p-figures nous le choisirons, en général, de telle sorte que le point de départ d soit à l'origine de la grille.

Définition 2.3.4 *L'*inverse *d'une p-figure* $f = [e]_\top$ *est la p-figure notée* $\overline{f} = [\overline{e}]_\top$.

Définition 2.3.5 *La* concaténation *de g et g', deux p-figures, notée g.g', est la p-figure définie par:* $g.g' = \{h.h' \mid h \in g, h' \in g',$ *et* $h.h'$ *définie* $\}$.

La concaténation de deux p-figures est une unique classe d'équivalence.

Elle n'est pas vide : soient les dessins pointés $h = (e, d, a)$ et $h' = (e', d', a')$, si on pose $\delta = a - d'$, on a $t'_\delta(h') \in [h']_\top$ et clairement $h.t'_\delta(h')$ est définie et appartient à $g.g'$. En fait, pour toute p-figure f, si on prend un point p de $\mathbb{Z}^2$, alors on peut toujours trouver un dessin pointé (e, x, p) et un dessin pointé (e', p, x') qui sont représentant de f.

Elle est unique : soient les dessins pointés $h = (e, d, a)$ et $h' = (e', d', a')$ tels que $h.h'$ soit définie, et les dessins pointés $h_1 = (e_1, d_1, a_1)$ et $h'_1 = (e'_1, d'_1, a'_1)$ tels que $h_1.h'_1$ soit définie, avec $h, h_1 \in g$ et $h', h'_1 \in g'$, on a $[h.h']_\top \ni h_1.h'_1$. Examinons $\delta = d_1 - d$ tel que $t'_\delta(h) = h_1$, et $\delta' = d'_1 - d'$ tel que $t'_{\delta'}(h') = h'_1$, comme les deux concaténations sont définies, nous avons $d' = a$ et $d'_1 = a_1$, et aussi $\tau_\delta(a) = a_1 = d'_1 = \tau_\delta(d')$. Nous en déduisons que $\delta = \delta'$ et donc $h_1.h'_1 = t'_\delta(h.h')$.

Remarquons que la p-figure vide est élément neutre pour la concaténation. L'ensemble des p-figures munis de la concaténation a donc une structure de monoïde.

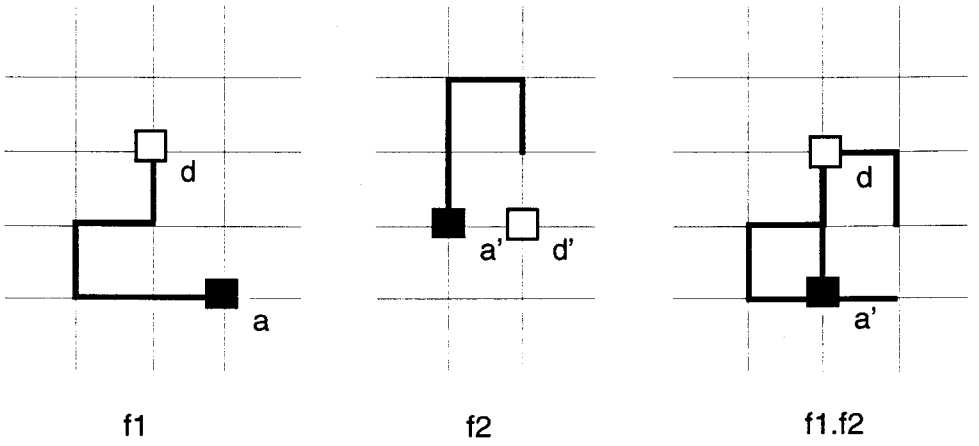


FIG. 2.4 - *Deux p-figures et leur concaténation.*

2.4 Langages de figures

Définition 2.4.1 *Nous appellerons* langage de figures *un ensemble de figures, et* langage de p-figures *un ensemble de figures pointées.*

La concaténation des p-figures nous donne le moyen de définir la classe des langages de p-figures *rationnels*, à la manière de ce qui est fait pour les langages de mots.

Définition 2.4.2 *La concaténation de deux langages de p-figures $\mathcal{F}_1$ et $\mathcal{F}_2$ est définie par :* $\mathcal{F}_1.\mathcal{F}_2 = \{f.g \mid f \in \mathcal{F}_1, g \in \mathcal{F}_2\}$.

Définition 2.4.3 *La puissance $i^{ème}$ d'un langage de p-figures $\mathcal{G}$, pour un entier $i >= 0$, est définie par :* $\mathcal{G}^0 = \{f_\epsilon\}$ *et* $\mathcal{G}^i = \mathcal{G}^{i-1}.\mathcal{G}$ *pour* $i \geq 1$.
L'étoile d'un langage de p-figures est définie par : $\mathcal{G}^* = \bigcup_{i \geq 0} \mathcal{G}^i$

Définition 2.4.4 *La classe des langages de p-figures* rationnels *est définie inductivement par :*

- $\emptyset$ *est un langage de p-figures rationnel.*
- *Soit f une p-figure, $\{f\}$ est un langage de p-figures rationnel.*
- *Si $\mathcal{F}_1$ et $\mathcal{F}_2$ sont des langages de p-figures rationnels, alors $\mathcal{F}_1 \cup \mathcal{F}_2$, $\mathcal{F}_1.\mathcal{F}_2$, et $\mathcal{F}_1^*$ sont des langages de p-figures rationnels.*
- *Il n'existe pas d'autres langages de p-figures rationnels que ceux donnés par cette définition.*

Définition 2.4.5 *Un langage de figures $\mathcal{K}$ est* rationnel *si et seulement si il existe un langage de p-figures rationnel $\mathcal{K}'$ tel que $\mathcal{K} = Base(\mathcal{K}')$.*

Pour donner une vision plus complète, et notamment faire le lien avec [MRW82] où seules les figures connexes étaient abordées, nous définissons également les langages de p-figures rationnels-connexes.

Définition 2.4.6 *La classe des langages de p-figures* rationnels-connexes *est définie inductivement par:*

- $\emptyset$ *est un langage de p-figures rationnel-connexe.*
- *Soit f une p-figure connexe, $\{f\}$ est un langage de p-figures rationnel-connexe.*
- *Si $\mathcal{F}_1$ et $\mathcal{F}_2$ sont des langages de p-figures rationnels-connexes, alors $\mathcal{F}_1 \cup \mathcal{F}_2$, $\mathcal{F}_1.\mathcal{F}_2$, et $\mathcal{F}_1^*$ sont des langages de p-figures rationnels-connexes.*
- *Il n'existe pas d'autres langages de p-figures rationnels-connexes que ceux donnés par cette définition.*

Définition 2.4.7 *Un langage de p-figures $\mathcal{K}$ est* rationnel-connexe *si et seulement si il existe un langage de p-figures rationnel-connexe $\mathcal{K}'$ tel que $\mathcal{K} = Base(\mathcal{K}')$.*

Nous notons $\mathcal{F}_{\mathrm{Rat}}$ la famille des langages de figures rationnels et $\mathcal{F}_{\mathrm{RatC}}$ la famille des langages de figures rationnels-connexes.

2.5 Figures et graphes

Ainsi que nous l'avons remarqué dans l'introduction, il est fréquent de rencontrer des problèmes portant sur une figure considérée comme un graphe non dirigé. Nous aurons aussi recours à ce point de vue pour la rédaction de certaines preuves, ce qui nous amène à définir la notion de *graphe associé* à un dessin. L'idée en est simple: les points de son armature seront les nœuds du graphe, les segments en seront, grosso modo, les arêtes.

Définition 2.5.1 *Le* graphe associé *à un dessin d est le graphe non dirigé, noté $G_d = (N, A)$, où l'ensemble des nœuds est une partie de $\mathbb{Z}^2$ définie par $N = Arm(d)$ et l'ensemble des arêtes est inclus dans $N \times N$ défini par $A = \{\{i, j\} \mid Seg(i, j) \in d\}$.*

Nous dirons qu'un dessin vérifie une propriété sur les graphes si son graphe associé vérifie cette propriété: par exemple le dessin de la Fig. 2.5 est eulérien. Les propriétés qui nous intéressent sur les graphes sont habituellement conservées par translation. C'est pourquoi nous dirons qu'une figure vérifie une propriété sur les graphes si le graphe associé à l'un de ses représentants vérifie cette propriété. Pour les p-figures et les dessins pointés, nous dirons qu'ils vérifient une propriété sur les graphes si leur base la vérifie.

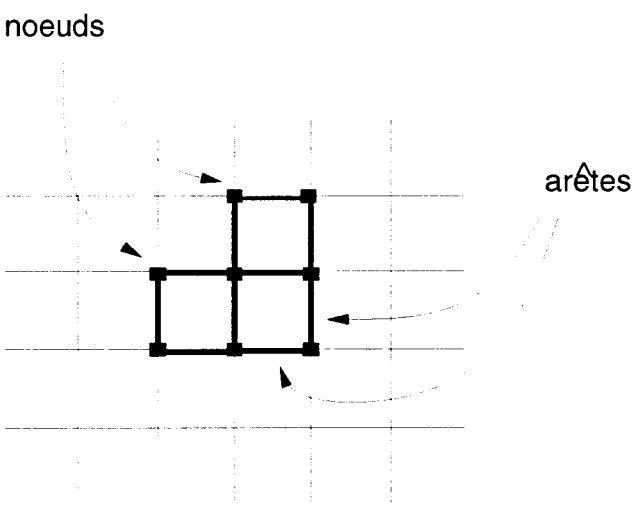


FIG. 2.5 - *Dessin vu comme un graphe.*

Chapitre 3

Mots de figures

Après avoir défini au chapitre précédent les objets graphiques auxquels nous nous intéressons, nous présentons ici leur autre facette : leur description par des mots, où chaque lettre correspond à une instruction élémentaire de dessin dans la métaphore de la table traçante.

Dans une première partie nous rappellerons le formalisme, que nous qualifierons de "classique", proposé par Maurer *et al.* [MRW82], puis, dans la partie suivante, nous abordons la représentation d'objets non obligatoirement connexes. Pour ce faire nous sommes alors amené à modifier la nature des instructions associées aux symboles de l'alphabet, voire à rajouter de nouvelles lettres : c'est ce que nous appellerons changer de sémantique.

3.1 Mots de figures classiques

Nous représentons les figures par des mots sur l'alphabet à quatre lettres $\Pi = \{u, r, l, d\}$. Chaque lettre de Π code un déplacement de longueur unitaire sur le plan $\mathbb{Z}^2$. La lettre u, respectivement r, d, l, code un déplacement vers le haut, respectivement vers la droite, le bas, la gauche (de l'anglais up, right, down, left). En d'autre termes, on considère le mot comme une suite de commandes envoyées à une table traçante : le crayon est placé tout d'abord, par convention, à l'origine et on le déplace selon les directions indiquées par les lettres du mot. Lorsque l'on passe d'un point du plan à un de ses voisins, on dessine le segment qui correspond à l'arête parcourue (Fig. 3.2). Notons qu'un segment parcouru plusieurs fois ne se distingue pas d'un segment dessiné une seule fois.

Définition 3.1.1 *Soit l'alphabet* $\Pi = \{u, r, d, l\}$. *Un* mot de figure *est un mot fini sur* Π, *et un* langage de mots de figures *est un ensemble de mots de figures. Pour faciliter la rédaction de certaines définitions ou preuves, nous remplacerons parfois chaque lettre de* Π *par* x_i, *avec* i *dans* $\mathbb{Z}/4\mathbb{Z}$, *selon la convention suivante :* $u = x_0$, $r = x_1$, $d = x_2$ *et* $l = x_3$.

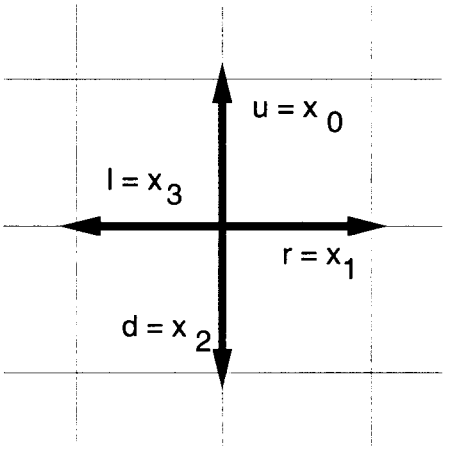


FIG. 3.1 - *Lettres de* Π.

Définition 3.1.2 *Le* décalage *associé à un mot de figure* μ*, noté* $Dec(\mu)$*, est un point de* $\mathbb{Z}^2$ *donné par le morphisme de* $(\Pi, ., \varepsilon)$ *dans* $(\mathbb{Z}^2, +, (0,0))$ *défini par:*

$$\begin{cases} u & \mapsto & (0,+1) \\ r & \mapsto & (+1,0) \\ d & \mapsto & (0,-1) \\ l & \mapsto & (-1,0) \end{cases}$$

Par exemple le décalage associé à $\mu = uurrld$ est : $(0,+1)+(0,+1)+(+1,0)+(+1,0)+(-1,0)+(0,-1) = (+1,+1)$. Nous avons en fait $\mathrm{Dec}(\mu) = (|\mu|_r - |\mu|_l, |\mu|_u - |\mu|_d)$.

Définition 3.1.3 *Le* dessin *associé à un mot de figure* μ*, noté* $Des(\mu)$*, est défini récursivement par:*

$Des(\varepsilon) = \emptyset$
$Des(x_i) = \{Seg((0,0), Dec(x_i))\}$
$Des(\mu x_i) = Des(\mu) \cup t_{m,n}(Des(x_i))$ *avec* $(m,n) = Dec(\mu)$

Soit $\mathcal{L}$ *un ensemble de mots de figures, alors on note* $Des(\mathcal{L}) = \{Des(\mu) \mid \mu \in \mathcal{L}\}$.

Nous dirons aussi dessin *représenté par* un mot à la place de dessin *associé à* un mot. De même pour les définitions qui vont suivre.

Définition 3.1.4 *Le* dessin pointé *associé à un mot de figure* μ*, noté* $Desp(\mu)$*, est le triplet* $Desp(\mu) = (Des(\mu), (0,0), Dec(\mu))$*. C'est à dire le triplet constitué du dessin, et des points de départ et d'arrivée du tracé (Fig. 3.2). Soit* $\mathcal{L}$ *un ensemble de mots de figures, alors on note :* $Desp(\mathcal{L}) = \{Desp(\mu) \mid \mu \in \mathcal{L}\}$.

Définition 3.1.5 *La* figure *associée à un mot de figure* μ *est* $Fig(\mu) = [Des(\mu)]_\perp$*, la* p-figure *associée à* μ *est* $p\text{-}Fig(\mu) = [Desp(\mu)]_\top$*. Soit* $\mathcal{L}$ *un ensemble de mots*

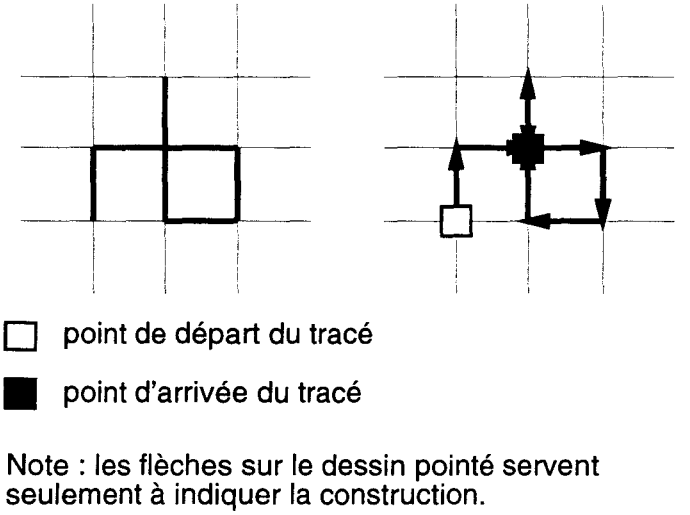


FIG. 3.2 - *Dessin et dessin pointé associés au mot urrdluud.*

de figures, alors on note $Fig(\mathcal{L}) = \{Fig(\mu) \mid \mu \in \mathcal{L}\}$ *et* $p\text{-}Fig(\mathcal{L}) = \{p\text{-}Fig(\mu) \mid \mu \in \mathcal{L}\}$.

On remarquera que, pour un mot μ, $\mathrm{Des}(\mu)$ et $\mathrm{Desp}(\mu)$ définissent respectivement un représentant bien particulier des classes d'équivalence $\mathrm{Fig}(\mu)$ et $\mathrm{p\text{-}Fig}(\mu)$. Ces représentants sont tels que, pour deux mots μ et ν, on a $\mathrm{p\text{-}Fig}(\mu) = \mathrm{p\text{-}Fig}(\nu) \Rightarrow \mathrm{Desp}(\mu) = \mathrm{Desp}(\nu)$, mais par contre $\mathrm{Fig}(\mu) = \mathrm{Fig}(\nu) \not\Rightarrow \mathrm{Des}(\mu) = \mathrm{Des}(\nu)$, comme le montre le contre-exemple $\mu = ll$ et $\nu = rll$, où :

$$\begin{aligned} \mathrm{Des}(\mu) &= \{\mathrm{Seg}((0,0),(-1,0)), \mathrm{Seg}((-1,0),(-2,0))\} \\ \mathrm{Des}(\nu) &= \{\mathrm{Seg}((0,0),(1,0)), \mathrm{Seg}((0,0),(-1,0))\} \end{aligned}$$

Par définition, on établit immédiatement la correspondance entre la concaténation des mots de figures et celle des dessins et des figures pointés.

Fait 3.1.6 *Soient* μ *et* μ' *des mots de figures, avec* $(x,y) = Dec(\mu)$. *On a* $Desp(\mu\mu') = Desp(\mu)\ .t'_{x,y}(Desp(\mu'))$ *et* $Des(\mu\mu') = Base(Desp(\mu\mu'))$.
Soient $f = p\text{-}Fig(\mu)$ *et* $f' = p\text{-}Fig(\mu')$ *deux p-figures, on a* $f.f' = p\text{-}Fig(\mu\mu')$.

Comme montré dans [MRW82], ce fait nous permet de déduire le résultat suivant.

Fait 3.1.7 *Un langage de figures* $\mathcal{L}$ *est rationnel-connexe si et seulement si il existe* $\mathcal{R}$, *un langage rationnel de mots de figures sur* Π, *tel que* $\mathcal{L} = Fig(\mathcal{R})$.

Par moments nous aurons besoin, pour un mot μ traçant une p-figure f, d'un mot μ' qui trace l'inverse de f. Comme les lettres sont symétriques deux à deux,

nous définissons la notion correspondante de mot inverse, à ne pas confondre avec le mot miroir.

Définition 3.1.8 *Le mot* inverse *d'un un mot de figure μ, noté $\overline{\mu}$, est défini récursivement comme suit :*

$$\begin{aligned} \overline{\varepsilon} &= \varepsilon \\ \overline{u} &= d \\ \overline{r} &= l \\ \overline{d} &= u \\ \overline{l} &= r \\ \overline{(\mu_1\mu_2)} &= (\overline{\mu_2})(\overline{\mu_1}) \end{aligned}$$

avec μ_1, μ_2 des mots de figures.

La notion d'inverse dans les mots est bien sûr compatible avec celle définie sur les p-figures au chapitre précédent : on a en effet p-Fig$(\overline{\mu})$ = $\overline{\text{p-Fig}(\mu)}$.

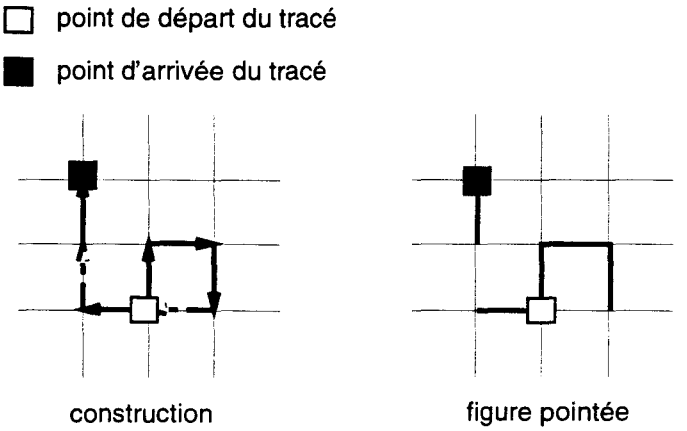
On peut faire aussi la remarque suivante: soit μ un mot de figure, alors Fig(μ) = Fig$(\overline{\mu})$, mais au contraire on n'a pas Des(μ) = Des$(\overline{\mu})$, p-Fig(μ) = p-Fig$(\overline{\mu})$, Desp(μ) = Desp$(\overline{\mu})$, sauf si Dec$(\mu) = (0,0)$.

3.2 Une sémantique pour la non-connexité

Dans la section précédente, nous avons rappelé précisément les notions de langages de mots de figures, telles quelles ont été introduites par Maurer *et al.*. Le lecteur n'aura pas manqué de remarquer que cette définition des mots de figures ne permet pas de décrire des objets non connexes (ni d'ailleurs d'envisager de les représenter avec plusieurs couleurs). On peut cependant remédier à cet inconvénient sans remettre fondamentalement en cause les idées que nous venons de présenter. Nous allons compléter notre alphabet par des symboles qui, dans la métaphore de la table traçante, correspondent à des intructions de lever et de baisser de crayon, ou à leur équivalent.

Diverses propositions se sont faites jour. Nous rappelons d'abord celle de Hinz et Welzl (voir [HW88]), où les mots sont écrits sur un alphabet étendu par des symboles correspondant aux différentes couleurs employées : un mot sous forme normale *alternative* se compose alors d'une suite de lettres où alternent successivement une couleur puis une direction. Par exemple, si @ et # codent respectivement les couleurs bleue et rouge, un exemple de mot sur $\Pi \cup \{@, \#\}$ en forme normale alternative est donné par $\mu = @u@r\#d\#l$, qui trace un carré où les côtés gauche et dessus sont en bleu, les côtés droit et dessous en rouge.

La lourdeur de ce système conduit à utiliser couramment des mots qui ne sont pas en forme normale : une fois qu'une couleur est spécifiée, elle reste valable tant qu'aucune autre spécification n'est donnée. Une indication de couleur peut alors être suivie de plusieurs lettres consécutives indiquant des directions qui toutes seront exécutées avec cette même couleur. Pour les figures non connexes, une



Les déplacements effectués crayon levé sont indiqués par des tirets dans le schéma de gauche. Cette p-figure peut être associée aux mots :

- sur Π_f : $urd \uparrow l \downarrow l \uparrow u \downarrow u$
- sur Π_i : $urdl'lu'u$

FIG. 3.3 - *Exemple de tracé avec lever de crayon.*

version de ce système est donnée qui utilise deux couleurs seulement, noir et "invisible", avec des mots qui ne sont pas sous forme normale et elle est aussi reprise dans [Hin88]. Les mots sont donc écrits sur l'alphabet $\Pi_f = \{u, r, d, l, \uparrow, \downarrow\}$, où les symboles $\uparrow$ et $\downarrow$ correspondent aux indications "passer en couleur invisible" et "passer en couleur noire", c'est à dire encore "lever le crayon" et "baisser le crayon". Le principe de description des dessins ne change pas par rapport à ce que nous avions vu dans les sections précédentes, sinon que les actions de déplacement effectuées crayon levé n'ajoutent pas de nouveaux segments. Un exemple est donné en Fig. 3.3.

Telle que nous l'avons sommairement présentée, la sémantique de ce dernier système reste assez floue. En effet, lorsqu'un mot commence par des instructions de tracé et sans indication de couleur, c'est à dire s'il appartient à $\Pi^+\Pi_f^*$ comme celui de l'exemple de la Fig. 3.3, il est nécessaire d'adopter une convention : le crayon est-il levé ou baissé au départ du tracé? Dans l'exemple nous avons considéré que le crayon était baissé. Toutefois, quelle que soit la convention choisie, la concaténation des mots n'est plus similaire à celle des figures. On peut le constater avec les deux mots $\mu = r \uparrow$ et $\nu = r$, où on a clairement $\mathrm{Fig}(\mu\nu) \neq \mathrm{Fig}(\mu).\mathrm{Fig}(\nu)$.

Pour éviter cet inconvénient le recours à la forme normale alternative s'impose. Cette dernière n'est pas, à notre avis, sans présenter des inconvénients. Outre son manque de concision, certaines notions habituelles de la théorie des langages perdent leur adéquation : les facteurs d'un mot sous forme normale ne

sont pas tous sous forme normale, le "shuffle" de deux formes normales n'en est pas une.

Une autre façon de procéder est suggérée dans [MRW82], et a été étudiée plus en détail notamment par K. Slowinski (voir [Slo90, Slo93]). Elle consiste à ajouter, à chaque lettre codant un déplacement, une nouvelle lettre qui correspond au même déplacement crayon levé, sans tracé. Nous représentons par des lettres primées ces nouveaux symboles de notre alphabet, lequel devient : $\Pi_i = \{u, r, d, l, u', r', d', l'\}$. Sur l'exemple de la Fig. 3.3, un mot possible pour la p-figure est $\mu = urdl'lu'u$.

C'est cette sémantique que nous avons choisie, car elle évite les problèmes mentionnés plus haut (elle constitue en fait une notation plus compacte de la forme normale alternative). Avant d'exposer plus avant les particularités qui la caractérisent, il convient d'en donner une définition plus formelle. Pour éviter des répétitions inutiles, nous ne précisons pour ces nouveaux mots de figures que les points qui diffèrent des définitions données dans la section précédente. Nous n'avons pas changé les termes employés. En effet, dans la suite du mémoire le contexte indiquera clairement si l'on parle de mots sur Π ou sur Π_i : le lecteur se reportera aux définitions correspondantes.

Définition 3.2.1 *Soit l'alphabet* $\Pi_i = \{u, r, d, l, u', r', d', l'\}$. *Un* mot de figure *est un mot fini sur* Π_i, *et un* langage de mots de figures *est un ensemble de mots de figures. Pour faciliter la rédaction de certaines définitions ou preuves, nous remplacerons parfois chaque lettre de* Π *par* x_i *ou* x'_i, *avec* i *dans* $\mathbb{Z}/4\mathbb{Z}$, *selon la convention suivante :* $u = x_0$, $r = x_1$, $d = x_2$, $l = x_3$ *d'une part et* $u' = x'_0$, $r' = x'_1$, $d' = x'_2$, $l' = x'_3$ *d'autre part.*

Définition 3.2.2 *Le* décalage *associé à un mot de figure* μ, *noté* $Dec(\mu)$, *est un point de* $\mathbb{Z}^2$ *donné par le morphisme de* $(\Pi_i, ., \varepsilon)$ *dans* $(\mathbb{Z}^2, +, (0,0))$ *défini par :*

$$\begin{cases} u & \mapsto & (0,+1) \\ u' & \mapsto & (0,+1) \\ r & \mapsto & (+1,0) \\ r' & \mapsto & (+1,0) \\ d & \mapsto & (0,-1) \\ d' & \mapsto & (0,-1) \\ l & \mapsto & (-1,0) \\ l' & \mapsto & (-1,0) \end{cases}$$

Définition 3.2.3 *Le* dessin *associé à un mot de figure* μ, *noté* $Des(\mu)$, *est défini récursivement par :*

$$\begin{array}{lcl} Des(\varepsilon) & = & \emptyset \\ Des(x_i) & = & \{Seg((0,0), Dec(x_i))\} \\ Des(x'_i) & = & \emptyset \\ Des(\mu x_i) & = & Des(\mu) \cup t_{m,n}(Des(x_i)) \text{ avec } (m,n) = Dec(\mu) \\ Des(\mu x'_i) & = & Des(\mu) \end{array}$$

Définition 3.2.4 *Le mot* inverse *d'un un mot de figure μ, noté $\overline{\mu}$, est défini récursivement comme suit :*

$$\begin{aligned}
\overline{\varepsilon} &= \varepsilon \\
\overline{u} &= d \\
\overline{r} &= l \\
\overline{d} &= u \\
\overline{l} &= r \\
\overline{u'} &= d' \\
\overline{r'} &= l' \\
\overline{d'} &= u' \\
\overline{l'} &= r' \\
\overline{(\mu_1\mu_2)} &= (\overline{\mu_2})(\overline{\mu_1}) \\
&\quad \textit{avec } \mu_1, \mu_2 \textit{ des mots de figures.}
\end{aligned}$$

Les autres définitions relatives aux mots de figures restent valables avec notre nouvelle définition du dessin. Une fois définie notre sémantique, nous pouvons faire quelques remarques à son sujet.

Fait 3.2.5 *Il existe une fonction rationnelle ρ qui transforme tout mot de figure μ sur Π_f en un mot $\omega = \rho(\mu)$ sur Π_i, tel que $Desp(\mu) = Desp(\omega)$. De même, il existe une fonction rationnelle ρ' qui transforme tout mot de figure μ sur Π_i en un mot $\omega = \rho'(\mu)$ sur Π_f, tel que $Desp(\mu) = Desp(\omega)$.*

Nous concluons que les deux formalismes présentés ici pour les dessins non connexes décrivent bel et bien les mêmes objets. La démonstration du fait 3.2.5, de même que les transducteurs dont il est question ainsi qu'une étude plus approfondie des relations entre sémantique sur Π_f et sur Π_i sont détaillés dans un mémoire de D. Simplot [Sim93].

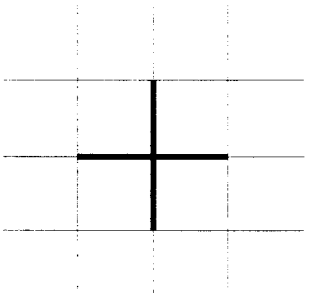
On remarquera aussi le fait suivant, qui fait écho à la fin de la section précédente, et se déduit de l'équivalence entre concaténation de figures non nécessairement connexe et concaténation des mots sur Π_i

Fait 3.2.6 *Un langage de figures (respectivement p-figures) $\mathcal{L}$ est rationnel si et seulement si il existe $\mathcal{R}$, un langage rationnel de mots de figures sur Π_i, tel que $\mathcal{L} = Fig(\mathcal{R})$ (respectivement $\mathcal{L} = p\text{-}Fig(\mathcal{R})$).*

3.3 Définitions complémentaires

Nous regroupons ici quelques définitions communes à toutes les sémantiques dont il est fait mention dans cet ouvrage, et qui cernent certaines classes de mots plus particulièrement intéressantes.

Définition 3.3.1 *Un mot de figure μ est une* boucle *si et seulement si $Dec(\mu) = (0,0)$. On appelle $\mathcal{B}$ l'ensemble des boucles.*



non minimal udrllrd

minimaux uudlrr

ulrrlu

urllru

(parmi d'autres possibles)

FIG. 3.4 - *Figure et mots minimaux.*

Notons que dans le cas des mots sur Π, le mot μ est une boucle si et seulement si $|\mu|_r = |\mu|_l$ et $|\mu|_u = |\mu|_d$.

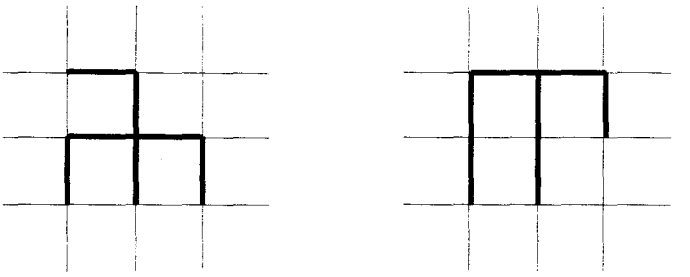
Comme nous l'avons remarqué plus haut, on peut tracer plusieurs fois un même segment sans que cela change le dessin obtenu. Par conséquent il existe une infinité de mots de figures décrivant un même dessin. Par exemple le dessin constitué d'un unique segment horizontal est décrit par chaque mot du langage infini $r(lr)^*$. Cette constatation nous conduit à définir le concept de mot le plus court représentant un dessin donné. Cette notion est, bien évidemment, dépendante de la méthode de description choisie, et nous verrons un peu plus en détail comment elle peut évoluer lorsque nous changeons la sémantique qui détermine les actions associées aux lettres de l'alphabet, comme montré en Fig. 3.5 où les deux figures de même poids ont des mots minimaux de longueurs différentes sur Π mais identiques sur Π_i. À chaque sémantique nous associerons un alphabet de nom différent pour éviter les ambiguïtés, et nous indiquerons la sémantique dont il est question par le nom de l'alphabet associé.

Définition 3.3.2 *Un mot de figure est* minimal *sur l'alphabet Ψ si et seulement si il n'existe pas de mot plus court représentant la même figure. Plus formellement : μ minimal sur $\Psi \Leftrightarrow (\forall \nu \in \Psi^*, |\nu| < |\mu| \Rightarrow Fig(\nu) \neq Fig(\mu))$.*

Définition 3.3.3 *Un mot de figure est* p-minimal *sur l'alphabet Ψ si et seulement si il n'existe pas de mot plus court représentant la même p-figure. Plus formellement : μ p-minimal sur $\Psi \Leftrightarrow (\forall \nu \in \Psi^* \Rightarrow |\nu| < |\mu|, p\text{-}Fig(\nu) \neq p\text{-}Fig(\mu))$.*

Notons qu'en général il n'y a pas unicité des mots minimaux (l'inverse d'un mot minimal est aussi minimal, à l'exception toutefois de ce qui concerne la sémantique vue dans la dernière partie de la thèse). L'unicité est parfois vérifiée, par contre, pour les mots p-minimaux, et ce quelle que soit la sémantique.

La longueur minimale d'un mot représentant une figure donnée dépend non seulement de la taille, en nombre de segments, du dessin, mais aussi de sa structure. Idéalement, on aimerait qu'à chaque segment corresponde une lettre du mot,



- Exemples de mots minimaux *sans* lever de crayon: figure de gauche *rdldurdurd* ($\mathrm{Com}(f,\Pi) = 10/7$), figure de droite *uurddrrrd* ($\mathrm{Com}(f',\Pi) = 9/7$).
- Exemples de mots minimaux *avec* lignes invisibles: figure de gauche *rddl'urrd* ($\mathrm{Com}(f,\Pi_i) = 8/7$), figure de droite *ulldd r'uu* ($\mathrm{Com}(f',\Pi_i) = 8/7$).

FIG. 3.5 - *Figures et complexité descriptive.*

mais on constate sur les exemples très simples qui sont donnés plus haut que cela n'est pas toujours possible, et en effet on doit assez souvent coder un mouvement du crayon destiné à rejoindre une partie de la figure qui n'a pas encore été tracée. Ainsi, que l'on puisse lever le crayon ou pas, un certain nombre de lettres d'un mot minimal ne tracent rien (ou rien de nouveau). Pour traduire cette notion, une mesure de l'efficacité avec laquelle une figure se prête à une représentation sous forme de mot est donnée dans [MRW82] sous le nom de complexité descriptive (nous rappelons que $|f|$ est la taille de la figure en nombre de segments unitaires).

Définition 3.3.4 *Soit f une figure non vide, la* complexité descriptive *de f sur l'alphabet Ψ est : $Com(f,\Psi) = |\mu|/|f|$, où μ est un mot minimal sur l'alphabet Ψ représentant f.*

Soit f' une p-figure non vide, la complexité descriptive *de f' sur l'alphabet Ψ est : $Com(f',\Psi) = |\mu|/|f'|$, où μ est un mot p-minimal sur l'alphabet Ψ représentant f'.*

Bien sûr, il faut, dans notre formalisme et quelle que soit la sémantique, au moins une lettre pour tracer chaque segment de la figure (ou chaque pixel dans la dernière partie de la thèse). Par conséquent la complexité descriptive de toute figure non vide est supérieure ou égale à 1.

Deuxième partie

Langages classiques et avec lever de crayon

Chapitre 1

Mots minimaux sur Π_i

Dans ce chapitre nous nous intéressons aux mots minimaux sur Π_i, c'est à dire à ceux qui peuvent contenir des lettres "invisibles".

1.1 Figures connexes

1.1.1 Motivations

Consacrer une section à parler de figures connexes représentées par des mots avec lever de crayon peut sembler paradoxal. En effet, la motivation première qui a conduit à introduire la sémantique de Π_i réside dans l'impossibilité de décrire des figures non-connexes avec les mots à la Maurer *et al.* Cependant l'intérêt de cette représentation excède ce cas particulier et concerne aussi la description des figures connexes (*cf* [HW88, Sim93]). Un exemple frappant nous est donné par le langage de figures connexes $\mathcal{L}_1$ défini par le langage de mots sur Π donné par la grammaire linéaire G_1 ci-dessous.

$$G_1 : \{ S \rightarrow dSu \mid r \}$$

Les premières figures de ce langages sont montrées en Fig. 1.1. Alors qu'il n'existe pas de langage rationnel $\mathcal{K}$ sur Π tel que $\mathrm{Fig}(\mathcal{K}) = \mathcal{L}_1$, au contraire il en

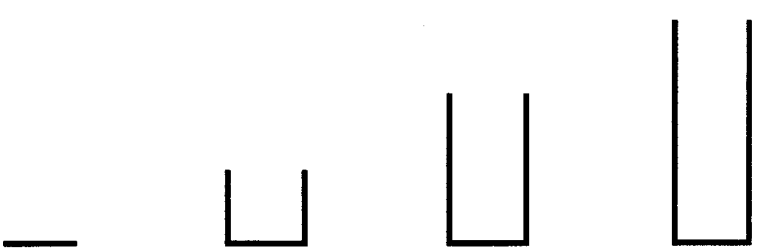


FIG. 1.1 - *Quelques figures de* $\mathcal{L}_1$.

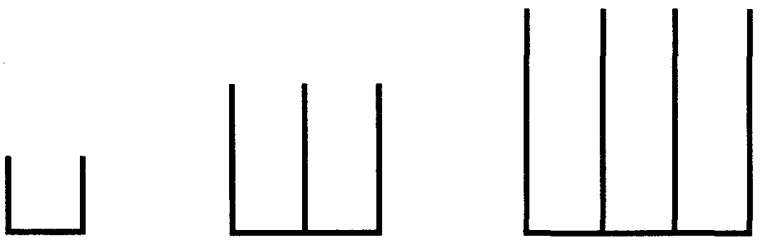


FIG. 1.2 - *Quelques figures de* $\mathcal{L}_2$.

existe sur Π_i, par exemple le langage $r(l'ur'du)^*$.

Fait 1.1.1 *Par conséquent la famille des langages de figures rationnels-connexes est incluse mais ne coïncide pas avec la famille des langages de figures rationnels qui ne contiennent que des figures connexes.*

Un deuxième avantage se révèle lors de la recherche de mots minimaux. Nous avons montré dans la section 3.3, Fig. 3.5, deux exemples de figures où les mots minimaux étaient plus courts sur l'alphabet Π_i. Le gain obtenu en changeant de système peut être plus important comme le montre un autre exemple, tiré de [MRW82]. Soit le langage de figures connexes $\mathcal{L}_2$ défini par $\mathcal{L}_2 = \mathrm{Fig}(\mathcal{K}_2)$ avec $\mathcal{K}_2$ le langage de mots sur Π donné par $\mathcal{K}_2 = \{\omega_n = (d^n r u^n)^n \mid n \in \mathbb{N}\}$. Des exemples de figures de $\mathcal{L}_2$, pour $n \leq 3$, sont donnés en Fig. 1.2.

Notons f_n la figure décrite par ω_n pour un n donné. Le poids de f_n vaut n^2+n. Si on cherche un mot minimal par la méthode du postier chinois (*cf.* [Bra89, GM84]), on s'aperçoit que le mot ω_n est minimal. Par ailleurs sa longueur vaut $2n^2 + n$, ce qui nous donne pour complexité descriptive $\lim_{n\to\infty} \mathrm{Com}(f_n, \Pi) = 2$.

Soit le mot sur l'alphabet Π_i, $\mu_n = d^n(ru^n r' d^n lr)^{n/2}$ si n est pair, $\mu_n = d^n(ru^n r' d^n lr)^{(n-1)/2} ru^n$ si n est impair. On a clairement $\mathrm{Fig}(\mu_n) = f_n$, et, bien sûr, la longueur de μ_n, qui vaut $n^2 + 3n$ ou $n^2 + 3n - 1$ selon la parité de n, majore celle d'un mot minimal pour f_n, et donc on calcule facilement que, en ce qui concerne la complexité descriptive sur Π_i, le résultat précédent devient : $\lim_{n\to\infty} \mathrm{Com}(f_n, \Pi_i) = 1$.

1.1.2 Mot minimal sur Π_i

Le problème de l'obtention d'un mot minimal représentant un dessin connexe est tout à fait traitable.

Sur l'alphabet Π, chercher un mot minimal correspondant au dessin d, c'est chercher sur le graphe associé un parcours de longueur minimale où chaque arête soit parcourue au moins une fois (pour la tracer). Comme la remarqué Brandenburg [Bra89], ceci n'est autre que l'énoncé du "problème du postier chinois" dont on sait trouver une solution en temps polynomial, voir [EJ73, GM84, Lee90b]. Nous ne donnons pas ici une présentation formelle et détaillée de ce résultat connu,

ce qui serait hors de notre propos, mais nous nous contenterons d'en rappeler le principe, en dirigeant le lecteur vers [GM84] pour une explication détaillée.

En résumé, la solution consiste à ajouter au graphe des arêtes reliant entre eux les nœuds d'arité impaire qui sont surnuméraires, de façon à obtenir un graphe eulérien, c'est à dire qui n'a pas plus de deux nœuds d'arité impaire. Les arêtes ajoutées doivent dupliquer des arêtes existantes, pour ne pas changer la figure, et elles doivent être choisies pour que leur poids cumulé soit minimal, ce que l'on sait obtenir par un algorithme de couplage parfait minimal, qui fonctionne en temps polynomial. Notons que si le graphe est eulérien dès le départ, il suffit alors de trouver un parcours eulérien, ce que l'on sait faire facilement. Remarquons enfin que l'algorithme esquissé ci-dessus construit par défaut un parcours qui est un circuit (*i.e.* une boucle). Si il n'est pas nécessaire de revenir au point de départ, on ajoute une arête fictive de poids nul entre le nœud de départ et celui d'arrivée. Cette dernière remarque s'applique immédiatement dans le cas de la recherche d'un mot p-minimal, où la donnée du problème comporte effectivement les deux points de départ et d'arrivée. Dans le cas où l'on veut un mot minimal et non plus p-minimal, on calcule, pour chaque paire de nœuds, le poids du couplage parfait minimal où ces deux nœuds sont considérés comme point de départ et point d'arrivée, et on opte alors pour la sélection qui donne le couplage de poids le plus faible. L'algorithme ainsi modifié s'effectue toujours en temps polynomial.

Le problème est très similaire pour la recherche de mots minimaux sur l'alphabet Π_i, et est donc une variante du problème du postier chinois, à la différence près que les arêtes ajoutées au graphe peuvent être invisibles, et n'ont donc pas à dupliquer des arêtes existantes. On peut donc les choisir les plus directes possibles, par rapport à notre métrique, par exemple en les décrivant par des mots inclus dans $\{(u'^* + d'^*)(r'^* + l'^*)\}$. Le principe du couplage, quant à lui, ne change pas.

Nous utiliserons cet algorithme lors du prochain chapitre.

1.2 Figures non connexes

En ce qui concerne les figures non connexes, nous constatons qu'il est facile de générer des situations *ad hoc* où la représentation par un mot de figure n'est, à l'évidence, pas adaptée. Considérons par exemple l'ensemble des figures $\mathcal{A} = \{f_n = [\{\mathrm{Seg}((0,0),(1,0)), \mathrm{Seg}((n,0)(n+1,0))\}]_\perp \mid n \in \mathbb{N}^+\}$. Toutes ces figures sont composées uniquement de deux segments, mais cependant la description par un mot sur Π_i (ou Π_f) du motif f_n nécessite au moins $n+1$ lettres, ce qui n'est pas un modèle d'économie. Cela ne porte toutefois pas préjudice à l'utilisation de ce système pour les cas raisonnables.

Nous avons déjà souligné que l'un des points importants pour l'efficacité de la représentation de dessins par des mots réside dans la possibilité d'obtenir des descriptions de longueur minimale. Dans le cas de dessins non connexes, il n'est

hélas pas aisé de générer de telles descriptions, comme il est montré ci-après.

Proposition 1.2.1 *Construire pour toute figure non connexe, un mot minimal sur Π_i qui la représente, est un problème NP-dur.*

Preuve. Nous utilisons un résultat de Papadimitriou [Pap77] qui énonce que le problème du voyageur de commerce rectilinéaire (noté par la suite PVCR) est NP-complet.

Le PVRC s'énonce comme suit : les n clients d'un voyageur de commerce sont répartis dans n villes différentes, dont les coordonnées sont prises dans le plan $\mathbb{Z}^2$. Dans quel ordre le voyageur doit-il visiter les villes pour que la distance rectilinéaire totale parcourue soit minimale? La donnée formelle du problème se réduit à la liste des coordonnées des cités.

Nous allons montrer qu'à chaque instance du PVCR on peut associer une figure de telle sorte que trouver un mot minimal pour la figure nous donne la solution du PVCR, les transformations à effectuer se faisant en temps polynomial.

Soit $n \in \mathbb{N}$ et $p = \{c_0, \ldots, c_{n-1}\}, p \subset \mathbb{Z}^2$ une instance du PVCR. Nous admettons sans perte de généralité que $n > 2$. Appelons $s = (s_0, \ldots, s_{n-1})$ la suite des indices des villes qui caractérise une des solutions de p, et $\delta(i,j)$ la distance rectilinéaire entre la ville d'indice i et la ville d'indice j, et enfin $\Delta = \sum_{0 \leq i < n-2} \delta(s_i, s_{i+1})$ la distance totale associée à la solution s.

Nous construisons un PVCR dérivé $p' = \{h(c_0), \ldots, h(c_{n-1})\}$ en appliquant l'homothétie :

$$h = \begin{cases} \mathbb{Z}^2 & \to & \mathbb{Z}^2 \\ (x,y) & \mapsto & (4nx, 4ny) \end{cases}.$$

Puis nous obtenons une figure f non connexe en prenant chacune des coordonnées de p' comme l'extrémité gauche d'un segment unitaire : $f = \{[\mathrm{Seg}(h(c_0), h(c_0) + (1,0)), \ldots, \mathrm{Seg}(h(c_{n-1}), h(c_{n-1}) + (1,0))]_\perp\}$.

Soit ω un mot minimal décrivant f, généré par un algorithme de minimalisation noté $\mathcal{AM}$. Nous admettons pour la clarté de l'explication, sans perte de généralité, que pour chaque segment de f seule la première lettre qui le trace est une lettre de Π (autrement dit, une lettre visible). On peut en effet remplacer les autres lettres qui tracent ce segment par le déplacement invisible qui leur correspond : cela ne change ni la figure, ni la taille du mot. Le mot ω est donc sous la forme : $\omega = a_{v_0}\beta_{v_0v_1}a_{v_1}\beta_{v_1v_2}\ldots\beta_{v_{n-2}v_{n-1}}a_{v_{n-1}}$, où chaque a_{v_i} est une lettre de Π qui trace le segment dont l'extrémité gauche est la ville d'indice v_i du problème dérivé p', et chaque $\beta_{v_iv_{i+1}}$ est un mot invisible qui relie le segment tracé par a_{v_i} à celui tracé par $a_{v_{i+1}}$.

Nous allons minorer la taille de ω. Les a_{v_i} apportent une contribution de n lettres. Les extrémités de chaque $\beta_{v_iv_{i+1}}$ sont éloignées au plus d'un segment des villes d'indice v_i et v_{i+1} dans p', et donc on obtient : $|\beta_{v_iv_{i+1}}| \geq 4n\delta(v_i, v_{i+1}) - 2$ (cette majoration pourrait être affinée, mais elle nous suffira telle quelle pour la suite). Par conséquent on obtient : $|\omega| \geq n + 4n\sum_{0 \leq i < n-2} \delta(v_iv_{i+1}) - 2(n-1)$, ou

plus clairement :

$$|\omega| \geq 4n \sum_{0 \leq i < n-2} \delta(v_i v_{i+1}) - n + 2 \tag{1.1}$$

Le mot ω étant minimal, on peut majorer sa taille par celle de n'importe quel mot ω' qui trace f. Construisons ω' en respectant l'ordre des villes donné par la solution s, par exemple : $\omega' = rl\mu_{s_0 s_1} rl\mu_{s_1 s_2} \ldots \mu s_{n-2} s_{n-1} rl$, où chaque doublet de lettres rl trace un des segments de la figure, et chaque mot $\mu_{s_i s_{i+1}}$ est un chemin invisible le plus direct allant de la ville d'indice s_i à celle d'indice s_{i+1}. On obtient donc :

$$|\omega| \leq |\omega'| = 4n\Delta + 2n \tag{1.2}$$

Raisonnons par l'absurde, et supposons que la suite des indices $(v_0, \ldots, v_{n-1})$ donnée par une lecture du mot minimal ω ne caractérise pas une solution de p. Alors on a : $\sum_{0 \leq i < n-2} \delta(v_i v_{i+1}) \geq \Delta + 1$. Nous en déduisons une contradiction, d'après les equations 1.1 et 1.2 :

$$\begin{aligned} |\omega| &\geq 4n\Delta + 3n + 2 \\ |\omega| &\leq 4n\Delta + 2n \end{aligned}$$

En conclusion, une lecture d'un mot minimal pour f, qui se fait en temps linéaire relativement à la longueur du mot et donc en temps polynomial si l'algorithme $\mathcal{AM}$ fonctionne en temps polynomial, nous donne une solution de p. □

Chapitre 2

Un résultat de complexité descriptive

Dans ce chapitre nous allons nous intéresser à un résultat de Maurer *et al.* portant sur la complexité descriptive sur Π des figures connexes, et montrer qu'un résultat similaire est obtenu sur l'alphabet Π_i. Par défaut, les figures que nous utiliserons seront toutes connexes.

2.1 Complexité descriptive et sémantique

La notion de complexité descriptive a été introduite dans [MRW82], où est montré le résultat suivant :

Fait 2.1.1 *1. Pour toute figure non vide f, on a :*

$$1 \leq Com(f, \Pi) < 2$$

2. Quelque soit le réel ρ, avec $0 < \rho < 1$, il existe une figure f telle que :

$$Com(f, \Pi) > 2 - \rho$$

Cependant, lorsque l'on choisit de représenter les figures par des mots sur Π_i, la complexité descriptive peut chuter dans des proportions importantes, comme nous l'avons souligné au chapitre précédent. Remarquons au passage que, en ce qui concerne les figures connexes, on peut les décrire par des mots sur Π_i qui n'utilisent que les lettres de Π, et donc :

Fait 2.1.2 *Quelque soit une figure non vide f, on a :*

$$1 \leq Com(f, \Pi_i) \leq Com(f, \Pi) < 2$$

Une question naturelle posée à ce propos par G. Păun de l'institut mathématique de Bucarest ([Pău90]) est la suivante :

> L'usage de déplacements invisibles permet souvent d'obtenir des représentation plus courtes, pourtant existe-t-il des figures connexes telles que leur complexité descriptive sur Π_i soit aussi proche de 2 qu'on le souhaite ?

Après un travail effectué en commun avec G. Păun et K. Slowinski, nous répondons par l'affirmative, en construisant une famille de figures qui vérifie cette propriété. La fin de ce chapitre est consacrée à la démonstration de ce résultat dont on peut trouver une version plus primitive dans [PRS92]. Nous l'avons affiné dans cet ouvrage, et nous l'avons complété par une remarque sur les familles de langages concernés par ces problèmes.

2.2 La famille des figures f_n

Nous nous proposons de construire un mot de figure à l'aide du TAG-système constitué des substitutions h_1 et h_2 :

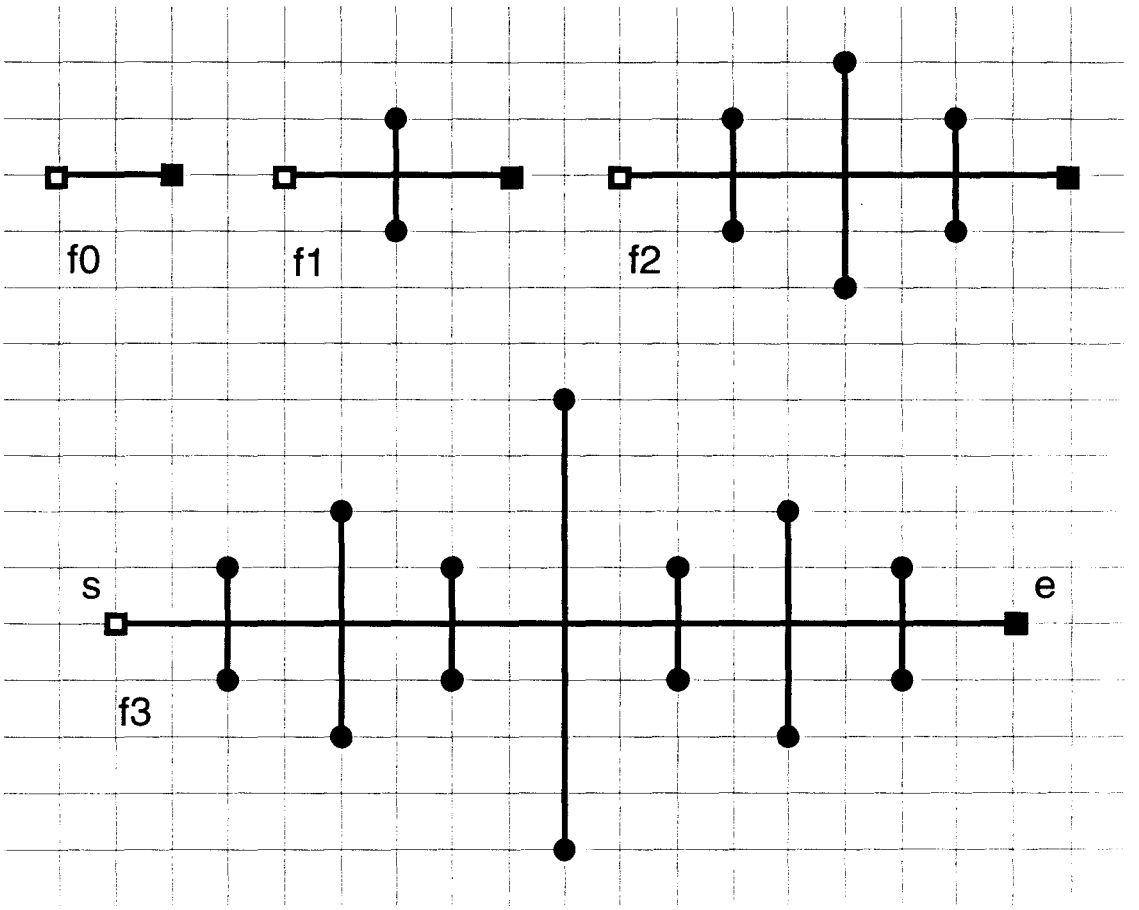
$$\begin{aligned} h_1 \ : \ & \{X,U,D\}^* \to \{X,U,D\}^* \\ & X \mapsto XUD^2UX \\ & U \mapsto U^2 \\ & D \mapsto D^2 \end{aligned}$$

$$\begin{aligned} h_2 \ : \ & \{X,U,D\}^* \to \Pi_i^* \\ & X \mapsto r^2 \\ & U \mapsto u \\ & D \mapsto d' \end{aligned}$$

Nous noterons $h_1^n(X)$ le résultat de l'itération n fois de la substitution h_1 appliquée à X.

Définition 2.2.1 *Quelque soit l'entier positif n, on note ω_n le mot défini par $\omega_n = h_2(h_1^n(X))$, on note $e_n = (d_n, s, e)$ le dessin pointé décrit par ω_n, et f_n la figure représentée par ω_n.*

Nous allons mettre en évidence et quantifier certaines caractéristiques de la figure f_n (voir Fig. 2.1) qui nous seront nécessaires par la suite pour pouvoir minorer la taille d'un mot minimal. Nous notons s et e respectivement le point de départ et d'arrivée du tracé de e_n. Nous dirons que, d'une manière naïve, la forme générale de f_n est celle d'un peigne, la poignée étant constituée de l'unique ligne horizontale, et les dents du peigne étant les lignes verticales. Nous désignerons par la suite ces lignes verticales et elles seules sous le terme de *barres*.



Les nœuds d'arité impaire sur le graphe associé au dessin e_n correspondant, sont signalés par un rond (les points de départ et d'arrivée du tracé correspondent aussi à des nœuds d'arité impaire).

FIG. 2.1 - *Les premières figures de la famille* f_n.

Dans un premier temps, nous voulons calculer le nombre de segments unitaires verticaux et horizontaux de e_n pour connaître le poids de la figure. Pour les segments verticaux, nous isolons les facteurs construits par application de h_1, notés V_i, et ceux obtenus après avoir appliqué h_2, notés v_i, qui génèrent les barres de la figure.

Définition 2.2.2 *Nous notons V_i et v_i les facteurs qui génèrent les barres dans la figure f_n.*

$$\begin{aligned} \forall i > 0, \quad V_i &= U^{2^{i-1}} D^{2^i} U^{2^{i-1}} \\ v_i &= h_2(V_i) = u^{2^{i-1}} d'^{2^i} u^{2^{i-1}} \quad . \end{aligned}$$

Proposition 2.2.3 *Les mots V_i et ω_n vérifient les propriétés suivantes :*

1. $\forall i > 0, h_1(V_i) = V_{i+1}$
2. $h_1(X) = XV_1X$
3. $|\omega_n|_r = 2^{n+1}$
4. $|\omega_n|_u = |\omega_n|_{d'} = n2^n$
5. $\omega_n \in r^2[(v_1 + \cdots + v_n)r^2]^*$

Preuve. Les points 1 à 3 sont évidents.

Point 4. Clairement $|\omega_n|_u = |\omega_n|_{d'}$. Faisons une récurrence sur n: $|w_0|_u = 0$, nous supposons la propriété vraie pour $n = k$, $k \geq 0$.

$$\begin{aligned} |w_{k+1}|_u &= 2|w_k|_u + |w_k|_r \\ &= k2^{k+1} + 2^{k+1} \\ &= (k+1)2^{k+1} \end{aligned}$$

Point 5. Par récurrence sur n: $w_0 = r^2$, nous supposons la propriété vraie pour $n = k$, $k \geq 0$.

$$\begin{aligned} & w_k \in r^2[(v_1 + \cdots + v_k)r^2]^* \\ \Leftrightarrow \quad & w_k \in h_2(X[(V_1 + \cdots + V_k)X]^*) \\ \Leftrightarrow \quad & w_{k+1} \in h_2(XV_1X[(V_2 + \cdots + V_{k+1})XV_1X]^*) \\ & (\text{ par définition de } \omega_n \text{ points 1 et 2}) \\ \Leftrightarrow \quad & w_{k+1} \in h_2(X[(V_1 + \cdots + V_{k+1})X]^*) \\ \Leftrightarrow \quad & w_{k+1} \in r^2[(v_1 + \cdots + v_{k+1})r^2]^* \end{aligned}$$

□

Corollaire 2.2.4 *Il y a $\mathcal{H}_n = 2^{n+1}$ segments unitaires horizontaux et $\mathcal{V}_n = n2^n$ segments unitaires verticaux dans le dessin e_n.*

Preuve. D'après le point 5 de la prop. 2.2.3, nous déduisons que les points 3 et 4 recensent bien la totalité des segments du dessin e_n.□

Afin de trouver un couplage parfait de poids minimal, dans le cadre du déroulement de l'algorithme du postier chinois, nous devons définir formellement l'ensemble des nœuds d'arité impaire du graphe associé au dessin pointé e_n. Nous cherchons pour l'instant un mot p-minimal, aussi nous considérons les points s et e du dessin comme étant reliés par une arête fictive de poids nul, conformément à l'algorithme présenté lors du chapitre précédent. Leur arité devient paire, et donc nous ne les prendrons pas en considération dans le couplage.

Proposition 2.2.5 *L'ensemble des nœuds d'arité impaire du graphe G_{e_n}, associé à e_n et complété par l'arête fictive (s,e), est :*

$$\begin{aligned}\mathcal{I} \;=\; & \{(x,y) \mid (x,y) = Dec(x_1 a) \text{ avec } \omega_n = x_1 a b x_2, \\ & ab \in \{ud', d'u\}, x_1, x_2 \in \Pi_i^*\}\end{aligned}$$

Preuve. Évident par construction de la figure.□

Nous définissons la hauteur d'un nœud de $\mathcal{I}$ comme le nombre de segments unitaires qui le séparent de l'axe horizontal de la figure.

Définition 2.2.6 *Soit P un nœud de $\mathcal{I}$, nous dirons que $Hauteur(P) = k$ si P est dans $\{Dec(x_1 u^{2^{k-1}}), Dec(x_1 u^{2^{k-1}} d'^{2^k})\}$ avec $\omega_n = x_1 v_k x_2$ et x_1, x_2 dans Π_i^*.*

Notre couplage va faire intervenir des couples de nœuds d'arité impaire qui sont situés symétriquement par rapport à l'axe horizontal de la figure. Nous les caractérisons ici.

Définition 2.2.7 *Soit $P = (x,y)$ dans $\mathcal{I}$, alors nous notons $\underline{P} = (x,-y)$.*

Proposition 2.2.8 *Soit P dans $\mathcal{I}$ tel que $Hauteur(P) = k$, alors :*

1. *$\underline{P}$ est dans $\mathcal{I}$*
2. *$Hauteur(\underline{P}) = Hauteur(P)$*
3. *$Dist(P, \underline{P}) = 2^k$*

*Preuve.*Évident par construction de la figure.□

2.3 Le mot ω_n est p-minimal

Nous voulons montrer qu'un couplage de poids minimal sur les nœuds d'arité impaire est obtenu en appariant les nœuds opposés définis en 2.2.7. Pour cela nous devons minorer la distance entre nœuds n'étant pas sur la même barre verticale.

Proposition 2.3.1 *La distance horizontale entre deux barres quelconques de la figure f_n a une borne inférieure qui est fonction de la taille des barres:*

$$\forall n, i, j \in I\!N, \forall w \in \Pi_i^* \text{ tels que } v_i w v_j \in Fac(\omega_n), \text{ alors } |w|_r \geq 2^{min(i,j)} \ .$$

Preuve. La proposition 2.3.1 est conséquence immédiate de la propriété suivante : $\forall n, i, j \in \mathbb{N}, \forall W \in \{X, U, D\}^*$ tels que $V_i W V_j \in \text{Fac}(h_1^n(X))$, alors $|W|_X \geq 2^{min(i,j)-1}$.

Récurrence sur n :

La propriété est vraie pour $n = 0, n = 1$. Nous la supposons vraie pour $n = p - 1$, avec $p \geq 1$. Soit $h_1^p(X) = W_1 V_i W V_j W_2$, avec $\{W_1, W, W_2\} \subset \{X, U, D\}^*$.

- Cas 1 : $i = 1$ ou $j = 1$. De la proposition 2.2.3 nous déduisons que $|W|_X \geq 1$.
- Cas 2 : $i > 1$ et $j > 1$. De la proposition 2.2.3 nous déduisons qu'il existe une décomposition : $h_1^{p-1}(X) = W_1' V_{i-1} W' V_{j-1} W_2'$ telle que $h_1(W_1') = W_1$ et $h_1(W') = W$ et $h_1(W_2') = W_2$. Alors :

$$\begin{cases} |W'|_X \geq 2^{min(i-1,j-1)-1} & \text{(par hypothèse de récurrence)} \\ |h_1(W')|_X \geq 2|W'|_X & \text{(d'après la proposition 2.2.3)} \end{cases}$$
$$\Rightarrow \quad |W|_X \geq 2 \times 2^{min(i-1,j-1)-1} \geq 2^{min(i,j)-1}$$

□

Corollaire 2.3.2 *Soit A_k et $A'_{k'}$ deux nœuds dans $\mathcal{I}$, de hauteur respective k et k', alors:*

$$Dist(A_k, A'_{k'}) \geq \frac{2^k + 2^{k'}}{2}$$

Preuve. Soit $A_k = (x_{A_k}, y_{A_k})$ et $A'_{k'} = (x_{A'_{k'}}, y_{A'_{k'}})$, nous admettons, sans perte de généralité, que ou bien $x_{A_k} < x_{A'_{k'}}$, ou bien $y_{A_k} > y_{A'_{k'}}$. Il existe une décomposition $\omega_n = \mu_1 v_k \mu_2 v_{k'} \mu_3$, avec $\mu_1, \mu_2, \mu_3 \in \Pi_i^*$ telle que :

$$\begin{cases} v_k = v'_k v''_k \\ v_{k'} = v'_{k'} v''_{k'} \end{cases} \text{ et } \begin{cases} (x_{A_k}, y_{A_k}) = \text{Dec}(\mu_1 v'_k) \\ (x_{A'_{k'}}, y_{A'_{k'}}) = \text{Dec}(\mu_1 v'_k v''_k \mu_2 v'_{k'}) \end{cases}$$

D'après la proposition 2.3.1 nous déduisons :

$$\begin{aligned} dist\,(A_k, A'_{k'}) &\geq |\mu_2|_r + |2^{k-1} - 2^{k'-1}| \\ &\geq 2^{min(k,k')} + |2^{k-1} - 2^{k'-1}| \\ &\geq 2^{k-1} + 2^{k'-1} \\ &\geq \frac{2^k + 2^{k'}}{2} \ . \end{aligned}$$

□

Nous pouvons désormais minorer la distance séparant deux nœuds quelconques de $\mathcal{I}$ en fonction de celle qui les sépare de leurs nœuds opposés respectifs.

Corollaire 2.3.3 *Soit A_k et $A'_{k'}$ deux nœuds dans $\mathcal{I}$, de hauteur respective k et k', alors :*

$$Dist(A_k, A'_{k'}) \geq \frac{Dist(A_k, \underline{A_k}) + Dist(A'_{k'}, \underline{A'_{k'}})}{2} .$$

Nous montrons alors qu'il n'existe pas de couplage parfait de poids plus petit que celui obtenu en appariant les nœuds opposés.

Lemme 2.3.4 *L'ensemble $M = \{(P, \underline{P}) \mid P = (x, y) \in \mathcal{I}, y > 0\}$ est un couplage parfait de poids minimal sur G_{e_n}.*

Preuve. Soient A et B une partition équitable de $\mathcal{I}$, et C un couplage parfait sur $A \times B$. Plus formellement : $A = \{P_1, \cdots, P_q\} \subset \mathcal{I}$, $B = \{P'_1, \cdots, P'_q\} \subset \mathcal{I}$ avec $A \cup B = \mathcal{I}$, $A \cap B = \emptyset$, $Card(A) = Card(B)$, et $C = \{(P_i, P'_i) \in A \times B \mid 1 \leq i \leq q\}$. Alors nous pouvons calculer le poids total du couplage C:

$$\begin{aligned}
\sum_{(P,P')\in C} \mathrm{Dist}(P, P') &= \mathrm{Dist}(P_1, P'_1) + \cdots + \mathrm{Dist}(P_q, P'_q) \\
&\geq \frac{1}{2}[\mathrm{Dist}(P_1, \underline{P_1}) + \mathrm{Dist}(P'_1, \underline{P'_1}) + \cdots \\
&\quad + \mathrm{Dist}\,(P_q, \underline{P_q}) + \mathrm{Dist}(P'_q, \underline{P'_q})] \text{ (d'après le corollaire 2.3.3)}
\end{aligned}$$

Pour faciliter la suite du raisonnement, nous renommons uniformément chacun des termes successifs de cette expression par un Q_i, pour obtenir :

$$\begin{aligned}
\sum_{(P,P')\in C} \mathrm{Dist}(P, P') &\geq \frac{1}{2}[\mathrm{Dist}(Q_1, \underline{Q_1}) + \mathrm{Dist}(Q_2, \underline{Q_2}) + \cdots \\
&\quad + \mathrm{Dist}(Q_{2q-1}, \underline{Q_{2q-1}}) + \mathrm{Dist}(Q_{2q}, \underline{Q_{2q}})]
\end{aligned}$$

Appelons $\mathcal{I}^+$ l'ensemble des nœuds situés au dessus de l'axe horizontal de la figure : $\mathcal{I}^+ = \{(x, y) \in \mathcal{I} \mid y > 0\}$. Remarquons que pour chaque Q_i, avec i dans $[1, \cdots, 2q]$, il y a exactement un seul Q_j, avec j dans $[1, \cdots, 2q]$ tel que $Q_j = \underline{Q_i}$ (et bien sûr $j \neq i$). L'un ou l'autre des Q_i et Q_j est dans $\mathcal{I}^+$. Sans perte de généralité, nous admettons que c'est Q_i qui se trouve dans $\mathcal{I}^+$, et nous pouvons alors remplacer chaque paire de termes $[\mathrm{Dist}(Q_i, \underline{Q_i}) + \mathrm{Dist}(Q_j, \underline{Q_j})]$ par un terme $[2 \times \mathrm{Dist}(Q_i, \underline{Q_i})]$. Nous obtenons :

$$\sum_{(P,P')\in C} \mathrm{Dist}(P,P') \geq \frac{1}{2}[2\mathrm{Dist}(Q_1,\underline{Q_1}) + 2\mathrm{Dist}(Q_2,\underline{Q_2}) + \cdots + 2\mathrm{Dist}(Q_q,\underline{Q_q})]$$

C'est à dire, plus clairement :

$$\sum_{(P,P')\in C} \mathrm{Dist}(P,P') \geq \sum_{P\in\mathcal{I}^+} \mathrm{Dist}(P,\underline{P})$$

□

Par conséquent, tout couplage parfait à un poids supérieur ou égal au couplage M. Nous pouvons alors déduire que :

Corollaire 2.3.5 *Le mot de figure ω_n est p-minimal.*

Preuve. Clairement ω_n décrit un chemin eulérien sur le graphe G_{ϵ_n} complété par l'ajout des arêtes du couplage M. □

2.4 Complexité descriptive des figures f_n

Les résultats précédents nous permettent de calculer la complexité descriptive des figures f_n.

Théorème 2.4.1 *Pour tout réel positif $k < 2$, il existe un naturel n tel que :*

$$Com(f_n, \Pi_i) > k \ .$$

Preuve. Nous avons montré précédemment que le mot ω_n est p-minimal, or nous nous intéressons à la complexité descriptive des figures et non pas des p-figures. Nous allons donc calculer, à partir de la longueur de ω_n, une minoration de la longueur d'un mot minimal (et non pas p-minimal) pour la figure f_n.

Soit α_n un mot minimal décrivant f_n, s' et e' étant les points de départ et d'arrivée de p-Fig(α_n). À partir de α_n nous construisons $\beta_n = m_1\alpha_n m_2$, avec m_1 et m_2 des mots de $\{u', r', d', l'\}^*$, construits pour que m_1 relie s à s', m_2 relie e' à e, et $|m_1| = \mathrm{Dist}(s,s')$, $|m_2| = \mathrm{Dist}(e',e)$ (en d'autres termes, m_1 et m_2 sont choisis aussi courts que possibles, par exemple dans $\{(u'^* + d'^*)(r'^* + l'^*)\}$).

Il est clair que α_n et β_n décrivent la même figure, car nous n'avons ajouté que des lettres invisibles. Nous avons aussi $|\alpha_n| = |\beta_n| - \mathrm{Dist}(s,s') - \mathrm{Dist}(e',e)$. Pour toute paire (X,Y) de points de l'armature de f_n, nous avons $\mathrm{Dist}(X,Y) \leq$

$\text{Dist}(s,e)$ et bien sûr $\text{Dist}(s,e) = \mathcal{H}_n$. Par conséquent nous déduisons que $\text{Dist}(s,s') \leq \mathcal{H}_n$ et $\text{Dist}(e',e) \leq \mathcal{H}_n$. D'un autre côté, comme ω_n est p-minimal, nous déduisons : $|\beta_n| \geq |\omega_n|$, c'est à dire $|\beta_n| \geq \mathcal{H}_n + 2\mathcal{V}_n$. Nous concluons :

$$\begin{aligned} |\alpha_n| &\geq |\beta_n| - 2\mathcal{H}_n \\ &\geq 2\mathcal{V}_n - \mathcal{H}_n \end{aligned}$$

et

$$\begin{aligned} \lim_{n\to\infty} \text{Com}(f_n, \Pi_i) &= \lim_{n\to\infty} \frac{|\alpha_n|}{|f_n|} \\ &\geq \lim_{n\to\infty} \frac{2\mathcal{V}_n - \mathcal{H}_n}{\mathcal{H}_n + \mathcal{V}_n} \\ &\geq \lim_{n\to\infty} \frac{(2n-2)2^n}{(n+2)2^n} \\ &= 2 \end{aligned}$$

□

2.5 Compléments

Si on note $\mathcal{K}$ le langage des mots ω_n générés par notre TAG-système, nous avons donc montré la propriété suivante :

$$\forall k \in \mathbb{R}, k < 2, \exists w \in \mathcal{K} \text{ tel que } \text{Com}(\text{Fig}(w), \Pi_i) > k \tag{2.1}$$

En fait nous avons démontré une propriété plus forte, que l'on peut présenter de manière imagée en disant "plus les mots sont grands, plus leur complexité est grande" :

$$\forall k \in \mathbb{R}, k < 2, \exists n \in \mathbb{N}, \forall w \in \mathcal{K}, |w| > n \Rightarrow \text{Com}(\text{Fig}(w), \Pi_i) > k \tag{2.2}$$

On remarquera que ces résultats, valables pour la complexité sur Π_i, sont aussi vérifiés *ipso facto* pour la complexité sur Π.

Le langage $\mathcal{K}$ faisant partie de la famille des langages EOL, on peut se demander s'il existe des langages qui vérifient la dernière propriété 2.2, et sont dans des familles plus simples. En ce qui concerne la propriété 2.1, ce serait un faux problème car, bien sûr, elle est vraie pour le langage $\Pi_i^* \supset \mathcal{K}$. Nous allons montrer que la propriété 2.2 n'est pas vérifiée pour les langages de mots de figures algébriques, en montrant l'existence d'une borne supérieure, strictement plus petite que 2, pour la complexité descriptive d'un sous-ensemble infini. Nous travaillerons

sur des mots sans lettres invisibles : en effet, de l'existence d'une borne supérieure pour la complexité sur Π se déduit immédiatement le même résultat sur Π_i.

Théorème 2.5.1 *Quelque soit $\mathcal{L} \subset \Pi^*$, un langage algébrique infini, alors :*

$$\exists k \in I\!R, k < 2, \exists \mathcal{L}' \subset \mathcal{L}, \mathcal{L}' \text{ infini, tel que } \forall w \in \mathcal{L}', Com(Fig(w), \Pi_i) < k \quad (2.3)$$

Preuve. Dans la suite nous noterons $\min_f$ un mot minimal sur Π décrivant la figure f. Par définition, nous avons donc : $\mathrm{Com}(f, \Pi) = \frac{|\min_f|}{|f|}$.

Soit $\mathcal{L} \subset \Pi^*$, un langage algébrique infini. D'après le lemme de pompage nous savons alors qu'il existe $\{\alpha, \beta, \gamma, \delta, \epsilon\} \subset \Pi^*$, avec $(\beta, \delta) \neq (\varepsilon, \varepsilon)$, tel que $\mathcal{L}' = \{\mu_n = \alpha\beta^n\gamma\delta^n\epsilon \mid n \in \mathbb{N}, n \geq 1\}$ soit inclus dans $\mathcal{L}$. Soit f_n la figure associée au mot μ_n, nous allons montrer qu'il existe une borne strictement inférieure à 2 et qui majore la complexité descriptive de f_n quelque soit n.

Nous traitons deux cas :

Cas 1 : β et δ sont des boucles (*i.e.* $\mathrm{Dec}(\beta) = \mathrm{Dec}(\delta) = (0,0)$). Alors $\mathrm{Fig}(\mathcal{L}') = \mathrm{Fig}(\alpha\beta\gamma\delta\epsilon)$, et donc $\mathrm{Fig}(\mathcal{L}')$ est un langage fini dont chaque figure vérifie bien sûr le fait 2.1.1, et donc $\mathcal{L} \supset \mathcal{L}'$ vérifie évidemment le théorème 2.5.1.

Cas 2 : β et/ou δ ne sont pas des boucles : sans perte de généralité nous admettrons que $\mathrm{Dec}(\beta) \neq (0,0)$, le raisonnement étant similaire dans l'autre cas. Il existe un rang n à partir duquel chaque nouvelle itération de β et de δ ajoute au moins un segment supplémentaire à la figure (et au plus $|\beta| + |\delta|$) :

$$n + c_1 \leq |f_n| \text{ avec } c_1 \text{ une constante.}$$

Considérons le dessin pointé $e_n = (\mathrm{Des}(\mu_n), (0,0), \mathrm{Dec}(\alpha\beta^n))$. Par construction nous avons $[e_n]_\perp = f_n$. Nous savons ([Slo92], p.75–76) qu'il existe ν_n un mot *réduit* qui décrit e_n, et où chaque segment est parcouru deux fois, à l'exception d'un chemin allant de l'origine à l'arrivée du tracé et dont les segments ne sont parcouru qu'une seule fois. La longueur de ce chemin est ici linéairement proportionnelle à $n\mathrm{Dec}(\beta)$. Nous pouvons déduire une majoration de la longueur de ν_n :

$$|\nu_n| \leq 2|f_n| - c_2 n + c_3 \text{ avec } c_2 > 0, c_3 \text{ des constantes.}$$

Comme $\min_{f_n}$ et ν_n décrivent la même figure, on a : $|\min_{f_n}| \leq |\nu_n|$.

Nous pouvons alors évaluer la complexité descriptive de f_n :

$$\begin{aligned}
\mathrm{Com}(f_n, \Pi) &= \frac{|\min_{f_n}|}{|f_n|} \\
&\leq \frac{2|f_n| - c_2 n + c_3}{|f|} \\
&\leq 2 - \frac{c_2 n}{|f|} + \frac{c_3}{|f|} \\
&\leq 2 - \frac{c_2 n}{n + c_1} + \frac{c_3}{n + c_1}
\end{aligned}$$

et nous concluons :

$$\lim_{n\to\infty} \mathrm{Com}(f_n, \Pi) \leq 2 - c_2 \text{ avec } c_2 > 0$$

□

Chapitre 3

Questions de décidabilité

L'emploi de mots pour décrire les figures permet, comme nous l'avons vu, d'en esquisser une classification basée sur la hiérarchie de Chomsky, et donne accès à la "boîte à outils" de la théorie des langages. Le besoin s'en fait particulièrement sentir pour aborder certains problèmes de décidabilité, auxquels nous consacrons ce chapitre.

3.1 Aspects méthodologiques

Nous nous sommes focalisés sur trois résultats connus, assez récents, et qui prouvent l'indécidabilité de l'existence, dans un langage de figures d'une famille donnée, d'une figure caractérisée par une propriété géométrique donnée.

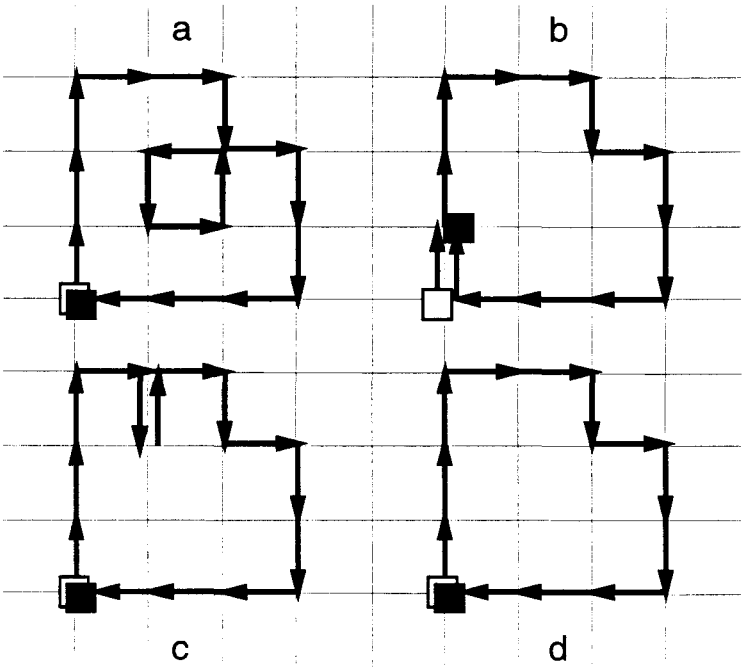
Les trois problèmes en question sont les suivants :

1. Peut-on décider si un langage rationnel sur Π contient un mot de figure décrivant un contour de polyomino [Bea91] ?

2. Peut-on décider si un langage rationnel sur Π contient un mot de figure décrivant un arbre [DH93] ?

3. Peut-on décider si un langage rationnel sur Π_i contient un mot de figure décrivant une figure connexe [DH93] ?

Avant d'aller plus loin, précisons le sens de la question 1. Nous allons parler plus en détail de polyominos dans la section 1.1 de la dernière partie de la thèse, où nous présentons de nombreux exemples. Pour l'instant, nous entendrons par polyomino, toute union finie non vide de carrés unitaires de la grille modélisée par $\mathbb{Z}^2$, union qui doit être 4-connexe et sans trous de surcroît. Nous décrivons une telle figure par un mot représentant son contour, c'est à dire sa frontière. Pour être considéré comme une description valide, chaque segment de la frontière ne



Dessin a : contre-exemple $u^3r^2dldrurd^2l^3$.

Dessin b : contre-exemple $u^3r^2drd^2l^3u$.

Dessin c : contre-exemple $u^3rdurdrd^2l^3$.

Dessin d : exemple $u^3r^2drd^2l^3$.

FIG. 3.1 - *Exemple et contre-exemples de mot de contour.*

doit être tracé qu'une seule fois par le mot de contour. Plus formellement :

Définition 3.1.1 *Un mot de figure est un* mot de contour *s'il est une boucle et si aucun de ses facteurs propres n'est une boucle.*

Cette définition est illustrée par la Fig. 3.1.

Dans le cas de la question 2, nous dirons qu'un mot de figure μ décrit un arbre si et seulement si le graphe associé à Des(μ) est homomorphe à un arbre, c'est à dire est un graphe sans cycles.

L'énoncé de la question 3 ne nécessite pas, par contre, d'explications supplémentaires.

Dans la littérature, la démonstration de l'indécidabilité de ces problèmes s'obtient en mettant un place un codage entre une figure et un calcul d'une machine de Turing. La construction de ce codage peut nécessiter des détours techniques assez tortueux. Par ailleurs, un bon nombre de problèmes indécidables voisins de ceux-ci (voir notamment [Das88, Das89]), ont été démontrés par la méthode

éprouvée qui consiste à se ramener au problème de correspondance de Post, dont nous rappelons le principe plus bas. Cette méthode, assez simple et directe, n'a été employée, à notre connaissance, que pour s'attaquer aux familles de langages linéaires ou algébriques, tant il est vrai qu'elle s'y prête particulièrement bien.

Nous avons cherché à mettre en œuvre cette technique dans le cas des langages rationnels, et nous nous sommes attachés à donner ici une nouvelle preuve pour démontrer l'indécidabilité de ces trois questions, en favorisant une approche uniforme et, nous l'espérons, simplifiée. De ce résultat nous déduisons aussi l'indécidabilité du problème suivant, posé par J. Berstel [Pol91] :

> Peut-on décider si un langage rationnel sur Π contient un mot de figure minimal?

3.2 Une variante du problème de Post

Les preuves des résultats suivants sont basées sur l'indécidabilité du problème de correspondance de Post (noté PCP), dont nous rappelons ici la forme habituelle.

Soit l'alphabet $\mathcal{A} = \{0, 1\}$ et $U = \{u_1, u_2, \ldots, u_n\}$, $V = \{v_1, v_2, \ldots, v_n\}$ deux ensembles de mots de $\mathcal{A}^*$, de cardinal n supérieur à 9 et tels que $\forall i$, $1 \leq i \leq n$, $u_i \neq \varepsilon$ et $v_i \neq \varepsilon$.

Fait 3.2.1 *On ne peut pas décider de l'existence d'une suite* $i_1, i_2, \ldots, i_k$, *avec* $k \geq 1$, *telle que :*

$$u_{i_1} u_{i_2} \ldots u_{i_k} = v_{i_1} v_{i_2} \ldots v_{i_k}$$

On peut dégager de cet énoncé une méthode assez générale pour prouver des résultats d'indécidabilité dans les langages linéaires. Ce schéma, mis en œuvre dans [Das88], consiste, en partant d'une instance d'un problème de Post, à associer un dessin (en fait un mot de figure) à chacun des u_i et v_i, puis à générer par une grammaire linéaire toutes les suites possibles de ces mots de figures, lesquels vont tracer deux motifs, dessinés en vis à vis sur la grille, de telle sorte que la propriété dont on souhaite montrer l'indécidabilité n'apparaisse que s'il existe une correspondance possible dans le PCP. Un exemple va clarifier cette technique : nous allons montrer l'indécidabilité de l'existence d'un mot de contour de polyomino convexe dans un langage linéaire sur Π. Ce résultat, inédit mais cependant prévisible, se montre sans difficulté, et nous a été utile pour compléter le tableau sur les problèmes de décision présenté dans l'introduction.

Théorème 3.2.2 *On ne peut pas décider si un langage linéaire sur* Π *contient un mot décrivant un contour de polyomino convexe.*

Preuve. Habituellement, on passe des u_i, v_i aux mots de figures par deux morphismes h et g de $\mathcal{A}^*$ dans Π^*. Dans le cas présent, nous pouvons trouver un seul

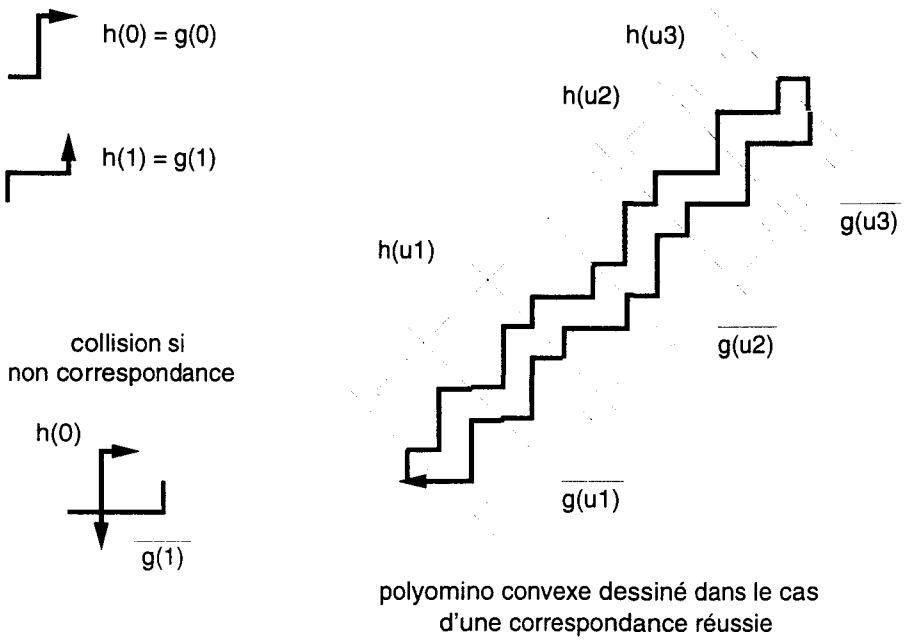


FIG. 3.2 - *Une application du problème de Post.*

morphisme $h = g$ qui convienne à la preuve. Nous emploierons tout de même les deux symboles h et g, pour illustrer le principe général de cette méthode. Donc, ici :

$$\begin{cases} h(0) = ruur \\ h(1) = urru \end{cases} \text{ et } \begin{cases} g(0) = h(0) \\ g(1) = h(1) \end{cases}$$

La grammaire qui génère notre langage sera définie par $G = [\{S, S_1\}, \Pi, S, R]$, l'ensemble des règles R étant, pour tout $1 \leq i \leq n$:

$$\begin{aligned} S &\rightarrow uS_1l \\ S_1 &\rightarrow h(u_i)S_1\overline{g(v_i)} \\ S_1 &\rightarrow h(u_i)rd\overline{g(v_i)} \end{aligned}$$

On notera que $\{h(0), h(1)\}$ et $\{g(0), g(1)\}$ sont des codes. Si on prend l'exemple suivant : $u_1 = 0010$, $u_2 = 1$, $u_3 = 1$, $v_1 = 00$, $v_2 = 101$ et $v_3 = 1$. Alors la suite $i_1 = 1$, $i_2 = 2$, $i_3 = 3$ établit effectivement une correspondance entre u_i et v_i. On constate en Fig. 3.2 que le mot de figure généré pour cette suite d'indice décrit un contour de polyomino convexe. S'il n'y avait pas correspondance, alors un cycle serait créé par l'alignement des motifs $h(0)$ et $g(1)$ ou bien $h(1)$ et $g(0)$, comme montré en bas à gauche du schéma.

Donc, par construction, le langage généré par G ne contient un mot représentant un contour de polyomino convexe que si et seulement si il existe une correspondance pour le PCP, ce que nous voulions montrer. □

La possibilité nous est offerte, par une grammaire linéaire, de générer, de manière symétrique, les motifs associés aux u_i et v_i, avec pour seul détour technique l'emploi dans les règles de production de l'inverse de $g(v_i)$, noté $\overline{g(v_i)}$, pour respecter l'ordre de tracé des segments. Cette facilité disparaît lorsque l'on passe aux langages rationnels. Il est cependant possible de concevoir une méthode construite sur les mêmes principes de base, mais qui s'accommode de cette restriction importante. Pour la mettre en œuvre, nous aurons besoin d'une variante plus contrainte du PCP, utilisée notamment dans [Sal81, LT90], et qui s'énonce :

Soit l'alphabet $\mathcal{A} = \{0, 1\}$ et $U = \{u_1, u_2, \ldots, u_n\}$, $V = \{v_1, v_2, \ldots, v_n\}$ deux ensembles de mots de $\mathcal{A}^*$, de cardinal n supérieur à 9 et tels que $\forall i$, $1 \leq i \leq n$, $u_i \neq \varepsilon$ et $v_i \neq \varepsilon$.

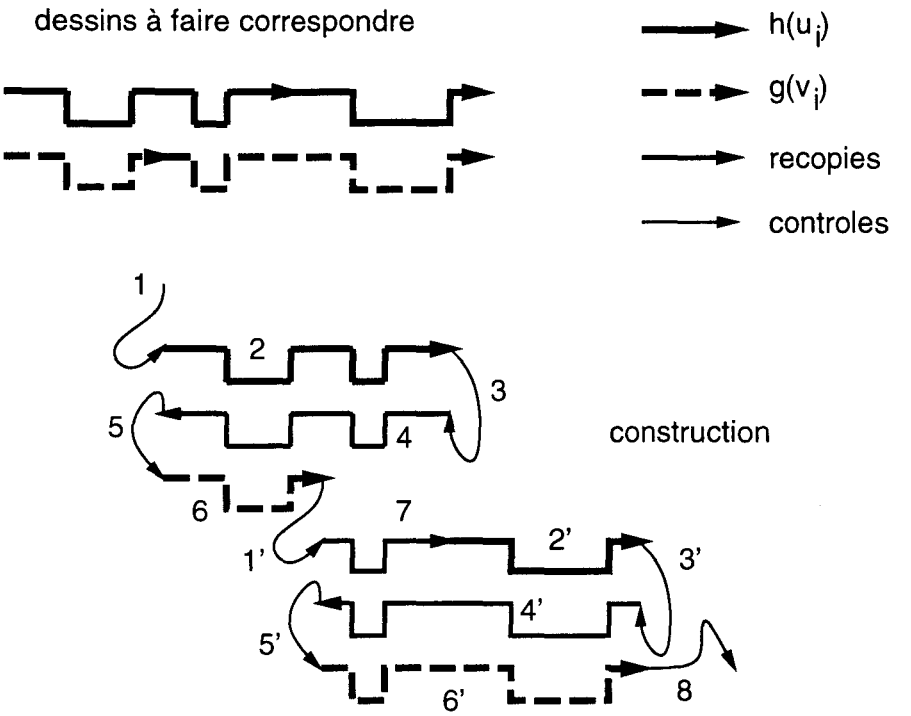
Fait 3.2.3 *On ne peut pas décider de l'existence d'une suite $i_1, i_2, \ldots, i_k$, avec $k \geq 1$, telle que :*

$$u_{i_1}u_{i_2}\ldots u_{i_k} = v_{i_1}v_{i_2}\ldots v_{i_k} \text{ et } |u_{i_1}u_{i_2}\ldots u_{i_m}| \geq |v_{i_1}v_{i_2}\ldots v_{i_m}|, \ 1 \leq m \leq k$$

Nous allons expliquer le principe de la construction que nous proposons, et nous conseillons au lecteur de se référer au schéma de la Fig. 3.3 au fur et à mesure qu'il avance dans les détails. Nous ne donnons pas l'explication d'un cas particulier de question indécidable, mais plutôt nous définissons le "patron", générique, d'une expression rationnelle. Dans l'écriture de cette expression nous avons recours à huit facteurs qui seront constitutifs des mots du langage, qui correspondent chacun à une étape logique bien définie dans notre méthodologie. Il faudra par la suite instancier ces facteurs pour obtenir, pour chaque preuve, un langage adapté, c'est à dire un langage qui possède un mot ayant la propriété dont on veut montrer le caractère indécidable, si et seulement si le PCP possède une solution.

Comme dans la méthode des langages linéaires, nous allons transformer par morphisme les mots du PCP en mots de figures. Il ne nous est désormais plus possible d'utiliser les propriété de la grammaire pour empiler les mots qui doivent se correspondre. Nous allons donc les générer par paire successives à l'aide d'une expression rationnelle bâtie, en essence, sur le modèle suivant : $(\biguplus_{i=1}^{n} h(u_i)\mu g(v_i)\nu)^*$, où le mot de figure μ sert à positionner le crayon pour dessiner $g(v_i)$ en dessous de $h(u_i)$ (parties 3, 4 et 5 sur le schéma). En fait, lors du dessin de μ, il nous faut recopier la forme tracée par $h(u_i)$, de façon à ce qu'elle soit vérifiée par celle issue de $g(v_i)$. La variante du PCP impose aux v_i de ne jamais être "en avance" sur les u_i, aussi il faudra recopier la partie des u_i non encore vérifiée (partie 7 sur le schéma) avant de la prolonger lors de la prochaine itération de l'expression rationnelle. Enfin, pour que les recopies ne soient ni trop courtes, ni trop longues, et que le retard éventuel pris par les v_i soit rattrapé à la fin, il nous faut ajouter des mots dont la forme se répond et qui servent de contrôles.

En isolant le rôle joué par chacun de ces facteurs (recopie dans un sens, dans l'autre, contrôle au début, à la fin de la recopie, à la fin de la suite de mots),



Chaque chiffre indique un motif qui est associé à une étape logique dans la construction. Un chiffre marqué d'un symbole "prime" signifie que la même étape a déjà été effectuée plus haut, lors d'une itération précédente.

Important : ce schéma vise à faire ressortir l'idée générale de la preuve, et n'illustre aucun résultat précis, contrairement à la figure précédente (Fig. 3.2). En particulier seule les positions relatives des différents motifs sont représentatives, leur forme exacte n'est pas à prendre en compte.

FIG. 3.3 - *Construction pour la variante du PCP.*

et en repérant chacun par une lettre particulière, nous aboutissons à l'expression rationnelle définissant le langage $\mathcal{K}$ suivant :

$$\mathcal{K} = \alpha \left[\biguplus_{i=1}^{n} (h(u_i)\beta(\gamma_0 + \gamma_1)^* \lambda g(v_i) \alpha(\delta_0 + \delta_1)^*) \right]^* \left[\biguplus_{i=1}^{n} (h(u_i)\beta(\gamma_0 + \gamma_1)^* \lambda g(v_i)) \right] \rho \tag{3.1}$$

L'approche de cette formule, d'apparence compliquée, sera sans aucun doute facilitée au lecteur par un lexique, attribuant à chaque facteur (par ordre d'apparition) sa signification méthodologique, le chiffre renvoyant à la Fig. 3.3 :

α : situé en (1), vérifie que la recopie de $(h(u_i)$, située en (4) n'est pas trop longue.

$h(u_i)$: situé en (2), c'est le motif associé au mot u_i, il vérifie aussi que la recopie située en (7) n'est pas trop courte.

β : situé en (3), vérifie que la recopie de $(h(u_i)$, située en (7) n'est pas trop longue, et que les v_i ne prennent pas d'avance sur les u_i.

γ_0 : situé en (4), effectue la recopie des lettres de $h(u_i)$ images par h de 0.

γ_1 : situé en (4), effectue la recopie des lettres de $h(u_i)$ images par h de 1.

λ : situé en (5), vérifie que la recopie de $(h(u_i)$, située en (4) n'est pas trop courte.

$g(v_i)$: situé en (6), motif associé au mot v_i, doit correspondre avec le motif associé à $h(u_i)$.

δ_0 : situé en (7), effectue la recopie des lettres de $h(u_i)$ non encore vérifiées.

δ_1 : situé en (7), effectue la recopie des lettres de $h(u_i)$ non encore vérifiées.

ρ : situé en (8), vérifie que les v_i ont rattrapé leur éventuel retard sur les u_i.

Dans cette expression, la première paire de crochets, mise à l'étoile, génère toutes les suites d'indices possibles, et la deuxième paire de crochets génère et vérifie la fin de la figure. La raison d'être de ce découpage en deux morceaux de l'expression rationnelle est essentiellement technique : il est nécessaire de s'assurer que la suite de mots que l'on tente de mettre en correspondance comprend au moins un mot. La condition supplémentaire apportée par cette variante du PCP permet de limiter la complication de la figure.

C'est cette méthode que nous emploierons dans les sections suivantes pour redémontrer les résultats mentionnés au début de chapitre.

3.3 Quatre résultats d'indécidabilité.

Nous avons exposé dans la section précédente le principe de la construction d'une preuve d'indécidabilité basée sur le problème de Post, pour la famille des langages rationnels. Nous la mettons en œuvre ici sur quatre cas concrets. Une première version de ces résultats est disponible dans [Rob94].

Théorème 3.3.1 *On ne peut pas décider si un langage rationnel sur* Π *contient un mot de figure décrivant un arbre.*

Preuve. Nous allons avoir besoin d'une notation pour désigner, dans un mot, le facteur qui commence à la $i^{\text{ème}}$ lettre et se termine à la $j^{\text{ème}}$ lettre, toutes deux incluses. Soit $s \in \Pi^*$, et $i, j \in \mathbb{N}, 0 \leq i \leq j \leq |s|$, nous noterons $F_i^j(s)$ ce facteur, vérifiant $s \in \Pi^{i-1} F_i^j(s) \Pi^{|s|-j}$.

Pour prouver ce résultat, nous construisons un langage rationnel $\mathcal{K}_1 \subset \Pi^*$, par instanciation des facteurs du langage $\mathcal{K}$ donné dans l'équation 3.1, section 3.2.

Nous définissons les morphismes h et g de $\mathcal{A}^*$ dans Π^* par :

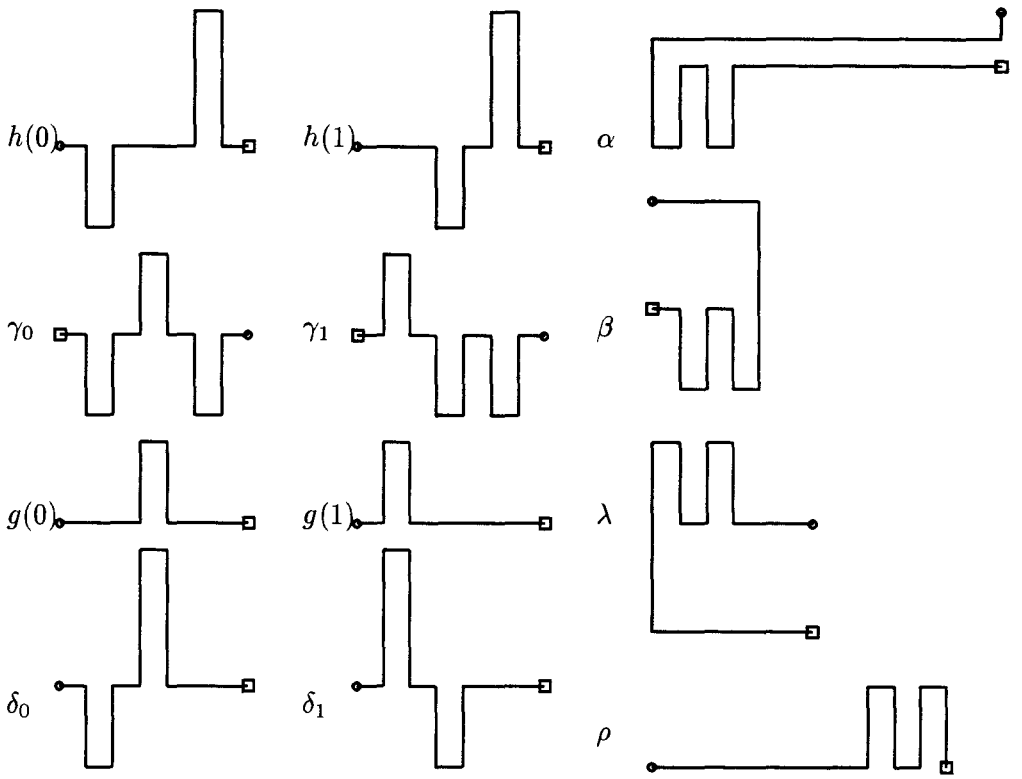
$$\begin{cases} h(0) = rd^3ru^3r^3u^5rd^5r \\ h(1) = r^3d^3ru^3ru^5rd^5r \end{cases} \quad \text{et} \quad \begin{cases} g(0) = r^3u^3rd^3r^3 \\ g(1) = ru^3rd^3r^5 \end{cases}$$

Les facteurs seront donnés par les mots de figures suivants (illustrés en Fig. 3.4) :

$$\begin{aligned} \alpha &= dl^{13}d^4ru^3rd^3ru^3r^{10} \\ \beta &= r^4d^7lu^3ld^3lu^3l \\ \gamma_0 &= ld^3lu^3lu^3ld^3ld^3lu^3l \\ \gamma_1 &= ld^3lu^3ld^3lu^3lu^3ld^3l \\ \lambda &= l^3u^3ld^3lu^3ld^7r^6 \\ \delta_0 &= rd^3ru^3ru^5rd^5r^3 \\ \delta_1 &= ru^5rd^5rd^3ru^3r^3 \\ \rho &= r^8u^3rd^3ru^3rd^3 \end{aligned}$$

Pour chaque suite $i_1, i_2, \ldots, i_k$, on vérifie la correspondance en dessinant l'une en dessous de l'autre l'image de u_i par h et celle de v_i par g, ce qui crée une boucle si et seulement si il n'y a pas correspondance. Les expressions α, β, γ_0, γ_1 et λ servent à aligner le dessin de ces images à la bonne position, en créant une boucle si l'alignement n'est pas bon. Les expressions δ_0, δ_1 servent à recopier le morceau des $h(u_i)$ qui n'a pas encore été mis en correspondance, (du fait de la condition $|u_{i_1}u_{i_2}\ldots u_{i_m}| \geq |v_{i_1}v_{i_2}\ldots v_{i_m}|$, $1 \leq m \leq k$). L'expression ρ vérifie que le retard éventuel pris par les v_i est rattrapé à la fin.

Les choses apparaissent plus clairement dans l'exemple suivant : on pose $u_1 = 0010$, $u_2 = 1$, $u_3 = 1$, $v_1 = 00$, $v_2 = 101$ et $v_3 = 1$; la suite $i_1 = 1$, $i_2 = 2$, $i_3 = 3$ permet dans ce cas de vérifier effectivement l'existence d'une correspondance entre



Note : le point de départ (resp. d'arrivée) du tracé est indiqué par un cercle (resp. un carré).

FIG. 3.4 - *Mots de figures élémentaires dans la construction de K.*

u_i et v_i. Un automate associé à K reconnaît le mot de figure dérivé de cet exemple (voir en section 3.4, Fig. 3.9 et Fig. 3.10) :

$$w = \alpha h(u_1)\beta\gamma_0\gamma_1\gamma_0^2\lambda g(v_1)\alpha\delta_1\delta_0 h(u_2)\beta\gamma_1\gamma_0\gamma_1\lambda g(v_2)\alpha h(u_3)\beta\gamma_1\lambda g(v_3)\rho$$

Proposition 3.3.2 *Si il existe une solution à l'instance (U, V) du PCP alors le langage $\mathcal{K}_1$ contient un mot dont aucun des facteurs n'est une boucle.*

Preuve. Par construction de notre langage. Nous allons donner effectivement le mot en question.

Soit $i_1, i_2, \ldots, i_p$ la suite solution de (U, V), nous avons alors :

$$s = u_{i_1}u_{i_2}\ldots u_{i_p} = v_{i_1}v_{i_2}\ldots v_{i_p}$$

Soient les suites l et m à valeurs dans $\mathbb{N}$ définies par :

$$\begin{cases} l_0 = 0 \\ l_k = |u_{i_1}u_{i_2}\ldots u_{i_k}| \end{cases} \quad \text{et} \quad \begin{cases} m_0 = 0 \\ m_k = |v_{i_1}v_{i_2}\ldots v_{i_k}| \end{cases}$$

Soient les morphismes p' et p'' de $\mathcal{A}^*$ dans Π^*, définis par :

$$\begin{cases} p'(0) = \gamma_0 \\ p'(1) = \gamma_1 \end{cases} \quad \text{et} \quad \begin{cases} p''(0) = \delta_0 \\ p''(1) = \delta_1 \end{cases}$$

Soit le mot :

$$w = \alpha \prod_{k=i_1}^{i_{p-1}} [h(u_k)\beta p'(\omega_{1,k})\lambda g(v_k)\alpha p''(\omega_{2,k})]h(u_{i_p})\beta p'(\omega_{3,k})\lambda g(v_{i_p})\rho$$

où $\omega_{1,k}$ est le miroir de $F^{l_k}_{m_{k-1}+1}(s)$, et $\omega_{2,k} = F^{l_k}_{m_k+1}(s)$ et $\omega_{3,k}$ est le miroir de $F^{l_k}_{m_{k-1}+1}(s)$.

Nous avons $w \in \mathcal{K}_1$ et, par construction, w ne contient pas de boucles. □

Proposition 3.3.3 *Si le langage $\mathcal{K}_1$ contient un mot dont aucun des facteurs n'est une boucle alors il existe une solution à l'instance (U, V) du PCP.*

Preuve. D'après la construction de $\mathcal{K}_1$. Il faut vérifier que les mots w_i suivants contiennent une boucle (c'est à dire qu'il font échouer la construction), et que les mots x_i n'en contiennent pas. Les numéros d'étape sont ceux des figures 3.9 et 3.10.

- Mots devant contenir une boucle :

  Mauvais retour (étape 2) : $w_0 = h(0)\beta\gamma_1$, $w_1 = h(1)\beta\gamma_0$,
  Retour trop court (étape 2) : $w_2 = h(0)\beta\lambda$, $w_3 = h(1)\beta\lambda$,
  Retour trop long (étape 2) : $w_4 = \alpha\beta\gamma_0\lambda$, $w_5 = \alpha\beta\gamma_1\lambda$,
  Mauvaise correspondance entre u_i et v_i (étape 3) : $w_6 = \gamma_0\lambda g(1)$, $w_7 = \gamma_1\lambda g(0)$,
  L'inégalité $|u_{i_1}u_{i_2}\ldots u_{i_m}| \geq |v_{i_1}v_{i_2}\ldots v_{i_m}|$, $1 \leq m \leq k$ n'est pas respectée (étape 3) : $w_8 = \beta\lambda g(0)$, $w_9 = \beta\lambda g(1)$,
  Mauvaise recopie (étape 4) : $w_{10} = \gamma_0\lambda\alpha\delta_1$, $w_{11} = \gamma_1\lambda\alpha\delta_0$,
  Recopie trop longue (étape 4) : $w_{12} = \beta\lambda\alpha\delta_0$, $w_{13} = \beta\lambda\alpha\delta_1$,
  Recopie trop courte à l'étape 4 (vérifié à l'étape 5) : $w_{14} = \gamma_0\lambda\alpha h(0)$, $w_{15} = \gamma_0\lambda\alpha h(1)$, $w_{16} = \gamma_1\lambda\alpha h(0)$, $w_{17} = \gamma_1\lambda\alpha h(1)$,
  Longueur totale des v_i trop courte (étape 6) : $w_{18} = \beta\gamma_0\lambda\rho$, $w_{19} = \beta\gamma_1\lambda\rho$. $w_{20} = \gamma_0\gamma_0\lambda\rho$, $w_{21} = \gamma_0\gamma_1\lambda\rho$. $w_{22} = \gamma_1\gamma_0\lambda\rho$, $w_{23} = \gamma_1\gamma_1\lambda\rho$.

- Mots ne devant pas contenir de boucle :

  Retour correct (étape 2) : $x_0 = h(0)\beta\gamma_0$, $x_1 = h(1)\beta\gamma_1$,
  Longueur du retour correcte (étape 2) : $x_2 = \alpha\beta\lambda$,

Bonne correspondance entre u_i et v_i (étape 3) : $x_3 = \gamma_0\lambda g(0)$, $x_4 = \gamma_1\lambda g(1)$,
L'inégalité $|u_{i_1}u_{i_2}\dots u_{i_m}| \geq |v_{i_1}v_{i_2}\dots v_{i_m}|$, $1 \leq m \leq k$ est respectée (étape 3) : $x_5 = \beta\lambda\alpha h(0)$, $x_6 = \beta\lambda\alpha h(1)$,
Bonne recopie (étape 4) : $x_7 = \gamma_0\lambda\alpha\delta_0$, $x_8 = \gamma_1\lambda\alpha\delta_1$,
Longueur totale des v_i correcte (étape 6) : $x_9 = \beta\lambda\rho$.

La figure 3.5 montre que les mots w_0, w_2, w_4, w_6, w_8 contiennent une boucle. La vérification des autres cas est laissée au lecteur. □

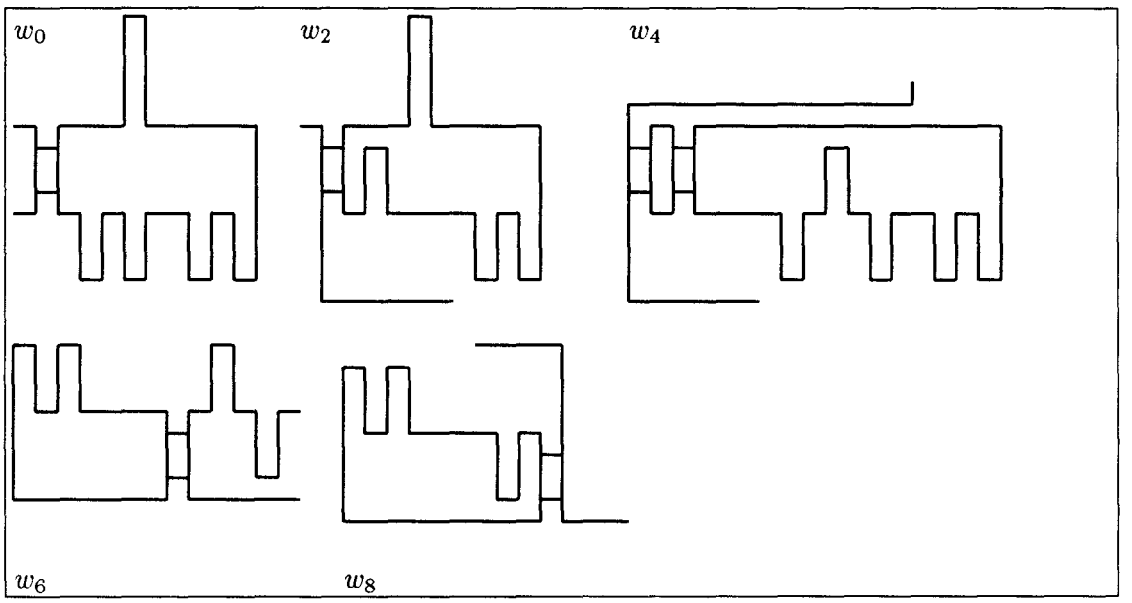


FIG. 3.5 - *Mots de figures contenant une boucle.*

Nous constatons enfin que dans la construction de $\mathcal{K}_1$ nous avons veillé à mettre en place la proposition suivante :

Proposition 3.3.4 *Soit $\omega \in \mathcal{K}_1$. Le graphe associé G_ω contient un cycle si et seulement si ω contient un facteur qui est une boucle.*

Preuve. Par construction de $\mathcal{K}_1$.
Le théorème 3.3.1 se déduit des propositions 3.3.2, 3.3.3 et 3.3.4. □

Corollaire 3.3.5 *On ne peut pas décider si un langage rationnel sur Π contient un mot de figure décrivant un contour de polyomino.*

Preuve. Remarquons tout d'abord que si un arbre est dessiné par $\mathcal{K}_1$, alors il est réduit à un fil, sans embranchements : quelque soit le nœud du graphe G_ω, son arité est inférieure ou égale à 2. Nous définissons $\mathcal{K}_2 = rd^*l^*u^*r^*$, que nous allons

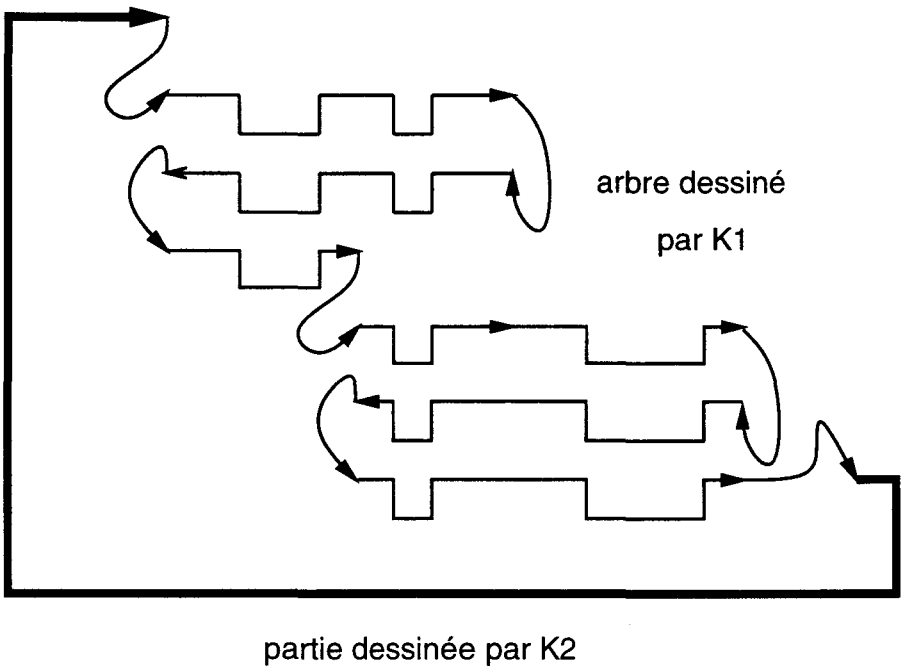


FIG. 3.6 - *Obtention d'un contour à partir d'un arbre.*

concaténer à $\mathcal{K}_1$ pour obtenir, à partir d'un arbre, un mot de contour. Nous déduisons immédiatement des propositions 3.3.2, 3.3.3 et 3.3.4 que le langage $\mathcal{K}_1.\mathcal{K}_2$ ne contient un mot de contour que si et seulement si l'instance (U, V) du PCP possède une solution (voir Fig.3.6). La propriété 3.3.4, en particulier, nous assure que nous restons dans le cadre de notre définition d'un mot de contour : il n'y a pas de boucle "plate" (contre-exemples de la Fig. 3.1). □

Corollaire 3.3.6 *On ne peut pas décider si un langage rationnel sur* Π *contient un mot de figure minimal.*

Preuve. Si $\mathcal{K}_1$ contient un mot codant un arbre, alors ce mot est clairement minimal (il a autant de lettres que la figure a de segments). Si $\mathcal{K}_1$ ne contient pas un tel mot, alors, par construction, tous les mots de $\mathcal{K}_1$ contiennent au moins une boucle de la forme $w = d^2ru^2w'u^2ld^2$ (avec $w' \in \Pi^*$), ou l'équivalent, laquelle boucle peut être optimisée en $d^2ru^2lrw'r$ (voir Fig. 3.7). Donc $\mathcal{K}_1$ ne contient un mot minimal que si et seulement si il contient un arbre. □

Théorème 3.3.7 *On ne peut pas décider si un langage rationnel sur* Π_i *contient un mot de figure décrivant une figure connexe.*

Preuve. Pour prouver ce résultat, nous construisons un langage rationnel $\mathcal{K}_3 \subset \Pi_i^*$, par instanciation de notre modèle $\mathcal{K}$, tel que Fig($\mathcal{K}_3$) contient une figure connexe si et seulement si l'instance (U, V) du PCP présentée plus haut possède une solution.

Note : les traits superposés sont représentés légèrement décalés.

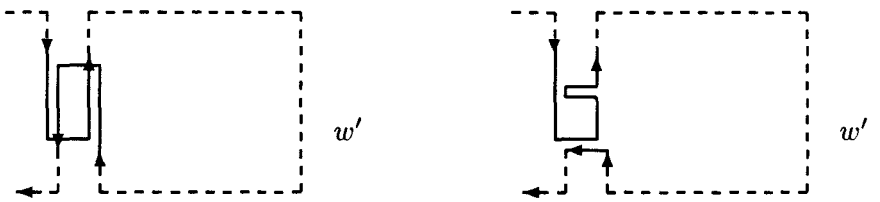


FIG. 3.7 - *Boucle optimisable dans les mots de* $\mathcal{K}_1$.

Les facteurs sont illustrés en Fig. 3.8 et sont définis comme suit :

$$\begin{cases} h(0) = ru'udd'ru'udd'rr \\ h(1) = ru'udd'rru'udd'r \end{cases} \quad \text{et} \quad \begin{cases} g(0) = r'r'udr'r' \\ g(1) = r'r'r'udr' \end{cases}$$

$$\begin{aligned} \alpha &= ddd \\ \beta &= 1 \\ \gamma_0 &= l'l'l'udl' \\ \gamma_1 &= \gamma_0 \\ \lambda &= 1 \\ \delta_0 &= ru'udd'ru'uu'udd'dd'rr \\ \delta_1 &= ru'udd'rru'uu'udd'dd'r \\ \rho &= 1 \end{aligned}$$

Note : le point de départ (resp. d'arrivée) du tracé est indiqué par un cercle (resp. un carré), les mouvements crayon levé par des pointillés.

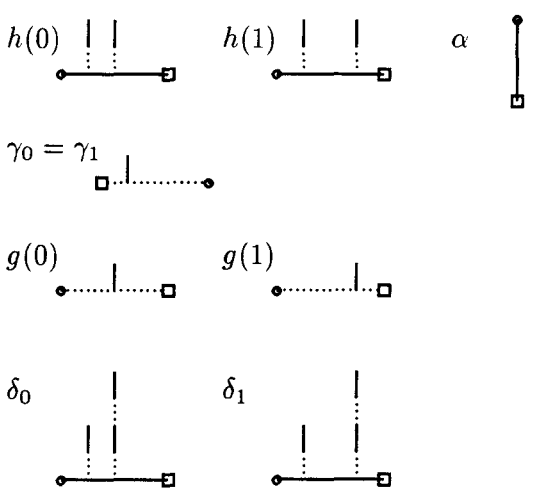


FIG. 3.8 - *Mots de figures élémentaires dans la construction de* $\mathcal{K}_3$.

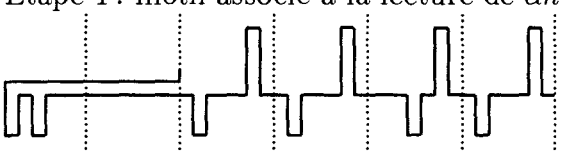
L'idée de la construction est tout à fait identique à celle décrite plus haut. Les étapes de la mise en correspondance apparaissent dans la section 3.4, en Fig. 3.11, pour le même exemple : $u_1 = 0010$, $u_2 = 1$, $u_3 = 1$, $v_1 = 00$, $v_2 = 101$ et $v_3 = 1$ avec la suite $i_1 = 1$, $i_2 = 2$, $i_3 = 3$.

Le théorème 3.3.7 se déduit d'après la construction de $\mathcal{K}_3$, similairement au théorème 3.3.1. □

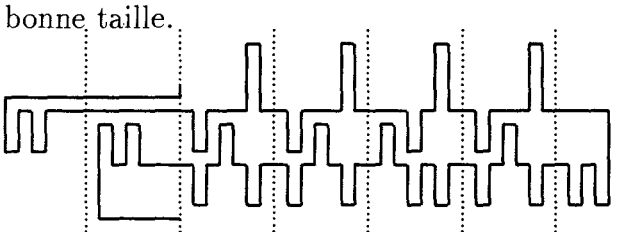
3.4 Exemples.

Nous avons réuni dans les pages suivantes les deux exemples complets de mise en correspondance dont il est fait mention plus haut : pour la question de l'existence d'un arbre et pour celle d'une figure connexe.

Note : des repères pointillés indiquent l'alignement vertical des lettres mises en correspondance.
Étape 1 : motif associé à la lecture de $\alpha h(u_1)$.



Étape 2 : lecture de $\beta\gamma_0\gamma_1\gamma_0^2\lambda$: pour vérifier la bonne correspondance des mots, on place le crayon de façon à pouvoir par la suite dessiner le motif de v_1 sous celui de u_1. Il faut procéder lors de ce replacement en à une recopie de la forme associée à u_1 : c'est le rôle de γ_0 et γ_1, qui ne créent pas de boucle si et seulement si la forme recopiée est la bonne. Il faut vérifier que cette recopie ne s'arrête ni trop tôt ni trop tard : α et λ créent une boucle si et seulement si elle n'a pas la bonne taille.



Étape 3 : lecture de $g(v_1)$: il se crée une boucle s'il n'y a pas correspondance, ou si l'inégalité $|u_{i_1}u_{i_2}\dots u_{i_m}| \geq |v_{i_1}v_{i_2}\dots v_{i_m}|$, $1 \leq m \leq k$ n'est pas respectée : le mot est alors trop long et une boucle se forme avec le facteur β dessiné à l'étape 2.

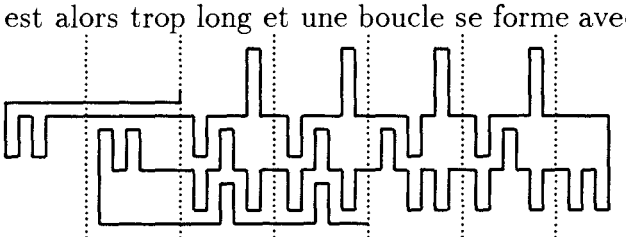
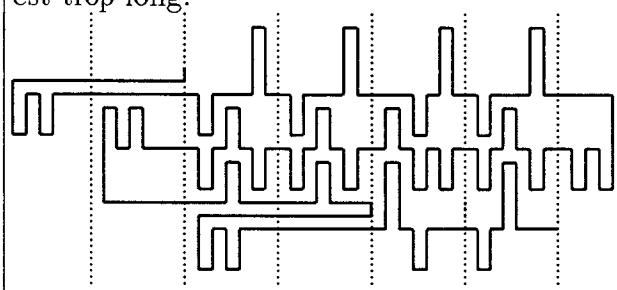
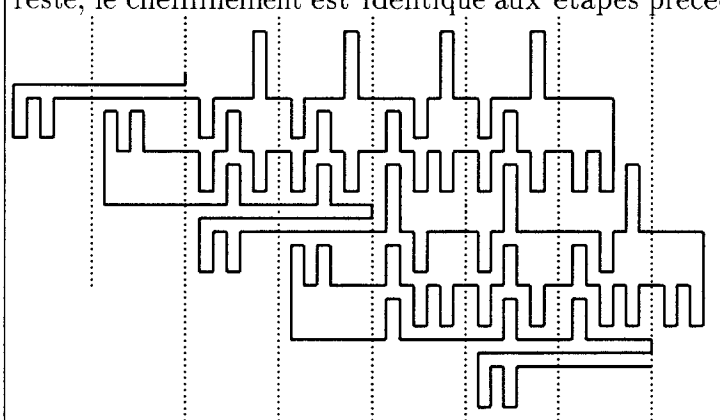


FIG. 3.9 - *Exemple d'une mise en correspondance réussie (partie I).*

Étape 4 : lecture de $\alpha\delta_1\delta_0$: le facteur α prépare le prochain retour (cf. étape 2). On recopie grâce à δ_0, δ_1 la partie de $h(u_1)$ qui n'a pas encore été mise en correspondance, afin de pouvoir la chaîner avec $h(u_2)$ lors de l'étape suivante. Le facteur β dessiné à l'étape 1 crée une boucle si et seulement si le morceau recopié est trop long.



Étape 5 : lecture de $h(u_2)\beta\gamma_1\gamma_0\gamma_1\lambda g(v_2)\alpha$: le contour de la première lettre de $h(u_2)$ vérifie que le morceau recopié à l'étape précédente n'est pas trop court. Pour le reste, le cheminement est identique aux étapes précédentes.



Étape 6 : lecture de $h(u_3)\beta\gamma_1\lambda g(v_3)\rho$: lorsque le dernier v_i est lu, il est immédiatement suivi du facteur ρ, lequel teste que l'égalité $|u_{i_1}u_{i_2}\dots u_{i_k}| = |v_{i_1}v_{i_2}\dots v_{i_k}|$ est désormais vérifiée.

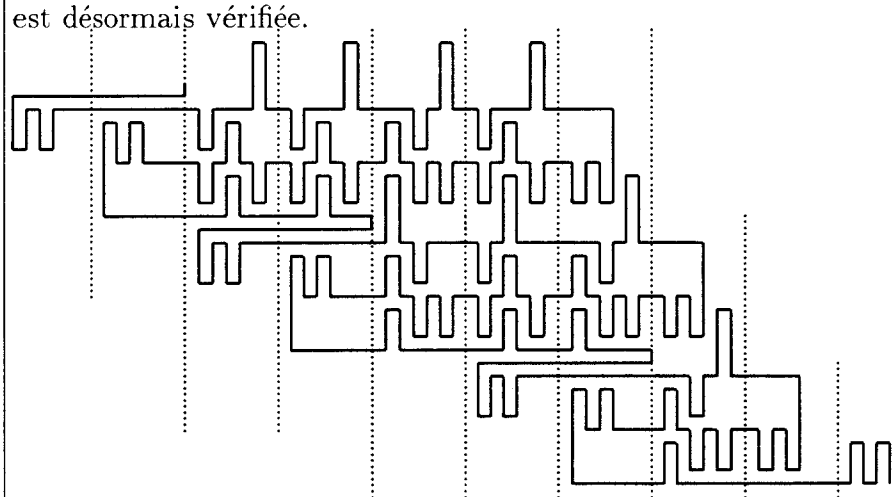
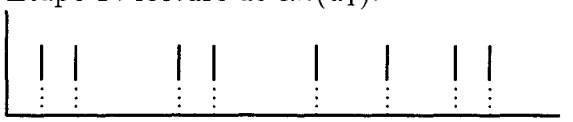


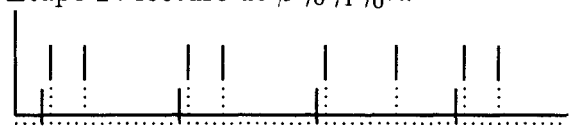
FIG. 3.10 - *Exemple d'une mise en correspondance réussie (partie II).*

Note : les segments ajoutés à chaque nouvelle étape sont représentés légèrement décalés. Les pointillés indiquent les déplacements invisibles de l'étape en cours.

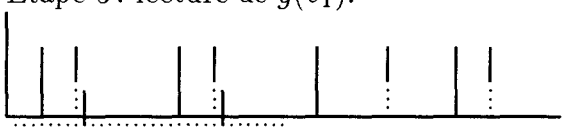
Étape 1 : lecture de $\alpha h(u_1)$.



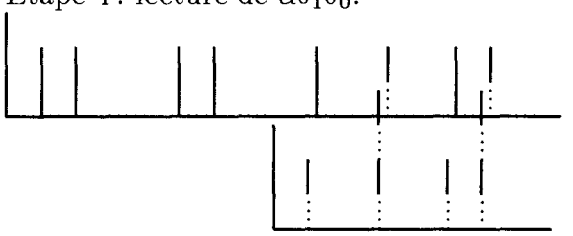
Étape 2 : lecture de $\beta\gamma_0\gamma_1\gamma_0^2\lambda$.



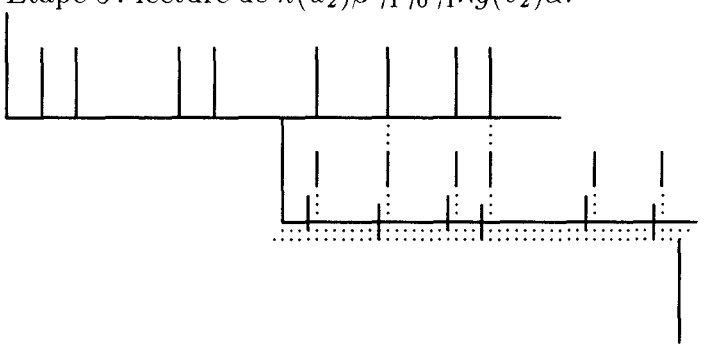
Étape 3 : lecture de $g(v_1)$.



Étape 4 : lecture de $\alpha\delta_1\delta_0$.



Étape 5 : lecture de $h(u_2)\beta\gamma_1\gamma_0\gamma_1\lambda g(v_2)\alpha$.



Étape 6 : lecture de $h(u_3)\beta\gamma_1\lambda g(v_3)\rho$.

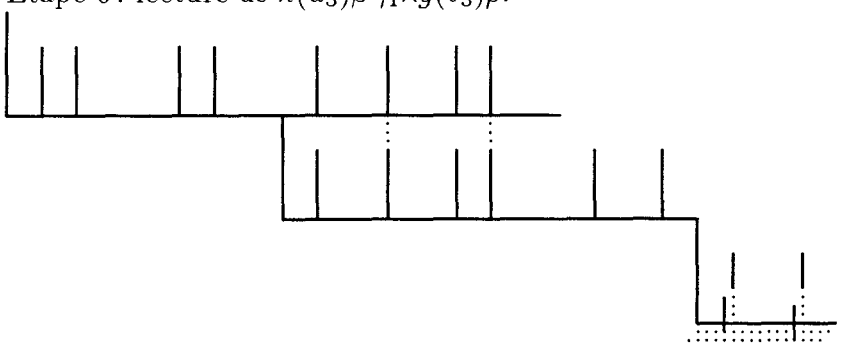


FIG. 3.11 - *Mise en correspondance pour une figure connexe.*

Troisième partie

Langages de mots avec branchements

Chapitre 1

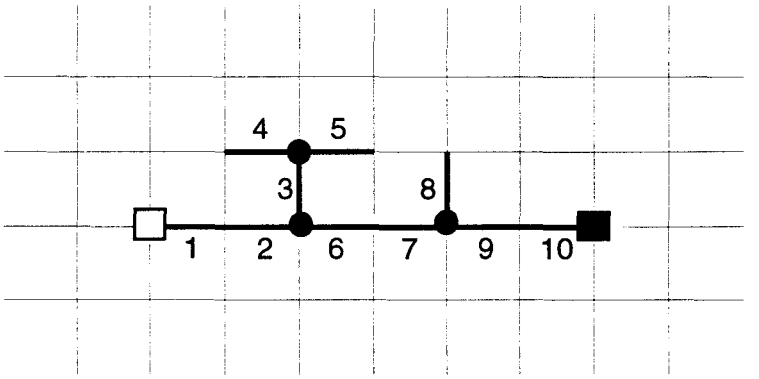
Langages avec branchements

Dans la partie précédente, nous avions notamment cherché à préciser, au niveau de la complexité descriptive, certaines différences existant entre les langages classiques et ceux avec lever de crayon. Nous allons, dans cette partie, nous livrer à une autre étude comparative, qui concernera le pouvoir d'expression des langages de mots de figures rationnels définis selon le modèle classique, d'une part, et selon celui proposé par R. Gutbrod, d'autre part. Cette comparaison sera fondée sur la capacité à représenter, dans ces deux systèmes, les résultats d'un opérateur de collage (ou superposition) travaillant sur les figures de base. Ces travaux ont fait l'objet d'une publication dans [RR94]. Avant de présenter, au chapitre prochain, cet opérateur et les résultats obtenus à son propos, nous allons définir formellement la sémantique proposée par Gutbrod, en lui adjoignant la notion d'inverse qui nous sera nécessaire.

1.1 Définitions fondamentales

Nous présentons ici le concept des langages de *mots de figures avec branchements*, que nous abrégerons par la suite en *langages avec branchements*, ou encore LAB. Nous nous inspirons de [Gut90, Gut91] où l'on trouve une étude de ce modèle. Nous procédons comme nous l'avons fait pour la sémantique avec lever de crayon, c'est à dire que nous ne redéfinissons que les points qui diffèrent d'avec la sémantique de Maurer *et al.*, et que nous nous autorisons à employer le même vocabulaire, le lecteur se reportant aux définitions adéquates selon les cas.

Les mots de figure dans les LAB sont définis sur l'alphabet $\Pi_b = \{u, r, d, l, \#, \&\}$. Dans la métaphore de la table traçante, on suppose que l'on dispose d'une pile permettant de stocker des coordonnées. Les lettres habituelles conservent le sens que nous leur avions défini dans les parties précédentes. Le symbole # signifie alors empiler les coordonnées de la position courante du crayon, et c'est ce que nous appellerons mettre un *point de branchement*. Le symbole & signifie, quant à lui, dépiler les coordonnées stockées au sommet de la pile, et placer le crayon



P-figure associée au mot $rr\#u\#l\&r\&rr\#u\&rr$. Les nombres indiquent l'ordre de tracé des segments.

FIG. 1.1 - *Mot de figure avec branchements.*

à l'endroit indiqué. Il faut noter que ce déplacement se fait sans laisser de traces (c'est à dire crayon levé), donc sans ajouter de segments à la figure. Un exemple est donné en Fig. 1.1. Notons que cette nouvelle possibilité ne permet que le tracé de figures connexes, à l'instar des mots définis sur l'alphabet Π.

Il est possible de supprimer des ordres d'empilement et de dépilement en simulant le déplacement opéré lors du dépilement par un mot sur Π, qui doit être choisi pour ne pas ajouter de nouveaux segments à la figure. Nous utilisons pour cela l'inverse des mots de figures classiques. Plus précisément :

Définition 1.1.1 *Soit $\omega \in \Pi_b^*$, l'ensemble des mots avec branchements réduits de ω, noté Bred(ω), est défini par :*

- $\omega \in Bred(\omega)$,
- *Si $\omega' = \omega_1\#\omega_2\&\omega_3 \in Bred(\omega)$, avec $\omega_2 \in \Pi^*$, alors $\omega_1\omega_2\overline{\omega_2}\omega_3 \in Bred(\omega)$.*

Quelque soit $\omega \in \Pi_b^*$, il est clair qu'il existe au plus un mot dans $\mathrm{Bred}(\omega) \cap \Pi^*$, c'est à dire qui soit sans branchements.

À l'évidence, certains mots n'ont pas de sens si on veut les interpréter dans la métaphore de la table traçante : c'est le cas par exemple du mot $ur\&$ qui comporte un ordre de dépilement lequel ne correspond à aucun empilement. Nous nous intéressons donc plus spécialement aux mots correctement parenthésés sur la paire de symboles $\{\#,\&\}$, que nous appellerons mots descriptifs. Cette notion peut justement s'exprimer grâce à l'ensemble Bred :

Définition 1.1.2 *Un mot est* descriptif *si et seulement si $Bred(\omega) \cap \Pi^* \neq \emptyset$. Un* LAB *est descriptif si et seulement si tous ses mots sont descriptifs. Un langage*

de figures (respectivement p-figures) est descriptif si et seulement si il existe un LAB *descriptif qui le représente.*

Nous pouvons alors définir la notion de figure associée à un mot avec branchements, cette définition ne portant que sur les mots descriptifs.

Définition 1.1.3 *Soit $\omega \in \Pi_b^*$, si $Bred(\omega) \cap \Pi^* \neq \emptyset$ alors on note $Br(\omega)$ l'unique mot dans $Bred(\omega) \cap \Pi^*$, et on définit le dessin associé à ω par $Des(\omega) = Des(Br(\omega))$, et, de la même façon, $Desp(\omega) = Desp(Br(\omega))$, $Fig(\omega) = Fig(Br(\omega))$ et $p\text{-}Fig(\omega) = p\text{-}Fig(Br(\omega))$.*

Nous ferons quant à nous deux remarques sur Br.

Proposition 1.1.4 *Soit μ un mot avec branchements descriptif, alors :*

$$Br(\#\mu\&) = Br(\mu)\overline{Br(\mu)}$$

Preuve. Par récurrence sur le degré p de μ. Si $p = 0$, la propriété est vraie par définition. On la suppose vraie pour tout $p < n$. Si $p = n$, alors μ est de la forme $\mu = \mu_1\#\mu_2\&\mu_3 \ldots \#\mu_{2k}\&\mu_{2k+1}$, avec $\mu_{2i} \in \Pi^*$ pour $1 \leq i \leq k$, et :

$$\begin{aligned}
\mathrm{Br}(\#\mu\&) &= \mathrm{Br}(\mu_1\#\mu_2\&\mu_3 \ldots \#\mu_{2k}\&\mu_{2k+1}) \\
&\overset{\text{def}}{=} \mathrm{Br}(\mu_1\mu_2\overline{\mu_2}\mu_3 \ldots \mu_{2k}\overline{\mu_{2k}}\mu_{2k+1}) \\
&\overset{\text{rec}}{=} \mathrm{Br}(\mu_1\mu_2\overline{\mu_2}\mu_3 \ldots \mu_{2k}\overline{\mu_{2k}}\mu_{2k+1})\,\overline{\mathrm{Br}(\mu_1\mu_2\overline{\mu_2}\mu_3 \ldots \mu_{2k}\overline{\mu_{2k}}\mu_{2k+1})} \\
&\overset{\text{def}}{=} \mathrm{Br}(\mu_1\#\mu_2\&\mu_3 \ldots \#\mu_{2k}\&\mu_{2k+1})\,\overline{\mathrm{Br}(\mu_1\#\mu_2\&\mu_3 \ldots \#\mu_{2k}\&\mu_{2k+1})} \\
&= \mathrm{Br}(\mu)\overline{\mathrm{Br}(\mu)}
\end{aligned}$$

□

Proposition 1.1.5 *Soient μ et ν des mots avec branchements descriptifs, alors :*

$$Br(\mu\nu) = Br(\mu)Br(\nu)$$

Preuve. La preuve se fait par récurrence sur le degré, de manière similaire à la preuve précédente. □

La profondeur de la pile nécessaire pour interpréter un mot de figure avec branchements est une information importante, notamment dans le cas d'une implantation sur machine. Nous appellerons cette notion le *degré*, et nous allons la mettre en évidence en définissant la famille de langage qui suit.

Définition 1.1.6 *Soit les langages avec branchements :*

- $\Sigma_0 = \Pi^*$,

- $\Sigma_{n+1} = (\Sigma_n \# \Sigma_n \&)^* \Sigma_n, n \in I\!N$
- $\Sigma^* = \bigcup_{i \geq 0} \Sigma_i$.

On remarque que Σ^* est l'ensemble de tous les mots descriptifs, et que Σ_n est l'ensemble des mots descriptifs de degré inférieur ou égal à n. Ce sont des monoïdes.

Fait 1.1.7 *Un* LAB $\mathcal{K}$ *est descriptif si et seulement si* $\mathcal{K} \subset \Sigma^*$.

Définition 1.1.8 *Le degré d'un mot descriptif* μ *est le plus petit* n *tel que* $\mu \in \Sigma_n$. *Le degré d'un* LAB *descriptif* $\mathcal{K}$ *est le plus petit* n *tel que* $\mathcal{K} \subset \Sigma_n$. *Le degré d'un langage de figures descriptif* $\mathcal{L}$ *est le plus petit des degrés des* LAB $\mathcal{K}$ *tels que* $\mathcal{L} = Fig(\mathcal{K})$.

Bien entendu cette définition s'étend de manière naturelle: par exemple le degré d'un langage de p-figures rationnel descriptif est le plus petit des degrés des LAB rationnels descriptifs qui le représentent.

Fait 1.1.9 *Soit un* LAB *rationnel descriptif* $\mathcal{K}$, *alors il existe* $n \geq 0$ *tel que* $\mathcal{K} \subset \Sigma_n$.

Cette dernière remarque est importante : elle signifie que le degré des mots d'un LAB rationnel descriptif est toujours borné. Par la suite nous limiterons notre étude aux langages avec branchements rationnels descriptifs. Faisons remarquer tout de suite que le pouvoir d'expression de ces langages est supérieur à celui des langages rationnels classiques : par exemple p-Fig$(\#r^*\&)$ = p-Fig$(\{r^n l^n \mid n \in I\!N\}$.

1.2 Inverse d'un LAB rationnel descriptif

Dans les LAB descriptifs, la définition d'une notion de mot inverse, analogue à celle de l'inverse d'un mot classique, nous serait utile : soit un mot et sa p-figure f associée, nous souhaiterions disposer de l'inverse de ce mot, décrivant l'inverse de f, c'est à dire qui la trace en partant du point d'arrivée pour se terminer au point de départ de f. Dans le cas où le mot ne contient pas de branchements, c'est bien sûr la définition classique de l'inverse, vue dans la première partie, qui est conservée. Dans les autres cas, lorsque le mot appartient à $\Sigma^* \setminus \Pi^*$, on ne peut pas se contenter de parcourir le mot de figure à l'envers, à cause des sauts effectués lors des dépilements. De plus nous désirons que cette notion ait de bonnes propriétés : l'ensemble des inverses d'un langage rationnel descriptif doit être aussi un langage rationnel descriptif, constructible.

Nous choisissons de poser que l'inverse d'un mot contenant au moins un branchement s'obtient, intuitivement, en parcourant en sens inverse les segments de

la figure tracés par les lettres situées à la profondeur 0 vis à vis des branchements, et en laissant les facteurs plus profonds dans leur sens originel. Nous en donnons une première définition formelle, et nous montrons par la suite, avec une deuxième définition, que nous pouvons la mettre en œuvre en respectant les contraintes données au paragraphe précédent :

Définition 1.2.1 *Soit un mot avec branchements descriptif* $\mu \in (\Sigma^*\backslash\Pi^*)$, *on peut toujours trouver une factorisation de la forme :* $\mu = \mu_0\#\mu_1\&\mu_2 \ldots \#\mu_{2n-1}\&\mu_{2n}$, *avec* $\mu_{2i} \in \Pi^*$ *pour* $0 \leq i \leq n$, *et* $\mu_{2i+1} \in \Sigma^*$ *pour* $0 \leq i \leq n-1$.

On note alors $\overline{\mu}$ *l'inverse de* μ, *défini comme suit :*

$$\overline{\mu} = \overline{\mu_{2n}}\#\mu_{2n-1}\& \ldots \overline{\mu_2}\#\mu_1\&\overline{\mu_0}$$

Par exemple, si $\mu = r\#u\&r\#ur\#u\&r\&rr$, alors $\overline{\mu} = ll\#ur\#u\&r\&l\#u\&l$ comme on peut le voir sur le schéma 1.2. On remarquera que, par définition, l'inverse d'un mot descriptif est descriptif, et que quelque soit le mot μ, alors $\overline{\overline{\mu}} = \mu$.

Proposition 1.2.2 *Soit* μ *un mot de figure avec branchements, alors :*

$$Br(\overline{\mu}) = \overline{Br(\mu)}$$

Preuve. Par récurrence sur le degré de μ. Si $\mu \in \Pi^*$, alors évidemment $\mathrm{Br}(\mu) = \mu$. Nous supposons la proposition vraie pour tout degré k avec $0 \leq k \leq p$. Soit μ de degré $p+1$, $\mu = \mu_0\#\mu_1\&\mu_2 \ldots \#\mu_{2n-1}\&\mu_{2n}$, avec $\mu_{2i+1} \in \Sigma^p$ pour $0 \leq i \leq n-1$. Nous avons :

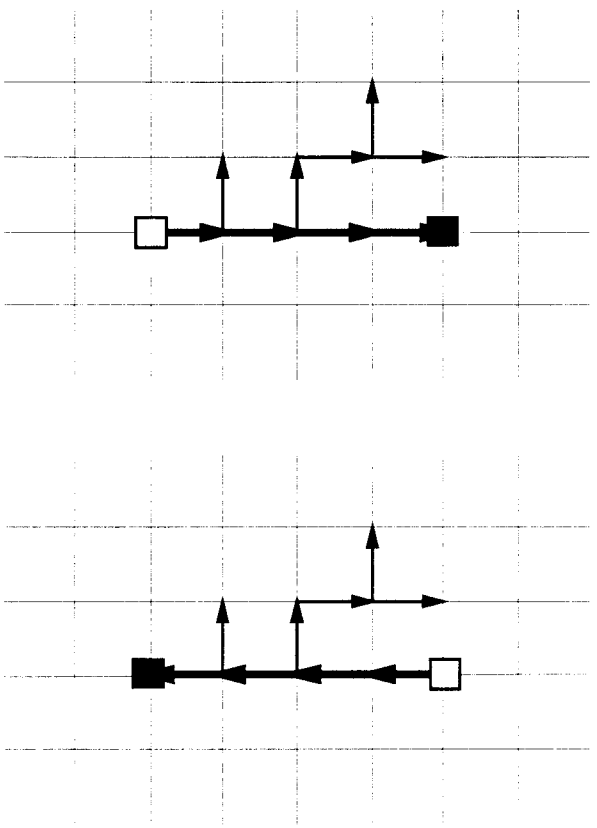
$$\begin{aligned}
\mathrm{Br}(\overline{\mu}) &= \mathrm{Br}(\overline{\mu_{2n}})\mathrm{Br}(\#\mu_{2n-1}\&)\ldots\mathrm{Br}(\overline{\mu_2})\mathrm{Br}(\#\mu_1\&)\mathrm{Br}(\overline{\mu_0}) \\
&= \mathrm{Br}(\overline{\mu_{2n}})\mathrm{Br}(\overline{\#\mu_{2n-1}\&})\ldots\mathrm{Br}(\overline{\mu_2})\mathrm{Br}(\overline{\#\mu_1\&})\mathrm{Br}(\overline{\mu_0}) \\
&\overset{\text{rec}}{=} \overline{\mathrm{Br}(\mu_{2n})}\ \overline{\mathrm{Br}(\#\mu_{2n-1}\&)}\ldots\overline{\mathrm{Br}(\mu_2)}\ \overline{\mathrm{Br}(\#\mu_1\&)}\ \overline{\mathrm{Br}(\mu_0)} \\
&= \overline{\mathrm{Br}(\mu_0)\mathrm{Br}(\#\mu_1\&)\mathrm{Br}(\mu_2)\ldots\mathrm{Br}(\#\mu_{2n-1}\&)\mathrm{Br}(\mu_{2n})} \\
&= \overline{\mathrm{Br}(\mu_0\#\mu_1\&\mu_2\ldots\#\mu_{2n-1}\&\mu_{2n})} \\
&= \overline{\mathrm{Br}(\mu)}
\end{aligned}$$

□

Corollaire 1.2.3 *L'inverse d'un mot descriptif* μ *de* Π_b^* *décrit l'inverse de la figure associée à* μ *: soit* $\mu \in \Sigma^*$, *alors* $\overline{p\text{-}Fig(\mu)} = p\text{-}Fig(\overline{\mu})$

Preuve. Directement d'après la proposition précédente.□

Cette première définition précise sans ambiguïté ce que nous entendons par mot inverse, mais elle ne peut nous servir telle quelle, faute de pouvoir montrer



La p-figure associée au mot de départ est en haut, celle associée à son inverse est dessous. Les flêches montrent le sens d'exécution du tracé. Les segments en trait épais correspondent aux lettres situées en profondeur 0.

FIG. 1.2 - *Inverse d'un mot de figure avec branchements.*

clairement qu'elle conserve la rationalité. Nous allons redéfinir cette notion pour les langages rationnels descriptifs. Remarquons d'abord qu'un automate reconnaissant un tel langage, peut, en quelque sorte, être partitionné en deux, une partie (états de Q_1) effectuant la reconnaissance des facteurs situés en profondeur 0, l'autre partie (états de Q_2) reconnaissant les facteurs dessinant des branchements.

Proposition 1.2.4 *Soit $\mathcal{L}$ un* LAB *rationnel descriptif reconnu par un automate fini dont tous les états sont accessibles et co-accessibles $A =< \Pi_b, Q, D, F, \delta >$, où D est l'ensemble des états de départ, F l'ensemble des états finaux, et δ la fonction de transition.*

Il existe une partition de $Q = Q_1 \cup Q_2$ où Q_1, Q_2 sont définis comme suit, avec q_d désignant un état de D :

- *soit un état $q \in Q$, alors $q \in Q_1 \Leftrightarrow (\forall \mu, \delta(q_d, \mu) \ni q \Rightarrow |\mu|_\# = |\mu|_\&)$*
- *soit un état $q \in Q$, alors $q \in Q_2 \Leftrightarrow (\forall \mu, \delta(q_d, \mu) \ni q \Rightarrow |\mu|_\# > |\mu|_\&)$*

Preuve. Montrons d'abord qu'il n'existe pas d'état dans l'automate auquel on puisse accéder par un mot comportant plus de & que de #. Par l'absurde, on suppose qu'un tel état existe : $\exists q \in Q, \exists \mu, |\mu|_\# < |\mu|_\&, \delta(q_d, \mu) \ni q$. Comme A est co-accessible, alors il existe μ' tel que $\mu\mu'$ soit reconnu mais n'est pas descriptif. Contradiction. □

Montrons ensuite que si on accède à un état par un mot comportant plus de # que de &, alors tous les mots permettant d'accéder à cet état vérifient la même propriété, et de même s'il y a autant de # que de &. Par l'absurde, supposons qu'il existe un état : $\exists q \in Q$, tel que $\exists \mu_1, |\mu_1|_\# > |\mu_1|_\&, \delta(q_d, \mu_1) \ni q$, et $\exists \mu_2, |\mu_2|_\# = |\mu_2|_\&, \delta(q_d, \mu_2) \ni q$. Alors il existe μ' tel que $\mu_1\mu'$ soit dans $\mathcal{L}$ descriptif, et donc $|\mu'|_\# < |\mu'|_\&$. Par conséquent $\mu_2\mu'$ est reconnu par A, et n'est pas descriptif. Contradiction.

Proposition 1.2.5 *Par définition de Q_1 et Q_2, nous avons les propriétés suivantes :*

- *$\forall q, q' \in Q_1$, tels que $\exists x \in \Pi_b, \delta(q, x) \ni q'$, alors $x \in \Pi$.*
- *$\forall q \in Q_1, q' \in Q_2$, tels que $\exists x \in \Pi_b, \delta(q, x) \ni q'$, alors $x = \#$.*
- *$\forall q \in Q_2, q' \in Q_1$, tels que $\exists x \in \Pi_b, \delta(q, x) \ni q'$, alors $x = \&$.*

Nous pouvons définir le langage inverse, en construisant un automate dérivé, où on transforme la partie Q_1 en l'inverse habituel (partie 1 de la définition de δ'), on laisse la partie Q_2 telle quelle (partie 2), et on "croise" les transitions allant de Q_1 dans Q_2 (parties 3 et 4).

Définition 1.2.6 *Soit $\mathcal{L}$ un* LAB *rationnel descriptif reconnu par un automate fini dont tous les états sont accessibles et co-accessibles $A =< \Pi_b, Q_1 \cup Q_2, D, F, \delta >$, le langage inverse, noté $\overline{\mathcal{L}}$, est reconnu par l'automate $\overline{A} =< \Pi_b, Q_1 \cup Q_2, F, D, \delta' >$, avec δ' définie comme suit :*

1. *Soient $q, q' \in Q_1$ et $a \in \Pi$ tels que $\delta(q,a) \ni q'$ alors $\delta'(q', \overline{a}) \ni q$.*
2. *Soient $q, q' \in Q_2$ et $a \in \Pi_b$ tels que $\delta(q,a) \ni q'$ alors $\delta'(q,a) \ni q'$.*
3. *Soient $q \in Q_1, q' \in Q_2$ tels que $\exists q'' \in Q_2, q''' \in Q_1$ avec $\delta(q'', \&) \ni q, \delta(q''', \#) \ni q'$ et $\exists \mu, \delta(q', \mu) \ni q''$ alors $\delta'(q, \#) \ni q'$.*
4. *Soient $q \in Q_1, q' \in Q_2$ tels que $\exists q'' \in Q_2, q''' \in Q_1$ avec $\delta(q', \&) \ni q''', \delta(q, \#) \ni q''$ et $\exists \mu, \delta(q'', \mu) \ni q'$ alors $\delta'(q', \&) \ni q$.*
5. *Soient $q \in Q$ et $a \in \Pi_b$, alors $\delta'(q,a)$ ne contient pas d'autres états que ceux définis par les points 1 à 4.*

Un exemple d'inversion d'automate est donné en Fig. 1.3. On remarque que la proposition 1.2.5 est vérifiée par définition pour $\overline{A}$.

Proposition 1.2.7 *L'automate $\overline{A}$ reconnaît un langage descriptif, de même degré que celui reconnu par A, et tous ses états sont accessibles et co-accessibles.*

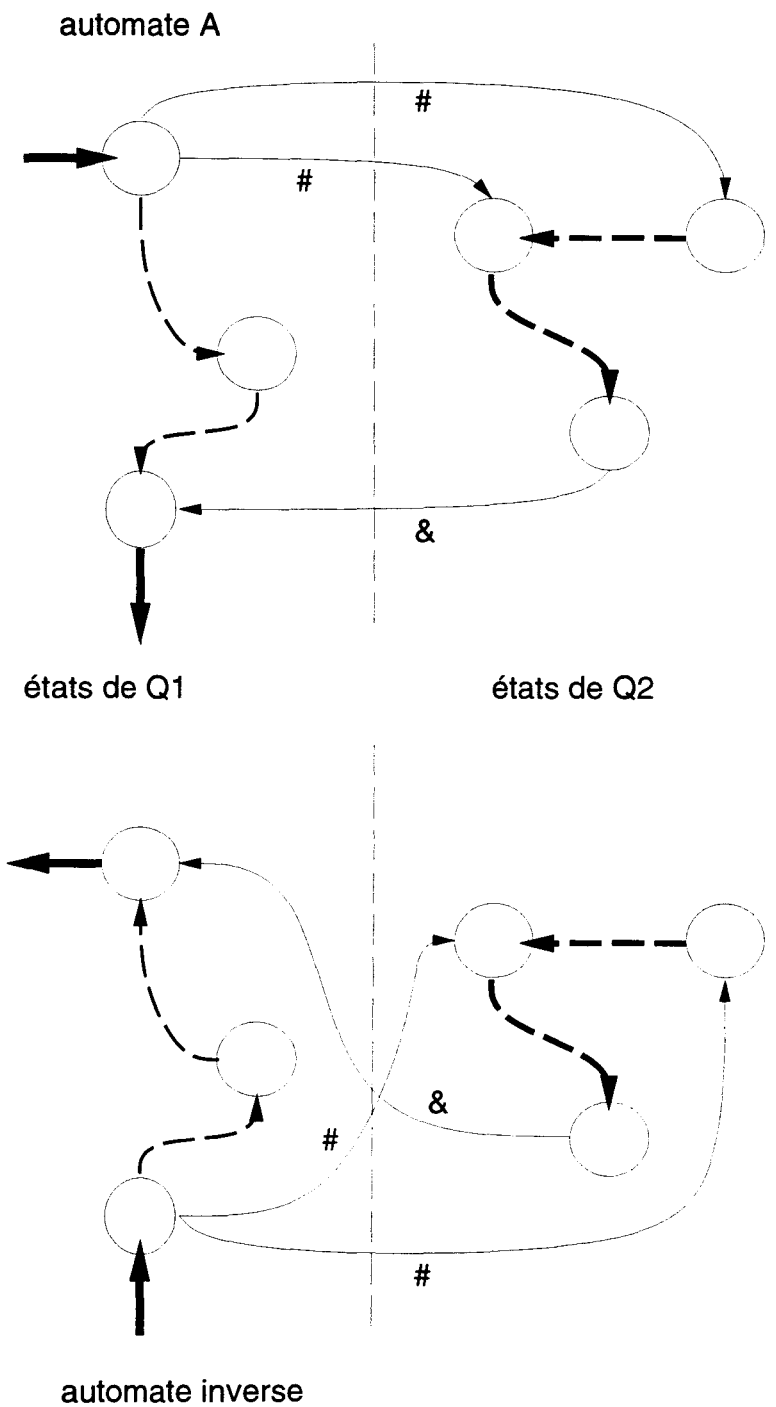
Preuve. Par l'absurde.

Supposons que le langage ne soit pas descriptif. Il existe alors $q, q''' \in Q_1$ et $q', q'' \in Q_2$ et μ non descriptif tels que $\delta'(q, \#) \ni q'$, $\delta'(q', \mu) \ni q''$, $\delta'(q'', \&) \ni q'''$. D'après les parties 3 et 4 de la définition 1.2.6, alors il existe dans l'automate A deux états $r, r' \in Q_1$ tels que transitions $\delta(r, \#) \ni q'$, $\delta(q', \mu) \ni q''$, $\delta(q'', \&) \ni r'$. Donc le langage reconnu par A n'est pas descriptif. Contradiction. Le raisonnement est similaire pour le degré.

Supposons que l'état s de $\overline{A}$ ne soit pas accessible. Si $s \in Q_1$ alors A n'est pas co-accessible. Si $s \in Q_2$ alors il est accessible dans A et il existe $q, q''' \in Q_1$ et $q', q'' \in Q_2$ et μ_1, μ_2 tels que $\delta(q, \#) \ni q'$, $\delta(q', \mu_1) \ni s$, $\delta(s, \mu_2) \ni q''$, $\delta(q'', \&) \ni q'''$. D'après les parties 3 et 4 de la définition 1.2.6, alors il existe dans l'automate $\overline{A}$ deux états $r, r' \in Q_1$ tels que les transitions $\delta'(r, \#) \ni q'$, $\delta'(q', \mu_1) \ni s$, $\delta'(s, \mu_2) \ni q''$, $\delta'(q'', \&) \ni r'$. Donc s est accessible. Contradiction. Le raisonnement est similaire pour la co-accessibilité.□

Cette définition de l'inverse sur les langages rationnels descriptifs est liée à celle que nous avons donné sur les mots descriptifs.

Proposition 1.2.8 *Soit $\mathcal{L}$, un langage avec branchements rationnel descriptif reconnu par un automate fini A dont tous les états sont accessibles et co-accessibles, alors $\overline{A}$ reconnaît le langage $\overline{\mathcal{L}} = \{\overline{\mu} \mid \mu \in \mathcal{L}\}$.*



Les flêches en tirets fins représentent les transitions à l'intérieur de Q_1, celle en tirets épais les transitions à l'intérieur de Q_2.

FIG. 1.3 - *Exemple d'inversion d'un automate.*

Preuve. Montrons que quelque soit μ un mot du langage $\mathcal{L}$, alors le mot inverse selon la définition 1.2.1 est reconnu par $\overline{A}$.

Si μ est sans branchements ($\mu \in \Pi^*$), alors clairement $\overline{\mu}$ est reconnu (partie 1 de la définition 1.2.6 de δ').

Sinon, on découpe μ sous la forme: $\mu = \mu_0\#\mu_1\&\mu_2 \ldots \#\mu_{2n-1}\&\mu_{2n}$ avec $\mu_{2i} \in \Pi^*$ pour $0 \leq i \leq n$, et $\mu_{2i+1} \in \Sigma^*$ pour $0 \leq i \leq n-1$. Nous indiquons, en alternance avec les facteurs de ce découpage, les états atteints lors de la reconnaissance du mot:

$$\mu = q_0\mu_0 q'_0\#q_1\mu_1 q'_1\&q_2\mu_2 q'_2 \ldots q'_{2n-2}\#q_{2n-1}\mu_{2n-1}q'_{2n-1}\&q_{2n}\mu_{2n}q'_{2n}$$

Nous remarquons que $q'_{2n} \in F$, et, plus généralement, pour $0 \leq i \leq n$ alors $q_{2i}, q'_{2i} \in Q_1$, et pour $0 \leq i \leq n-1$ alors $q_{2i+1}, q'_{2i+1} \in Q_2$. Nous constatons:

- D'après la partie 1 de la définition de δ' dans l'automate $\overline{A}$, il existe une transition $\delta'(q'_{2i}, \overline{\mu_{2i}}) \ni q_{2i}$ pour $0 \leq i \leq n$.

- D'après la partie 2, il existe une transition $\delta'(q_{2i+1}, \mu_{2i+1}) \ni q'_{2i+1}$ pour $0 \leq i \leq n-1$.

- D'après la partie 3, il existe une transition $\delta'(q_{2i}, \#) \ni q_{2i-1}$ pour $1 \leq i \leq n$. En effet, q_{2i} tient le rôle de l'état q de la définition, q_{2i-1} tient le rôle de l'état q', q'_{2i-1} tient le rôle de l'état q'', q'_{2i-2} tient le rôle de l'état q''', et enfin μ_{2i-1} tient le rôle du mot μ.

- D'après la partie 4, il existe une transition $\delta'(q'_{2i+1}, \#) \ni q'_{2i}$ pour $0 \leq i \leq n-1$. En effet, q'_{2i} tient le rôle de l'état q de la définition, q'_{2i+1} tient le rôle de l'état q', q_{2i+1} tient le rôle de l'état q'', q_{2i+2} tient le rôle de l'état q''', et enfin μ_{2i+1} tient le rôle du mot μ.

Nous en déduisons que le mot $\overline{\mu} = \overline{\mu_{2n}}\#\mu_{2n-1}\& \ldots \overline{\mu_2}\#\mu_1\&\overline{\mu_0}$ est reconnu par l'automate inverse.

Réciproquement, la proposition 1.2.7 nous autorise à dire que tout mot μ de $\overline{A}$ peut aussi être découpé sous la forme $\mu = \mu_0\#\mu_1\&\mu_2 \ldots \#\mu_{2n-1}\&\mu_{2n}$ avec $\mu_{2i} \in \Pi^*$ pour $0 \leq i \leq n$, et $\mu_{2i+1} \in \Sigma^*$ pour $0 \leq i \leq n-1$. Un raisonnement similaire à celui présenté ci-dessus nous permet alors de déduire que μ est l'inverse d'un mot reconnu par A.□

Chapitre 2

Superposition de figures

Par opposition aux figures, les figures pointées comportent deux points distingués, auxquels on peut faire jouer un rôle homologue à celui joué par les deux extrémités d'un mot de figure. De fait, on fait facilement le lien entre concaténation de mots de figures et concaténation de figures pointées. Il n'existe, à priori, rien de comparable pour les figures, qui sont dépourvues des points par où on pourrait les coller.

Il nous a pourtant semblé intéressant de prendre le point de vue du graphiste, lequel manipule plutôt ces objets bruts que sont les figures de base. On peut alors définir des opérations intuitives sur ces objets, et examiner le degré d'adéquation du codage classique pour représenter le résultat de ces opérations. Nous étudions donc une opération que nous avons appelée superposition, et qui réalise une notion intuitive de "concaténation" des figures de base. La superposition de deux figures f et g, notée $f \odot g$, est l'ensemble des figures que l'on peut obtenir si l'on trace f puis g par dessus, en ayant pour seule contrainte que la figure résultante soit connexe.

2.1 Notion de superposition

Définition 2.1.1 *Soient f et g, deux figures de base, on définit la superposition de f et g par:*

$$f \odot g = \{[d \cup t_c(d')]_\perp \mid d \in f, d' \in g, c \in \mathbb{Z}^2 \text{ avec } d \cup t_c(d') \text{ connexe}\}$$

Un exemple de superposition est donné en Fig. 2.1. On notera que la superpostion est une opération commutative, mais pas associative, comme le montre la Fig. 2.2.

On utilise les notations suivantes: $f^{\odot 0} = \emptyset$, $f^{\odot 1} = \{f\}$, et $f^{\odot n} = f \odot f^{\odot(n-1)}$ pour tout $1 < n$.

Définition 2.1.2 *On appelle* étoile arrondie *la clôture de l'opération de super-*

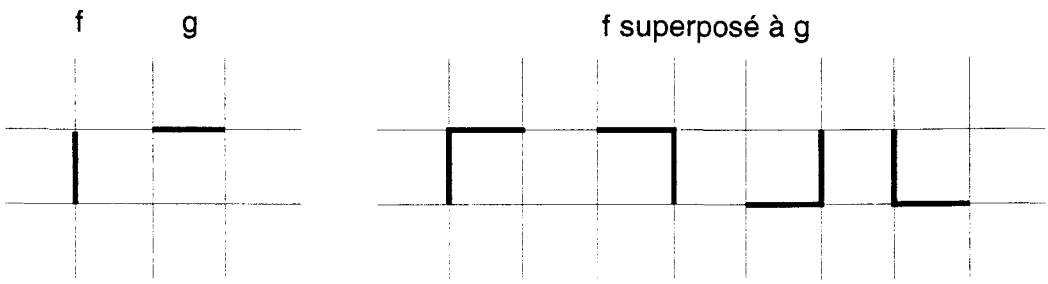
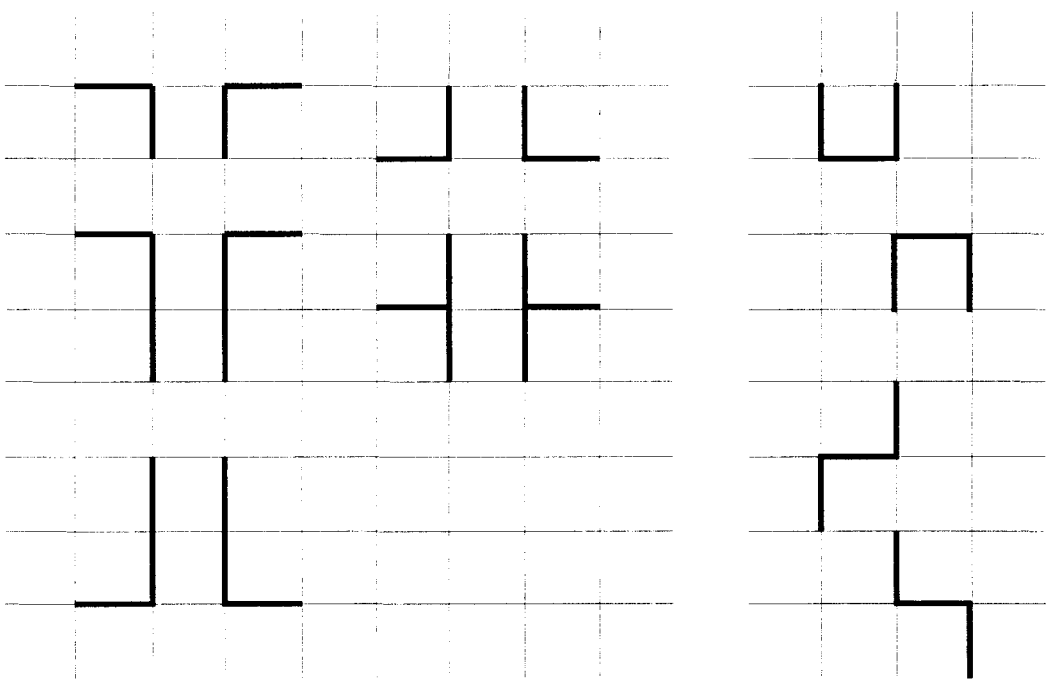


FIG. 2.1 - *Exemple de superposition de deux figures.*



Avec les figures f et g du schéma précédent (Fig. 2.1), nous avons représenté à gauche les figures appartenant aux deux ensembles $(f \odot f) \odot g$ et $f \odot (f \odot g)$, et à droite celles qui sont spécifiques à $f \odot (f \odot g)$.

FIG. 2.2 - *La superposition n'est pas associative.*

position :

$$f^{\otimes} = \bigcup_{n \in I\!N} f^{\odot n}$$

Ces définitions s'étendent aux langages de figures :

Définition 2.1.3 *Soient $\mathcal{G}$, $\mathcal{G}_1$ et $\mathcal{G}_2$, des langages de figures, alors :*

- $\mathcal{G}_1 \odot \mathcal{G}_2 = \bigcup_{\substack{f \in \mathcal{G}_1 \\ g \in \mathcal{G}_2}} f \odot g$
- $\mathcal{G}^{\odot 0} = \emptyset$, $\mathcal{G}^{\odot 1} = \mathcal{G}$, *et* $\mathcal{G}^{\odot n} = \mathcal{G} \odot \mathcal{G}^{\odot(n-1)}$ *pour tout* $1 < n$
- $\mathcal{G}^{\otimes} = \bigcup_{n \in I\!N} \mathcal{G}^{\odot n}$

2.2 Langages de mots de figures classiques

Il est naturel de se demander si la superposition a de bonnes propriétés lorsqu'on l'exprime dans le cadre de la sémantique classique. Nous nous sommes intéressés au cas des langages rationnels, sur lesquels les résultats sont en général les plus favorables.

Théorème 2.2.1 *La superposition et l'étoile arrondie ne préservent pas la rationalité des langages de mots de figures classiques.*

Plus formellement $\exists \mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3 \in \mathrm{Rat}(\Pi^*)$, tels que $\forall \mathcal{R} \in \mathrm{Rat}(\Pi^*)$, alors $\mathrm{Fig}(\mathcal{R}) \neq \mathrm{Fig}(\mathcal{R}_1) \odot \mathrm{Fig}(\mathcal{R}_2)$, et $\mathrm{Fig}(\mathcal{R}) \neq \mathrm{Fig}(\mathcal{R}_3)^{\otimes}$.
Preuve. Par l'absurde. Soient $\mathcal{R}_1 = r^*ddl^*$ et $\mathcal{R}_2 = l^*$, supposons qu'il existe un automate $A =< \sigma, Q, \{q_0\}, F, \delta >$, à n états, reconnaissant le langage $\mathcal{R}$ tel que $\mathrm{Fig}(\mathcal{R}) = \mathrm{Fig}(\mathcal{R}_1) \odot \mathrm{Fig}(\mathcal{R}_2)$. Alors nous pouvons dire que $\mathcal{R}$ contient, parmi d'autres, un mot ω qui représente la figure 2.3-(i), où toutes les branches horizontales sont de même longueur avec $\mathrm{Dist}(c_1, c_2) > n$.

Toutes les façons de dessiner la figure nécessitent bien sûr de tracer les portions de droites $[c_1, c_2]$, $[c_3, c_4]$ et $[c_5, c_6]$. On peut se ramener à l'étude d'un seul cas, en utilisant les symétries. Sans perte de généralité, nous admettons que l'on trace d'abord le segment $[c_1, c_2]$, puis $[c_3, c_4]$, puis $[c_5, c_6]$, les autres cas se résolvant par un raisonnement tout à fait similaire. Alors le crayon passe successivement par les points c_2, c_4, c_3, c_4, c_6 (entre autres). Plus formellement, $\omega = \omega_1\chi_1\chi_2\chi_3\chi_4\omega_2$, avec $\mathrm{Dec}(\omega_1) = c_2$, $\mathrm{Dec}(\omega_1\chi_1) = c_4$, $\mathrm{Dec}(\omega_1\chi_1\chi_2) = c_3$, $\mathrm{Dec}(\omega_1\chi_1\chi_2\chi_3) = c_4$, $\mathrm{Dec}(\omega_1\chi_1\chi_2\chi_3\chi_4) = c_6$, avec ω_1, χ_1, χ_2, χ_3, χ_4, $\omega_2 \in \Pi^*$.

On a $\mathrm{Dist}(c_3, c_4) > n$ et par conséquent, d'après le lemme de l'étoile, il existe au moins deux points distincts sur ce segment de droite qui correspondent au même état de l'automate lors de la lecture de χ_3. Plus formellement, il existe

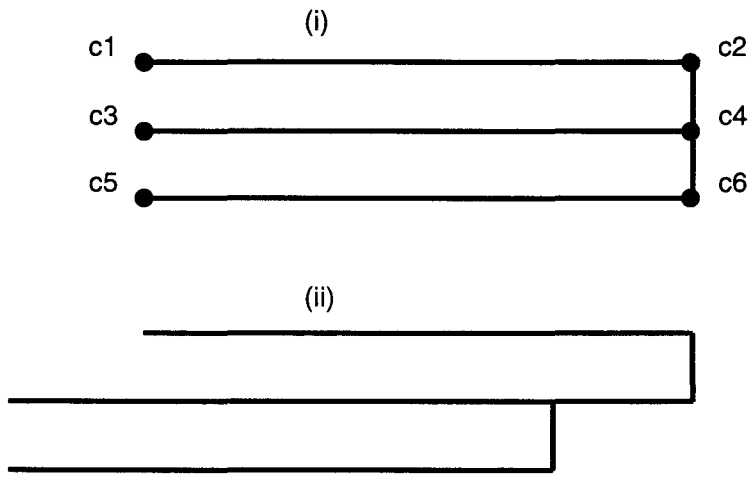


FIG. 2.3 - *Figures appartenant à Fig($\mathcal{R}$).*

$q \in Q, \alpha, \beta, \gamma \in \Pi^*$, avec $|\beta|_r - |\beta|_l > 0$ et $\chi_3 = \alpha\beta\gamma$, tels que $\delta(q_0, \omega_1\chi_1\chi_2\alpha) = q$ et $\delta(q_0, \omega_1\chi_1\chi_2\alpha\beta) = q$. On en déduit que l'automate A reconnaît les mots de la forme $\omega_1\chi_1\chi_2\alpha\beta^*\gamma\chi_4\omega_2$, et Fig($\mathcal{R}$) contient des figures similaires au dessin 2.3-(ii). Ces figures ne sont pas dans Fig($\mathcal{R}_1$) $\odot$ Fig($\mathcal{R}_2$). Contradiction.

Pour l'étoile arrondie il suffit de prendre $\mathcal{R}_3 = \mathcal{R}_1 \cup \mathcal{R}_2$, et l'on obtient la même contradiction. □

2.3 Langages avec branchements

Les langages avec branchements se comportent plus favorablement vis à vis de la superposition.

Théorème 2.3.1 *La superposition de deux langages de figures rationnels descriptifs est un langage de figures rationnel descriptif. L'étoile arrondie d'un langage rationnel descriptif est un langage rationnel descriptif.*

Plus formellement,

$$\forall \mathcal{R}_1, \mathcal{R}_2 \in \mathrm{Rat}(\Sigma^*), \exists \mathcal{R} \in \mathrm{Rat}(\Sigma^*), \text{ tel que :}$$

$$\mathrm{Fig}(\mathcal{R}) = \mathrm{Fig}(\mathcal{R}_1) \odot \mathrm{Fig}(\mathcal{R}_2)$$

et

$$\forall \mathcal{R} \in \mathrm{Rat}(\Sigma^*), \exists \mathcal{S} \in \mathrm{Rat}(\Sigma^*), \text{ tel que :}$$

$$\mathrm{Fig}(\mathcal{S}) = \mathrm{Fig}(\mathcal{R})^{\otimes}$$

Le reste de ce chapitre montre ce résultat. Nous définissons un opérateur, nommé Odes, qui va nous servir à exprimer la superposition, et dont voici une esquisse :

> Soit un mot μ de Σ^*, avec le dessin pointé associé p-Fig$(\mu) = [(e, d, a)]_\top$. Nous voulons que Odes(μ) contienne au moins un mot associé à chaque p-figure obtenue en prenant un point de départ n'importe où sur e et en terminant le tracé en d.

Définition 2.3.2 *Soit un mot avec branchements μ, de degré p, alors Odes(μ) est défini récursivement sur p par :*

- *si $p = 0$, alors* $Odes(\mu) = \{\#\mu_2\&\overline{\mu_1} \mid \mu_1, \mu_2 \in \Pi^*, \mu = \mu_1\mu_2\}$
- *si $p \geq 1$, alors* $Odes(\mu) = \{\#\mu_2\&\overline{\mu_1} \mid \mu_1, \mu_2 \in \Sigma^*, \mu = \mu_1\mu_2\} \cup \{Odes(\mu_2)\#\mu_3\&\overline{\mu_1} \mid \mu_1, \mu_2, \mu_3 \in \Sigma^*, \mu = \mu_1\#\mu_2\&\mu_3\}$

En d'autres termes, dans le cas $p = 0$, on coupe n'importe où dans le mot et on recolle les morceaux pour que le nouveau point de départ soit à l'endroit de la coupure. Dans les autres cas on procède de manière semblable, le premier ensemble de l'union correspond à une coupure dans un facteur de degré 0, le second ensemble à une coupure dans une branche : notons alors que l'appel récursif à Odes(μ_2) se fait sur un mot de degré $p - 1$.

Nous passons aux langages :

Définition 2.3.3 *Soit $\mathcal{L}$ un langage de mots avec branchements, alors :*

$$Odes(\mathcal{L}) = \bigcup_{\mu \in \mathcal{L}} Odes(\mu)$$

Montrons que Odes contient bien les mots que nous souhaitons :

Proposition 2.3.4 *Soient $\mu \in \Sigma^*$, et p-Fig$(\mu) = [(e, d, a)]_\top$, alors :*

$$\forall x \in Arm(e), \exists \omega \in Odes(\mu), p\text{-}Fig(\omega) = [(e, x, d)]_\top$$

Preuve. Nous n'allons pas travailler directement sur les figures, mais sur les représentants standards obtenus par Desp, dont le tracé commence, par définition, à l'origine du plan, notée O. La propriété peut alors être réécrite en :

> Soient $\mu \in \Sigma^*$, et Desp$(\mu) = (e, O, a)$, alors :
>
> $$\forall x \in \mathrm{Arm}(e), \exists \omega \in \mathrm{Odes}(\mu), \mathrm{Desp}(\omega) = (t_{-x}(e), O, -x)$$

Faisons une récurrence sur p, le degré de μ.
Cas 1 : $p = 0$. Par définition d'un dessin associé à un mot, quelque soit x un point de l'armature, il existe une factorisation $\mu = \alpha\beta$, telle que $x = \mathrm{Dec}(\alpha)$. Nous notons $\mathrm{Desp}(\alpha) = (e_\alpha, O, x)$, et $\mathrm{Desp}(\beta) = (e_\beta, O, y)$. Par définition de Odes, nous avons $\mathrm{Odes}(\mu) \ni \#\beta\&\overline{\alpha}$, et :

$$\begin{aligned}
\mathrm{Desp}(\#\beta\&\overline{\alpha}) &\stackrel{\text{def}}{=} \mathrm{Desp}(\beta\overline{\beta}\overline{\alpha}) \\
&= \mathrm{Desp}(\beta\overline{\beta}).\mathrm{Desp}(\overline{\alpha}) \\
&\quad \text{car } \beta\overline{\beta} \text{ est une boucle} \\
&= (e_\beta, O, O).(t_{-x}(e_\alpha), O, -x) \\
&= (t_{-x}(e), O, -x)
\end{aligned}$$

Nous supposons cette propriété vraie quelque soit $p < n$. Nous décomposons le cas $p = n$ en deux sous-cas, selon que le point x appartient à l'armature d'un segment tracé par une lettre dans un facteur de profondeur 0 ou non.
Cas 2a : il existe une factorisation $\mu = \alpha\beta$, avec $\alpha, \beta \in \Sigma^*$, telle que $x = \mathrm{Dec}(\mathrm{Br}(\alpha))$. Nous avons $\mathrm{Odes}(\mu) \ni \#\beta\&\overline{\alpha}$, et :

$$\begin{aligned}
\mathrm{Desp}(\#\beta\&\overline{\alpha}) &\stackrel{\text{def}}{=} \mathrm{Desp}(\mathrm{Br}(\#\beta\&)\mathrm{Br}(\overline{\alpha})) \\
&= \mathrm{Desp}(\mathrm{Br}(\beta)\overline{\mathrm{Br}(\beta)}\mathrm{Br}(\overline{\alpha}))
\end{aligned}$$

Chacun des trois termes $\mathrm{Br}(\beta)$, $\overline{\mathrm{Br}(\beta)}$, $\mathrm{Br}(\overline{\alpha})$ est dans Π^* et nous pouvons alors conclure le raisonnement comme dans le cas $p = 0$.
Cas 2b : il existe une factorisation $\mu = \alpha\#\beta\&\gamma$, avec $\alpha, \beta, \gamma \in \Sigma^*$ et $\mathrm{Br}(\beta) = \beta_1\beta_2$, telle que $x = \mathrm{Dec}(\mathrm{Br}(\alpha)\beta_1)$. Nous posons $\mathrm{Desp}(\alpha) = (e_\alpha, O, a)$, $\mathrm{Desp}(\beta) = (e_\beta, 0, b)$, $\mathrm{Desp}(\gamma) = (e_\gamma, 0, c)$. Nous notons $\mathrm{Dec}(\beta_1) = y$, et par suite $x = y + a$. Par hypothèse de récurrence, $\mathrm{Desp}(\mathrm{Odes}(\beta)) \ni (t_{-y}(e_\beta), O, -y)$, et nous avons $\mathrm{Odes}(\mu) \ni \mathrm{Odes}(\beta)\#\gamma\&\overline{\alpha}$, et donc :

$$\mathrm{Desp}(\mathrm{Odes}(\beta)\#\gamma\&\overline{\alpha}) \ni (t_{-y}(e_\beta), O, -y).t'_{-y}(\mathrm{Desp}(\#\gamma\&\overline{\alpha})$$

Alors :

$$\begin{aligned}
(t_{-y}(e_\beta), O, -y).t'_{-y}(\mathrm{Desp}(\#\gamma\&\overline{\alpha}) &= (t_{-y}(e_\beta), O, -y).t'_{-y}(\mathrm{Desp}(\mathrm{Br}(\gamma)\overline{\mathrm{Br}(\gamma)})) \\
&\quad .t'_{-y}(\mathrm{Desp}(\mathrm{Br}(\overline{\alpha}))) \\
&= (t_{-y}(e_\beta), O, -y).(t_{-y}(e_\gamma), -y, -y) \\
&\quad .(t_{-y-a}(e_\alpha), -y, -y - a) \\
&= (t_{-y-a}(e), O, -y - a) \\
&= (t_{-x}(e), O, -x)
\end{aligned}$$

□

Tous les mots de $\mathrm{Odes}(\mu)$ dessinent la même figure que μ :

Proposition 2.3.5 *Soit $\mu \in \Sigma^*$, alors $\forall \omega \in Odes(\mu)$, $Fig(\omega) = Fig(\mu)$.*

Preuve. La preuve se fait par récurrence sur le degré de μ, de manière similaire à la preuve précédente. □

Il nous reste à montrer que Odes conserve la rationnalité et la descriptivité.

Définition 2.3.6 *Soit $\mathcal{L}$ un* LAB *rationnel descriptif de degré p, reconnu par un automate $R =< \Pi_b, Q, \{q_0\}, F, \delta >$, nous notons $\mathcal{L}_1$ et $\mathcal{L}_2$ les langages, déduits de R, suivants :*

- $\mathcal{L}_1 = \bigcup_{\substack{q_1 \in Q \\ q_f \in F}} \#r_{1,q_1q_f}\&\overline{r_{2,q_0q_1}}$
  - $r_{1,q_0q_1} = R_{q_0q_1} \cap \Sigma_p$
  - $r_{2,q_1q_f} = R_{q_1q_f} \cap \Sigma_p$
- $\mathcal{L}_2 = \bigcup_{\substack{q_1, q_2 \in Q \\ q_f \in F}} Odes(r_{3,q_1q_2})\#r_{4,q_2q_f}\&\overline{r_{5,q_0q_1}}$
  - $r_{3,q_0q_1} = ((R_{q_0q_1})\#^{-1}) \cap \Sigma_p$
  - $r_{4,q_1q_2} = R_{q_1q_2} \cap \Sigma_{p-1}$
  - $r_{5,q_2q_f} = (\&^{-1}(R_{q_2q_f})) \cap \Sigma_p$

Proposition 2.3.7 *Soit L un* LAB *rationnel descriptif de degré p, alors :*

- *si* $p = 0$, $Odes(\mathcal{L}) = \mathcal{L}_1$
- *si* $p \geq 1$, $Odes(\mathcal{L}) = \mathcal{L}_1 \cup \mathcal{L}_2$

Preuve. Clairement, par construction de $\mathcal{L}_1$ et $\mathcal{L}_2$.

Proposition 2.3.8 *Soit $\mathcal{L}$ un* LAB *rationnel descriptif de degré p, alors $Odes(\mathcal{L})$ est un langage rationnel descriptif.*

Preuve. Par récurrence sur p. Si $p = 0$ alors $\text{Odes}(\mathcal{L}) = \mathcal{L}_1$ est rationnel par construction. On suppose la proposition vraie pour tout $p < n$. Si $p = n$, alors $\text{Odes}(\mathcal{L}) = \mathcal{L}_1 \cup \mathcal{L}_2$ avec $\mathcal{L}_1$ rationnel par construction. Notons que r_{3,q_1q_2} est de degré $p-1$, et, par hypothèse de récurrence, $\text{Odes}(r_{3,q_1q_2})$ est rationnel. Donc $\mathcal{L}_2$ est rationnel lui aussi. □

Appliqué à un langage rationnel descriptif, l'opérateur Odes conserve la descriptivité, et augmente le degré au plus de 1.

Proposition 2.3.9 *Soit $\mathcal{L} \in Rat(\Sigma_p)$, alors $Odes(\mathcal{L}) \in \Sigma_{p+1}$.*

Preuve. Par récurrence sur p, de manière similaire à la preuve précédente. □

La proposition suivante va nous servir par la suite à aborder la superposition itérée.

Proposition 2.3.10 *Soit $\mathcal{L} \in Rat(\Sigma^*)$, et $\mathcal{G} = p\text{-}Fig((Odes(\mathcal{L}) + \overline{Odes(\mathcal{L})})^*)$ un langage de p-figures . Quelque soit une figure pointée de $\mathcal{G}$, toutes les figures pointées ayant même base sont aussi dans $\mathcal{G}$. Plus formellement :*

$$\forall [(e,d,a)]_\top \in \mathcal{G}, \forall x,y \in Arm(e), \text{ alors } [(e,x,y)]_\top \in \mathcal{G}$$

Preuve. Quelque soit $\mu \in (\mathrm{Odes}(\mathcal{L})+\overline{\mathrm{Odes}(\mathcal{L})})^*$, avec $\mathrm{Desp}(\mu) = (e,O,a)$, quelque soient $x,y \in \mathrm{Arm}(e)$, par définition d'un dessin pointé il existe une factorisation : $\mu = \mu_1\mu_2\ldots\mu_i\ldots\mu_j\ldots\mu_{k-1}\mu_k$, avec $\mathrm{Br}(\mu_i) = \alpha_1\alpha_2$ et $\mathrm{Br}(\mu_j) = \beta_1\beta_2$ telle que : $x = \mathrm{Dec}(\mathrm{Br}(\mu_1)\mathrm{Br}(\mu_2)\ldots\alpha_1)$ et $y = \mathrm{Dec}(\mathrm{Br}(\mu_1)\mathrm{Br}(\mu_2)\ldots\mathrm{Br}(\mu_i)\ldots\beta_1)$. Par définition de Odes, il existe alors $\omega_1 \in \mathrm{Odes}(\mu_i)$ et $\omega_2 \in \overline{\mathrm{Odes}(\mu_j)}$, tels que : $\zeta = \omega_1\ldots\overline{\mu_2}\,\overline{\mu_1}\mu_1\mu_2\ldots\mu_i\ldots\mu_j\ldots\mu_{k-1}\mu_k\overline{\mu_{k-1}}\,\overline{\mu_k}\ldots\omega_2$ soit un mot de $(\mathrm{Odes}(\mathcal{L})+\overline{\mathrm{Odes}(\mathcal{L})})^*$, et $Figp(\zeta) = [(e,x,y)]_\top$. □

L'opérateur Odes nous permet de décrire la superposition et d'obtenir nos principaux résultats.

Lemme 2.3.11 *Soit deux langages avec branchements rationnels descriptifs $L_1 \in Rat(\Sigma^*)$ et $L_2 \in Rat(\Sigma^*)$, alors*

$$Fig(\mathcal{L}_1) \odot Fig(\mathcal{L}_2) = Fig(\overline{Odes(\mathcal{L}_1)}.Odes(\mathcal{L}_2))$$

Preuve. Nous montrons que $\mathrm{Fig}(\mathcal{L}_1) \odot \mathrm{Fig}(\mathcal{L}_2) \subset \mathrm{Fig}(\overline{\mathrm{Odes}(\mathcal{L}_1)}.\mathrm{Odes}(\mathcal{L}_2))$.

Soit $f \in \mathrm{Fig}(\mathcal{L}_1)\odot\mathrm{Fig}(\mathcal{L}_2)$ alors, par définition, il existe $\mu_1 \in \mathcal{L}_1, \mu_2 \in \mathcal{L}_2$, avec $f_1 = \mathrm{Fig}(\mu_1)$, $f_2 = \mathrm{Fig}(\mu_2)$ tels que $f = f_1 \odot f_2$. Soient $\mathrm{Desp}(\mu_1) = (e_1,O,a)$ et $\mathrm{Desp}(\mu_2) = (e_2,O,b)$, alors, par définition de la superposition, il existe $c, x \in \mathbb{Z}^2$, tels que $f = [e_1 \cup t_c(e_2)]_\perp$, et, comme $e_1 \cup t_c(e_2)$ est connexe, $x \in \mathrm{Arm}(e_1)$, $x - c \in \mathrm{Arm}(e_2)$. On a :

$$\begin{aligned} f &= [e_1 \cup t_c(e_2)]_\perp \\ &= [\mathrm{Base}((e_1 \cup t_c(e_2), O, c))]_\perp \\ &= [\mathrm{Base}((e_1,O,x).(t_c(e_2),x,c))]_\perp \\ &= [\mathrm{Base}((e_1,O,x).t_c(e_2,x-c,O))]_\perp \end{aligned}$$

Par définition de Odes, il existe $\omega_1 \in \overline{\mathrm{Odes}(\mu_1)}$ tel que $\mathrm{Desp}(\omega_1) = (e_1,O,x)$, et $\omega_2 \in \mathrm{Odes}(\mu_2)$ tel que $\mathrm{Desp}(\omega_2) = (e_2,x-c,O)$. Par conséquent :

$$\begin{aligned} f &= [\mathrm{Base}(\mathrm{Desp}(\omega_1).t_c(Desp(\omega_2)))]_\perp \\ &= [\mathrm{Base}(\mathrm{Desp}(\omega_1\omega_2))]_\perp \\ &= \mathrm{Fig}(\omega_1\omega_2) \end{aligned}$$

Nous laissons au lecteur la preuve de la réciproque, qui se construit de manière semblable. □

Corollaire 2.3.12 *Soit deux langages avec branchements rationnels descriptifs* $L_1 \in \Sigma_{p_1}$ *et* $L_2 \in \Sigma_{p_2}$, *alors:*

$$Deg(Fig(\mathcal{L}_1) \odot Fig(\mathcal{L}_2)) \leq sup(p_1, p_2) + 1$$

Preuve. Immédiat d'après 2.3.9 et 2.3.11. □

Lemme 2.3.13 *Soit un langage avec branchements rationnel descriptif* $L \in \Sigma_p$, *alors*

$$(Fig(\mathcal{L}))^{\otimes} = Fig((Odes(\mathcal{L}) + \overline{Odes(\mathcal{L})})^*)$$

Preuve. Nous appelons $L = \text{Odes}(\mathcal{L})$, pour clarifier la notation. Montrons l'inclusion : $(\text{Fig}(\mathcal{L}))^{\otimes} \subset \text{Fig}((L + \overline{L})^*)$. Par définition, $(\text{Fig}(\mathcal{L}))^{\otimes} = \bigcup_{i \in \mathbb{N}} \text{Fig}(\mathcal{L})^{\odot i}$. Nous faisons une récurrence sur i. Si $i = 0$ ou $i = 1$, alors l'inclusion est clairement vérifiée. Supposons la vérifiée pour tout $i < n$. Dans le cas $i = n$, alors :

$$\begin{aligned}
\text{Fig}(\mathcal{L})^{\odot n} &\overset{\text{def}}{=} \text{Fig}(\mathcal{L}) \odot \text{Fig}(\mathcal{L})^{\odot n-1} \\
&\overset{\text{rec}}{\subset} \text{Fig}(\mathcal{L}) \odot \text{Fig}((L + \overline{L})^*) \\
&\overset{2.3.11}{\subset} \text{Fig}(\overline{L}.\text{Odes}((L + \overline{L})^*)) \\
&\subset \text{Base}(\text{p-Fig}(\overline{L}).\text{p-Fig}(\text{Odes}((L + \overline{L})^*))) \\
&\overset{2.3.10}{\subset} \text{Base}(\text{p-Fig}(\overline{L}).\text{p-Fig}((L + \overline{L})^*)) \\
&\subset \text{Base}(\text{p-Fig}(\overline{L}(L + \overline{L})^*)) \\
&\subset \text{Base}(\text{p-Fig}((L + \overline{L})^*) \\
&\subset \text{Fig}((L + \overline{L})^*)
\end{aligned}$$

Montrons maintenant l'inclusion réciproque. Soit f une figure dans le langage $\text{Fig}((L + \overline{L})^*)$, nous avons :

$$\begin{aligned}
\text{Fig}((L + \overline{L})^*) &= \text{Base}(\text{p-Fig}((L + \overline{L})^*)) \\
&= \text{Base}((\text{p-Fig}(L) + \text{p-Fig}(\overline{L}))^*)
\end{aligned}$$

Et donc, $f = \text{Base}(f_1.f_2. \ \ldots \ .f_n)$, $n \in \mathbb{N}$ et $f_i \in \text{p-Fig}(L) \cup \text{p-Fig}(\overline{L})$, pour tout $1 \leq i \leq n$. Il existe donc des dessins pointés $(e_i, d_i, a_i) \in f_i$, pour tout $1 \leq i \leq n$, tels que $f = \text{Base}([e_1.e_2. \ \ldots \ .e_n]_{\top})$, et $d_i = a_{i-1}$, pour tout $1 < i \leq n$. Clairement $f \subset \text{Base}(e_1) \odot \text{Base}(e_2) \ldots \text{Base}(e_n)$. Nous en déduisons $f \in (\text{Fig}(\mathcal{L}))^{\otimes}$. □

Corollaire 2.3.14 *Soit* $\mathcal{L} \in \Sigma_p$, *alors* $Deg(Fig(\mathcal{L})^{\otimes}) \leq p + 1$

Preuve. Immédiat d'après 2.3.9 et 2.3.13. □

Le théorème 2.3.1 dérive immédiatement des lemmes 2.3.11 et 2.3.13.

La superposition peut donc s'exprimer de manière rationnelle dans les langages avec branchements descriptifs, ce qui est une propriété assez agréable. De plus cela se fait sans augmenter de manière importante le degré des langages considérés.

Quatrième partie

Langages à pixels

Chapitre 1

Sémantique des langages à pixels

Dans notre introduction, nous avons signalé l'intérêt d'une classe particulière de figures : les polyominos. En effet ces motifs sont notamment rencontrés dans les études de pavages et l'analyse combinatoire, et ils peuvent aussi modéliser certains problèmes physiques. Du point de vue du graphiste, il permettent aussi de ne pas se cantonner dans les structures filiaires que nous avons principalement rencontrées jusqu'ici, et d'envisager la description de motifs "pleins", comme les aplats de couleurs, par exemple. Avant de poursuivre, nous allons définir précisément le concept de polyomino.

1.1 Pixels et polyominos

Nous formalisons tout d'abord la notion de pixel, et, pour un ensemble de pixels, nous précisons certains ensembles de points du plan qui coïncident avec les pixels et qui nous seront utiles par la suite.

Définition 1.1.1 *Un* pixel *est la surface du plan discret $\mathbb{Z}^2$ définie par $Pix(i,j) = [i, i+1] \times [j, j+1]$, avec $i, j \in \mathbb{Z}$. Par exemple, si $p = [2,3] \times [5,6]$, on note $p = Pix(2,5)$.*

Définition 1.1.2 *L'*armature *du pixel $p = Pix(i,j)$, est la partie de $\mathbb{Z}^2$ définie par $Arm(p) = \{(i,j),(i,j+1),(i+1,j),(i+1,j+1)\}$. C'est à dire l'ensemble des quatre points de $\mathbb{Z}^2$ qui coïncident avec les coins de p. L'armature d'un ensemble de pixels e est l'union des armatures des pixels de e, soit $Arm(e) = \bigcup_{p \in e} Arm(p)$.*

Un exemple de pixels et d'armatures est donné en Fig. 1.1. Comme dans le cas des segments, des ensembles de pixels différents peuvent avoir même armature.

La notion de connexité qui nous intéresse correspond à ce que l'on appelle habituellement la 4-connexité en géométrie discrète. Nous considérons donc comme étant non connexe tout ensemble de pixels 8-connexe mais pas 4-connexe. Ce choix

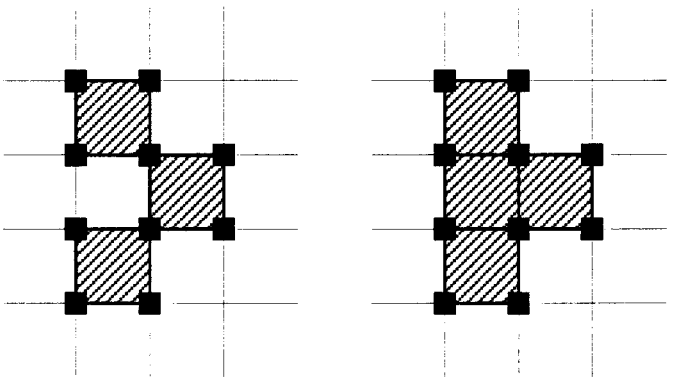
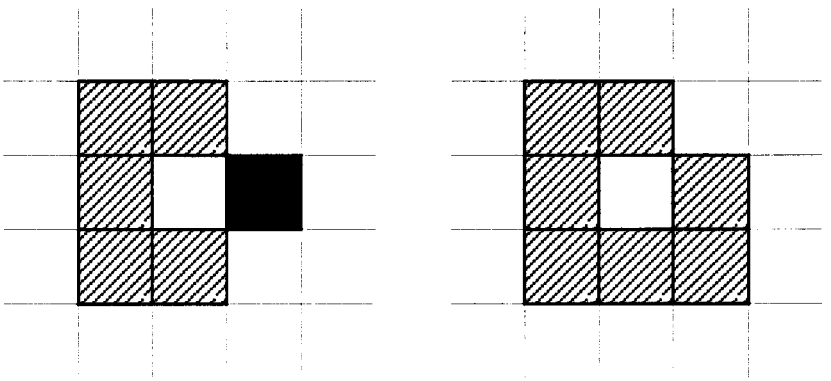


FIG. 1.1 - *Pixels et armatures.*



L'ensemble de pixels situé à gauche n'est pas connexe (le pixel en noir n'est connecté que par des coins). Celui à droite est connexe.

FIG. 1.2 - *Pixels et connexité.*

est motivé par le fait que l'étude présentée dans cet ouvrage sera centrée sur les polyominos, qui sont des objets 4-connexes.

Définition 1.1.3 *Un ensemble de pixels e est* connexe *si et seulement si quelque soient p, q deux pixels de e, il existe une suite de pixels* $\{p_0, \ldots, p_n\}$ *dans e tels que :* $\forall i \in [0, n-1]$, $Card(Arm(p_i) \cap Arm(p_{i+1})) = 2$ *et* $p = p_0$, $q = p_n$.

C'est à dire que si l'on prend une paire de pixels quelconques de l'ensemble, alors on peut aller de l'un à l'autre en empruntant une suite de pixels de l'ensemble tels que chacun d'eux est contigu par un côté avec le suivant (voir Fig. 1.2).

Définition 1.1.4 *Une* composante connexe *d'un ensemble de pixels e est un ensemble non vide de pixels f tel que :*

- $f \subset e$

- f *connexe*
- $\forall p \in e \setminus f \Rightarrow f \cup \{p\}$ *non connexe*

Définition 1.1.5 *Un* polyomino *est un ensemble fini connexe de pixels.*

Un exemple de polyomino est donc donné dans la partie droite de la Fig. 1.2. Notons qu'un polyomino ne contient qu'une seule composante connexe. Nous allons séparer les polyominos en différentes classes, dont ceux avec ou sans trous. Pour cela, le concept de *l'extérieur* d'un ensemble va nous être utile.

Définition 1.1.6 *L'*extérieur *d'un ensemble fini de pixels e, noté Ext_e, est le plus grand, au sens de l'inclusion, des ensembles infinis de pixels connexes qui ne contiennent pas de pixels appartenant à e. Un point p du plan $\mathbb{Z}^2$ est à l'extérieur de e si et seulement si il existe un pixel i dans l'extérieur de e tel que p appartienne à l'armature de i.*

Définition 1.1.7 *Soient e un ensemble fini de pixels, un ensemble de pixels non vide f est un* trou *de e si et seulement si:*

- f *est connexe*
- $f \cap e = \emptyset$
- $f \cap Ext_e = \emptyset$

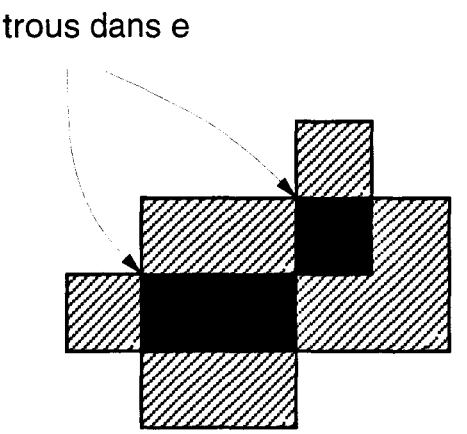
La Fig. 1.3 illustre ces définitions. Notons qu'un trou est forcément un ensemble fini.

Proposition 1.1.8 *Soit f et e deux figures. Si f est un trou de e, alors aucun des pixels de f n'est connexe avec l'extérieur de e.*

Preuve. Si f est un trou de e alors par définition $f \cap e = \emptyset$ et $f \cap \mathrm{Ext}_e = \emptyset$. Appelons $E = \mathrm{Ext}_e \cup f$. Supposons qu'il existe p dans f connexe avec Ext_e. Par conséquent E est connexe, $E \cap e = \emptyset$, et $\mathrm{Ext}_e \subsetneq E$. Contradiction avec la définition de l'extérieur d'un ensemble de pixels.

1.2 Une nouvelle sémantique

Nous avons vu, à la section 3.1, comment représenter un polyomino par un mot de contour. Néanmoins, si l'on cherche à décrire un polyomino quelconque et non pas sans trous, le système précédent doit être adapté. Parmi les diverses méthodes qui peuvent être envisagées, nous en citons deux plus particulièrement intéressantes.



Les pixels de l'ensemble e sont représentés hachurés. L'ensemble e contient deux trous dont les pixels sont coloriés en noir. Tous les autres pixels du plan sont dans l'extérieur de e.

FIG. 1.3 - *Trou et extérieur d'un ensemble de pixels.*

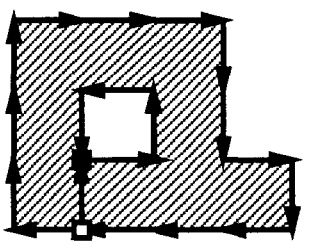


FIG. 1.4 - *Polyomino décrit par* $lu^3l^3d^2rdl^3uruld$.

La première méthode consiste à autoriser le mot de contour à traverser l'intérieur du polyomino, pour atteindre et contourner chaque trou. Alors on prendra souvent comme convention que la frontière intérieure, autour des trous, est parcourue en sens inverse de la frontière extérieure du polyomino, ceci afin de pouvoir en établir la distinction : l'intérieur du polyomino se trouve toujours à main droite (voir Fig. 1.4). Cette technique permet d'obtenir des résultats, notamment pour pouvoir caractériser l'existence d'un pavage. Elle a toutefois un inconvénient du point de vue du graphiste : la figure ainsi décrite ne correspond plus à ce que l'on pourrait obtenir d'une image numérisée et détourée.

Une deuxième méthode consiste à considérer que les mots de Π^* représentent un parcours de l'animal du polyomino (*i.e.* son intérieur), les cellules étant alors les intersections de la grille (Fig. 1.5). Ce principe ne permet toutefois de dessiner que les polyominos, à l'exclusion de toute figure non 4-connexe.

Le système de représentation que nous proposons, quant à nous, ressemble au précédent mais en augmente le pouvoir de description. En effet, il permet de

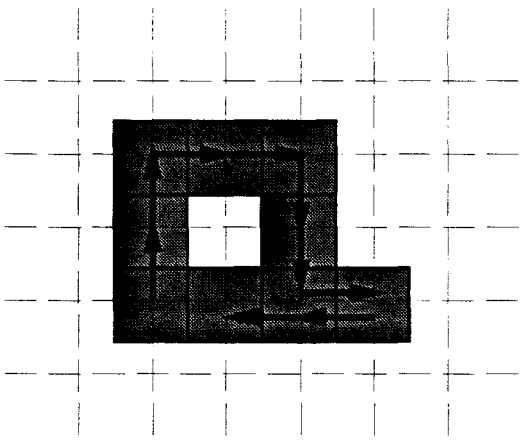


FIG. 1.5 - *Animal décrit par* $u^2r^2d^2rl^2$.

décrire n'importe quelle figure 8-connexe, et donc il peut être utilisé non seulement pour tous les polyominos, mais aussi dans un contexte plus étendu. Ainsi notre procédé permet, par exemple, de décrire tous les animaux dirigés à racine compacte (voir [BM94]), lesquels ne sont pas forcément 4-connexes. Nous nous placerons toujours sur le plan cartésien $\mathbb{Z}^2$ et nous continuerons d'utiliser un alphabet à quatre symboles pour coder les déplacements sur la grille, alphabet que nous noterons Π_p afin d'éviter toute confusion avec les sémantiques présentées précédemment. À chaque déplacement, nous ne tracerons pas un segment, mais nous allumerons le pixel (c'est à dire le carré unitaire) situé à droite du déplacement. Une première étude de cette méthode a été faite dans [LRRS95] où sont abordés entre autres les problèmes de complexité descriptive.

En fait ce système est très apparenté à l'ancien, même s'il en diffère sur certains points. Plutôt que d'en donner une définition complète, indépendante du premier, ce qui conduirait à de nombreuses répétitions, nous avons jugé préférable de restreindre notre exposé aux notions remaniées dans cette sémantique, en les accompagnant de schémas explicatifs. C'est l'objet des deux sections suivantes.

1.3 Langages de figures à pixels

Nos dessins et figures seront donc dorénavant composés de pixels. Une définition supplémentaire nous sera nécessaire pour la suite, celle de l'ensemble des points de la grille situés sur la frontière d'un dessin (voir Fig. 1.6).

Définition 1.3.1 *Le* bord *d'un ensemble de pixels e est défini par* $Bord(e) = Arm(e) \cap Arm(f)$ *avec f l'ensemble des pixels qui n'appartiennent pas à e.*

Par ailleurs, les notions de dessins, dessins pointés, figures, figures pointées et langages de figures et de p-figures sont tout à fait similaires à celles développées

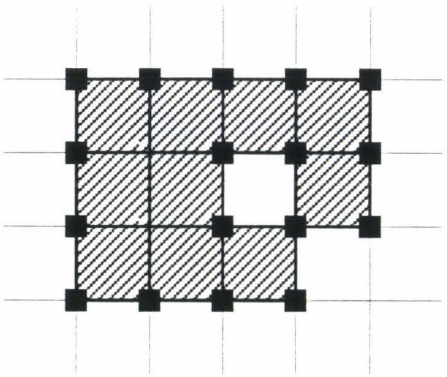


FIG. 1.6 - *Bord d'un ensemble de pixels.*

dans la première partie, à la seule exception qu'elles portent désormais sur des ensembles de pixels plutôt que sur des ensembles de segments. Un dessin est donc un ensemble de pixels, un dessin pointé est un triplet composé d'un dessin et de deux points de $\mathbb{Z}^2$. Les figures et p-figures sont toujours des classes d'équivalence respectivement de dessins et de dessins pointés, modulo une translation dans le plan. Dessins pointés et p-figures peuvent, bien entendu, être concaténés via leurs points distingués. Comme annoncé plus haut, nous ne détaillons pas davantage ces notions, et nous continuons par quelques remarques.

Nous affinons notre classification en définissant les dessins pointés et figures pointées prolongeables :

Définition 1.3.2 *Un dessin pointé* $f = (e, d, a)$ *est* prolongeable *si et seulement si* $\{d, a\} \cap Bord(e) \neq \emptyset$.

Une p-figure prolongeable *est une classe d'équivalence de dessins pointés prolongeables.*

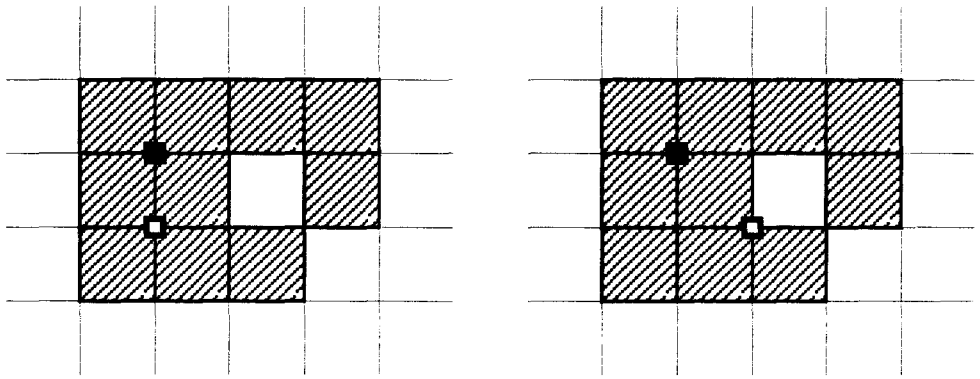
C'est à dire qu'il existe au moins une possibilité de "prolonger" le dessin pointé ou la p-figure en concaténant devant ou derrière un pixel qui ne lui appartient pas encore (voir Fig. 1.7).

La connexité étant évidemment une propriété conservée par translation, nous appellerons aussi polyomino toute figure connexe, et nous noterons $\mathcal{P}$ l'ensemble des polyominos figures et $\mathcal{P}'$ l'ensemble des polyominos figures pointées.

1.4 Langages de mots à pixels

Dans la lignée des travaux de Maurer *et al.*, nous représentons les dessins et figures par des mots sur un alphabet à quatre lettres, dont la sémantique, définie ci-dessous, s'écarte par certains points de l'usage classique.

Définition 1.4.1 *Soit l'alphabet* $\Pi_p = \{u, r, d, l\}$. *Un* mot de figure à pixels, *est un mot fini sur* Π_p. *Pour faciliter la rédaction de certaines preuves, nous rempla-*



La p-figure de gauche n'est pas prolongeable. Celle de droite est prolongeable : on peut lui concaténer un nouveau pixel (à l'emplacement du trou).

FIG. 1.7 - *P-figure non prolongeable et p-figure prolongeable.*

cerons parfois chaque lettre de Π_p *par* x_i, *avec* i *dans* $\mathbb{Z}/4\mathbb{Z}$, *selon la convention suivante :* $u = x_0$, $r = x_1$, $d = x_2$ *et* $l = x_3$.

Plus simplement, nous appellerons encore ces mots, "mots à pixels", ou même "mots de figures", le contexte établissant clairement que nous nous situons dans la sémantique définie sur l'alphabet Π_p.

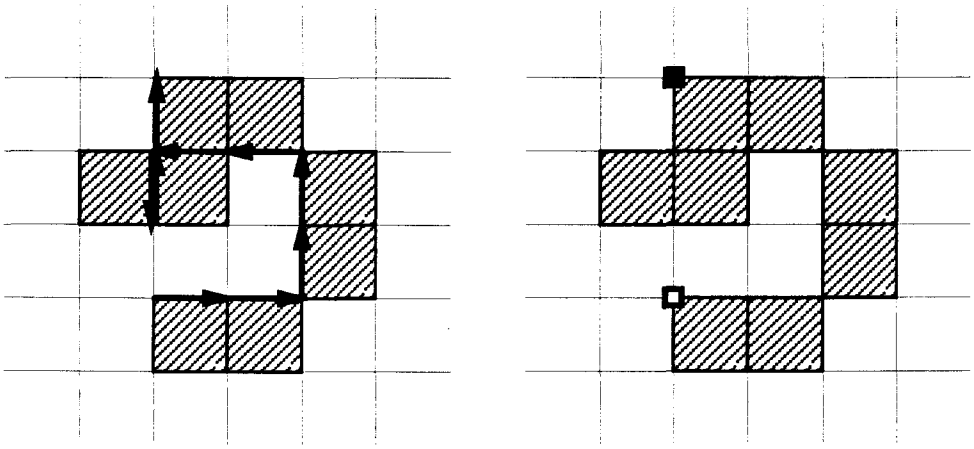
Le concept de décalage est identique à celui défini dans les langages classiques :

Définition 1.4.2 *Chaque lettre de* Π_p *code un déplacement sur le plan* $\mathbb{Z}^2$ *considéré comme une grille. La lettre* u, *respectivement* r, d, l *code un déplacement vers le haut, respectivement vers la droite, le bas, la gauche (de l'anglais up, right, down, left). Soit* μ *un mot de figure, le* décalage *associé à* μ, *noté* $Dec(\mu)$ *est un point de* $\mathbb{Z}^2$ *donné par le morphisme de* $(\Pi_p^*, ., \varepsilon)$ *dans* $(\mathbb{Z}^2, +, (0,0))$ *défini par :*

$$\begin{cases} u & \mapsto & (0,+1) \\ r & \mapsto & (+1,0) \\ d & \mapsto & (0,-1) \\ l & \mapsto & (-1,0) \end{cases}$$

La définition d'un dessin devient :

Définition 1.4.3 *Le dessin associé à un mot de figure* μ, *noté* $Des(\mu)$, *est défini récursivement par :*



À gauche, le dessin avec les flêches indiquant sa construction, à droite le dessin pointé. Notons que le dernier pixel colorié par le mot a été allumé deux fois.

FIG. 1.8 - *Dessin associé au mot à pixel* $r^2u^2l^2du^2$.

$$
\begin{array}{lcl}
Des(\varepsilon) & = & \emptyset \\
Des(u) & = & Pix(0,0) \\
Des(r) & = & Pix(0,-1) \\
Des(d) & = & Pix(-1,-1) \\
Des(l) & = & Pix(-1,0) \\
Des(\mu x_i) & = & Des(\mu) \cup t_{m,n}(Des(x_i)) \text{ avec } (m,n) = Dec(\mu), \mu \in \Pi^*, x_i \in \Pi
\end{array}
$$

En d'autre termes, on place le crayon à l'origine, on le déplace selon les directions indiquées par les lettres du mots et lorsque l'on passe d'un point du plan à un de ses voisins, on *colorie* le pixel situé à droite de l'arête parcourue (Fig. 1.8). Nous dirons aussi que nous *allumons* le pixel, que nous le *dessinons* ou encore que nous le *traçons.*

Les notions de dessins pointés, figures et figures pointées associés à un mot sont, relativement au dessin, similaires à celles présentées dans le cadre des mots de figures classiques. De même pour les langages de figures et de p-figures associés ou représentés par un langage de mots de figures à pixels.

Nous faisons toutefois remarquer que la notion de mot inverse ne s'étend pas immédiatement à cette nouvelle sémantique : le choix d'allumer le pixel à droite du déplacement induit de fait une "polarisation", et donc la figure représentée par ur n'est pas semblable à celle représentée par ld (voir Fig. 1.9). Il est possible de définir dans les langages de mots de pixels une notion similaire à celle de l'inverse dans les mots de figures classiques. Cette étude, présentée dans [LRRS95], ne sera pas abordée ici.

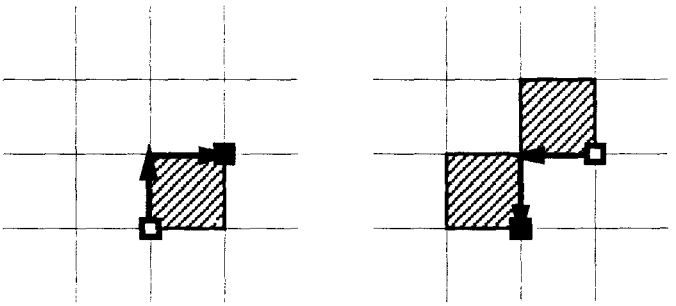


FIG. 1.9 - *P-figures associées à ur et ld.*

Dans le chapitre suivant nous focaliserons notre étude sur des figures pouvant être décrite par un mot où chaque lettre allume un pixel différent :

Définition 1.4.4 *Un mot μ est* optimal *si et seulement si $|Des(\mu)| = |\mu|$. En d'autre termes, chaque pixel n'est colorié qu'une seule fois. Un dessin (respectivement dessin pointé) e est optimal si et seulement si il existe un mot optimal μ tel que $Des(\mu) = e$ (respectivement $Desp(\mu) = e$). Une figure (respectivement figure pointée) f est optimale si et seulement si il existe un mot optimal μ tel que $Fig(\mu) = f$ (respectivement $p\text{-}Fig(\mu) = f$).*

Chapitre 2

Polyominos et collages.

On peut caractériser les figures que nous manipulons par une propriété intuitive : nous associons l'attribut "solides" à celles qui sont connexes, par opposition aux autres, que nous qualifions de "fragiles" car certains blocs de pixels ne sont reliés entre eux que par des liaisons ponctuelles. Bien sûr les figures solides ne sont autres que les polyominos, présentés au chapitre 1. Nous avons constaté dans la partie précédente qu'une opération de collage définie sur les figures de base n'a pas de bonnes propriétés relativement aux langages de mots sans branchements, et nous avons donc dirigé nos investigations vers un collage défini sur les figures pointées. Nous avons visé à nous rapprocher de la problématique du pavage par des polyominos, et ainsi nous avons opté pour un collage opérant sur des figures solides, ayant pour résultat une figure solide, et s'effectuant sans chevauchement ou recouvrement. C'est cette opération que nous définissons formellement dans la section suivante. Enfin nous présentons une étude de ce collage dans la suite du chapitre.

2.1 Collages

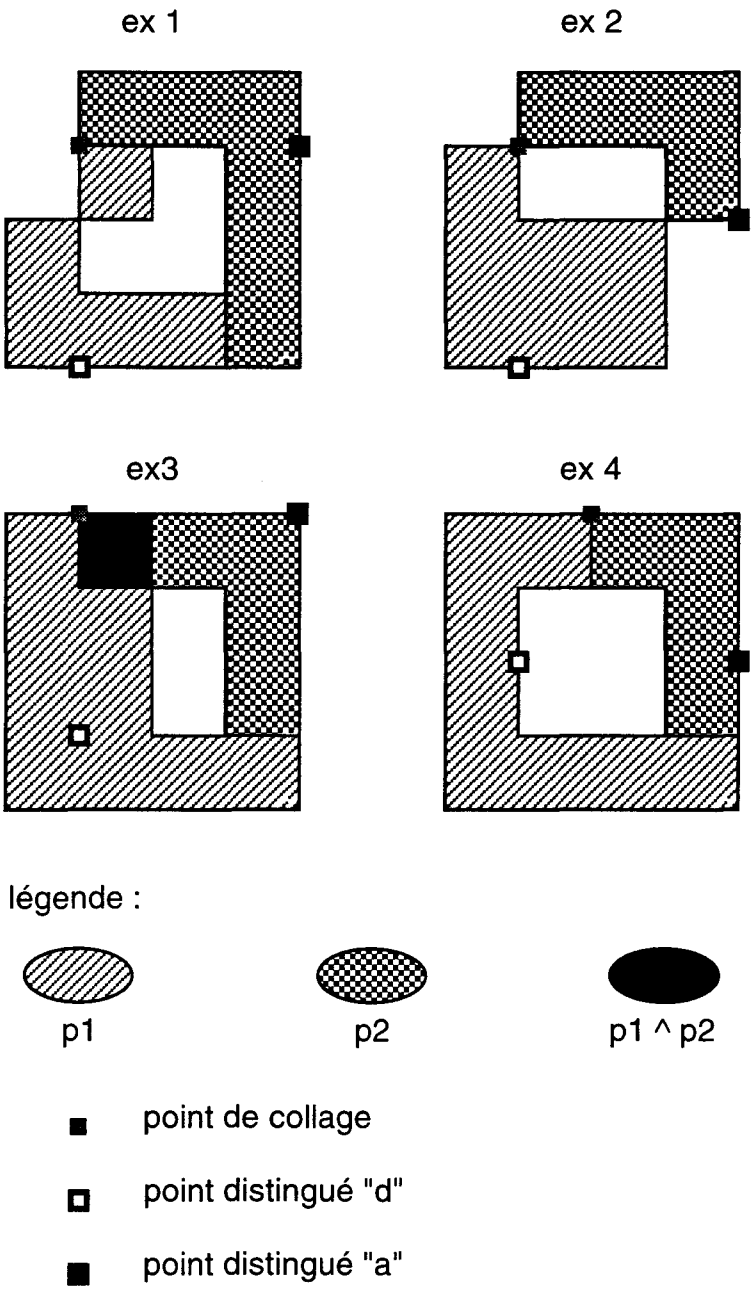
Nous définissons une opération de collage, qui consiste à concaténer deux polyominos pointés, sans chevauchement, pour obtenir un polyomino pointé :

Définition 2.1.1 *Soient deux p-figures p et p', qui sont des polyominos pointés. Le* collage *de p avec p', noté $p \oplus p'$, est défini si et seulement si $p.p'$ est un polyomino pointé et $|p.p'| = |p| + |p'|$. On a alors $p \oplus p' = p.p'$.*

La condition $|p.p'| = |p| + |p'|$ assure qu'il n'y a pas chevauchement. Des exemples de collages impossibles ou possibles sont exposés dans en Fig. 2.1

Fait 2.1.2 *Le collage n'est ni commutatif, ni associatif.*

Nous illustrons ce fait par deux exemples de non commutativité : p-Fig(r) $\oplus$ p-Fig(ll), où la résultante est différente selon l'ordre choisi, et p-Fig(lr)$\oplus$p-Fig(r),



Les trois premiers collages sont interdits : Dans l'exemple 1, la figure p_1 n'est pas connexe, en 2 la figure résultante n'est pas connexe, et en 3 il y a chevauchement. Le collage de l'exemple 4 est autorisé.

FIG. 2.1 - *Collages interdits et collage autorisé.*

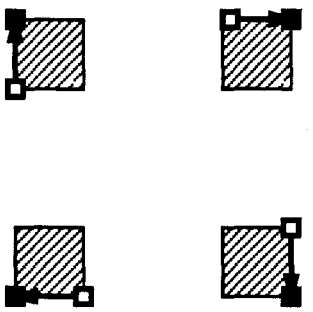


FIG. 2.2 - *Les carrés unitaires.*

où le collage n'est défini que dans un sens. La non associativité est montrée par : p-Fig(u) $\oplus$ (p-Fig(l) $\oplus$ p-Fig(r)) qui est un collage valide, alors que (p-Fig(u) $\oplus$ p-Fig(l)) $\oplus$ p-Fig(r) est interdit car la première étape ne donne pas un dessin connexe.

On étend la notion de collage aux ensembles de polyominos pointés : soient $\mathcal{G}$ et $\mathcal{G}'$, deux ensembles de polyominos pointés, alors $\mathcal{G} \oplus \mathcal{G}' = \{g \oplus g' \mid g \in \mathcal{G}, g' \in \mathcal{G}'$ et $g \oplus g'$ définie $\}$.

Soit $\mathcal{G}$ un ensemble de polyominos pointés. On définit le *collage itéré* de $\mathcal{G}$, noté $\mathcal{G}^{\oplus}$, comme le plus petit ensemble de polyominos pointés contenant $\mathcal{G}$ et clos par collage.

2.2 Collages de polyominos optimaux

Nous nous proposons d'étudier certaines caractéristiques du collage de polyominos pointés (par souci d'allégement, nous ne préciserons plus qu'il s'agit de figures pointées). Comme le collage est défini sans recouvrement, nous nous limitons à l'étude des polyominos dont la complexité descriptive est *optimale*, c'est à dire qui peuvent être décrits par un mot optimal, où chaque lettre allume un pixel différent des autres lettres.

Nous classons les polyominos en quatre familles :

Définition 2.2.1 *Nous notons $\mathcal{PO}$ l'ensemble des polyominos pointés optimaux, $\mathcal{POPRO}$ l'ensemble des polyominos pointés optimaux prolongeables, $\mathcal{POST}$ l'ensemble des polyominos pointés optimaux sans trous, $\mathcal{POSTPRO}$ l'ensemble des polyominos pointés optimaux sans trous prolongeables.*

Définition 2.2.2 *On note $\mathcal{U} = \{p\text{-}Fig(a) \mid a \in \Pi_p\}$, l'ensemble des polyominos optimaux unitaires, que nous appellerons par commodité* carrés unitaires.

Ces carrés unitaires sont illustrés en Fig. 2.2. Que pouvons-nous dire du collage de ces briques unitaires ?

Assertion 2.2.3 *Soit f, une figure obtenue par collage itéré des carrés unitaires, alors f est un polyomino optimal.*

Preuve. Par récurrence. La 4-connexité dérive immédiatement de la définition du collage, reste à vérifier l'optimalité. Si f est de taille égale à 1, l'assertion est vraie. Supposons qu'elle soit aussi vérifiée pour toutes les figures obtenues de taille inférieure ou égale à un entier n. Soit f de taille $n+1$, alors on a $f = g \oplus g' = g.g' = \text{p-Fig}(\mu.\mu')$ si $g = \text{p-Fig}(\mu)$ et $g' = \text{p-Fig}(\mu')$, les mots μ et μ' étant optimaux. Par définition des conditions nécessaires au collage, le mot $\mu\mu'$ ne colorie qu'une fois chaque pixel et est donc optimal. □

Assertion 2.2.4 *Si f est une figure de taille au moins 2, obtenue par collage itéré des carrés unitaires, alors il existe α, β deux mots optimaux non vides, tels que $f = p\text{-}Fig(\alpha) \oplus p\text{-}Fig(\beta)$ et $p\text{-}Fig(\alpha)$, $p\text{-}Fig(\beta)$ sont des polyominos.*

Preuve. On examine la dernière étape du collage itéré : soient f et g les deux figures collées à cette étape. Les figures f et g sont obtenues par collage itéré des carrés unitaires en au moins une étape, et sont donc des polyominos optimaux non vides, d'après l'assertion 2.2.3. □

Une question assez naturelle est la suivante :

> Pouvons-nous obtenir tous les polyominos optimaux par collage itéré des carrés unitaires ?

La réponse à notre question est non :

Assertion 2.2.5

$$\mathcal{PO} \not\subset \mathcal{U}^{\oplus}$$

Preuve. La figure 2.3, associé au mot *druullddruud*, est un contre-exemple. On énumère facilement la totalité des mots optimaux qui décrivent le contre-exemple : il y en a 12, et par symétrie seuls trois cas sont différents. On vérifie alors qu'aucun d'entre eux ne satisfait l'assertion 2.2.4. □

Cependant, notons tout de même que si l'on s'autorisait à changer les points de départ et d'arrivée, il deviendrait alors possible d'obtenir la figure par collage. On peut remarquer que le contre-exemple précédent est, en un certain sens, un cas limite, car il n'est pas prolongeable : il ne peut donc pas servir lui-même à un collage. Cela nous amène à reformuler notre question de la manière suivante :

> Pouvons-nous obtenir tous les polyominos optimaux prolongeables, par collage itéré des carrés unitaires ?

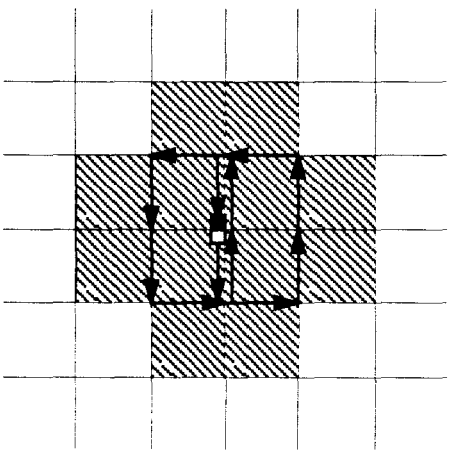


FIG. 2.3 - *Contre exemple pour* $= \mathcal{PO} \subset \mathcal{U}^{\oplus}$.

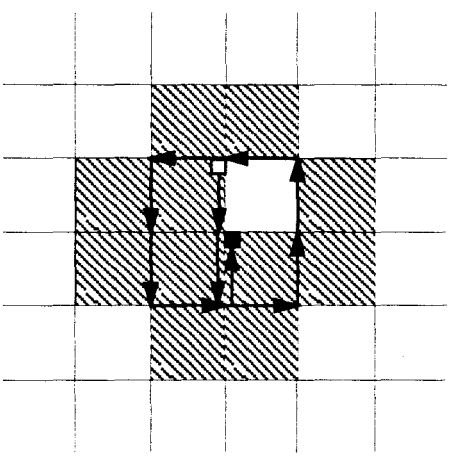


FIG. 2.4 - *Contre exemple pour* $\mathcal{U}^{\oplus} = \mathcal{POPRO}$.

Là encore, la réponse est non :

Assertion 2.2.6

$$\mathcal{POPRO} \not\subset \mathcal{U}^{\oplus}$$

Preuve. On montre que la figure 2.4, associée au mot $\omega = d^2ru^2l^2d^2ru$, est un contre-exemple, par la même méthode que précédemment. □

L'étude tend à montrer que la présence de trous est nécessaire dans la construction d'un contre-exemple de ce type. Notre question devient alors :

> Pouvons-nous obtenir tous les polyominos optimaux sans trous prolongeables, par collage itéré des carrés unitaires ?

Nous répondons alors par l'affirmative :

Théorème 2.2.7 *Tous les polyominos optimaux sans trous prolongeables s'obtiennent par collage itéré des carrés unitaires :*

$$\mathcal{POSTPRO} \subset \mathcal{U}^{\oplus}$$

La fin de ce chapitre est consacrée à cette preuve. Nous mettons d'abord en place certains résultats élémentaires, lesquels interviendront dans la démonstration du lemme principal 2.2.13.

2.2.1 Symétrie et figures

Nous définissons une symétrie sur les dessins pointés, et nous construisons, à partir d'un mot à pixels, le mot qui dessine le symétrique, à une translation près.

Proposition 2.2.8 *Soient les symétries ϕ_1 sur les entiers, ϕ_2 sur les dessins, Φ sur les dessins pointés et le morphisme ρ sur les mots définis par :*

$$- \begin{cases} \phi_1 : & \mathbb{Z}^2 & \to & \mathbb{Z}^2 \\ & (-x, y) & \mapsto & (x, y) \end{cases}$$

$$- \begin{cases} \phi_2 : & \mathcal{D} & \to & \mathcal{D} \\ & e & \mapsto & \{Pix(-x, y) \mid Pix(x, y) \in e\} \end{cases}$$

$$- \begin{cases} \Phi : & \mathcal{D}' & \to & \mathcal{D}' \\ & (e, d, a) & \mapsto & (\phi_2(e), \phi_1(a), \phi_1(d)) \end{cases}$$

$$- \begin{cases} \rho : & \Pi_p & \to & \Pi_p \\ & u & \mapsto & d \\ & r & \mapsto & r \\ & d & \mapsto & u \\ & l & \mapsto & l \end{cases}$$

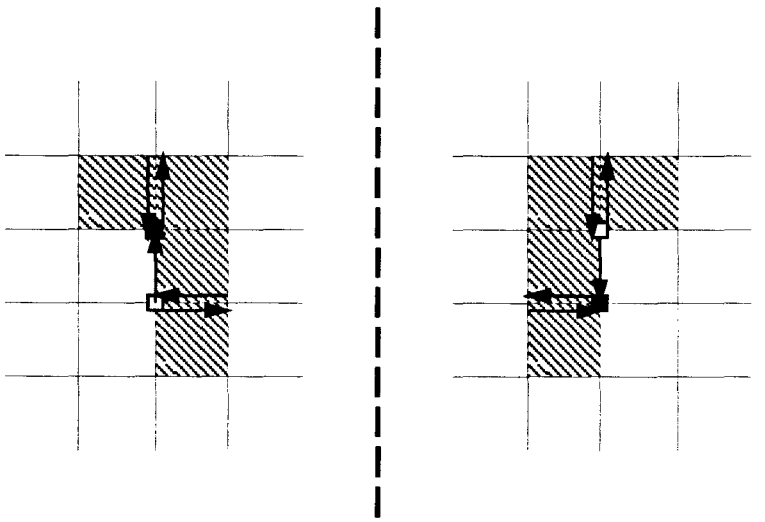


FIG. 2.5 - *P-figure associée à rluud et symétrique.*

Soit μ un mot à pixels, alors :

$$[\Phi(Desp(\mu))]_{\top} = [Desp(\rho(\underline{\mu}))]_{\top}$$

Un exemple de symétrie est illustré en Fig. 2.5.

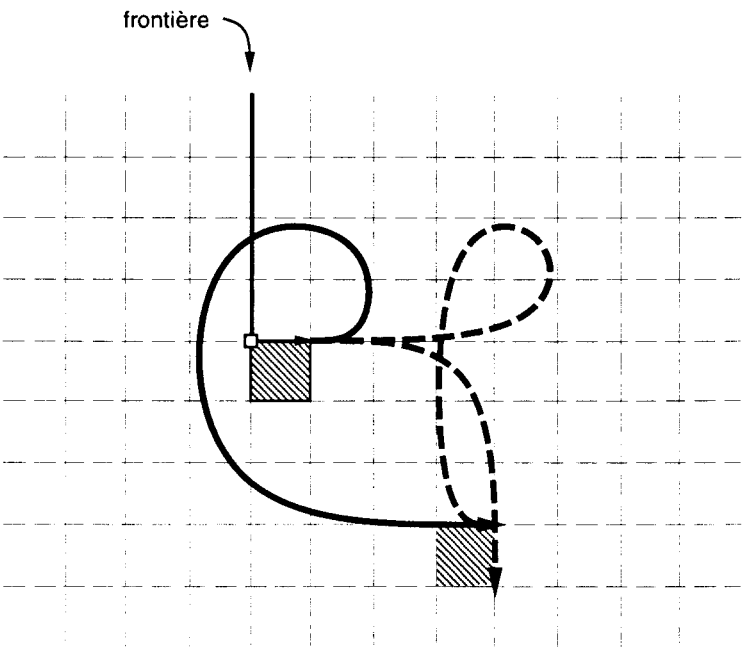
Preuve. Par récurrence sur la longueur de μ. Il est clair que la proposition est vraie pour les mots de longueur 1. On la suppose vérifiée pour tous les mots de longueur inférieure ou égale à k. Soit $\mu = \omega\chi$, avec $|\omega| = k$, $\chi \in \Pi_p$, et les dessins pointés $\mathrm{Desp}(\mu) = (f_\mu, O, a_\mu)$, $\mathrm{Desp}(\omega) = (f_\omega, O, a_\omega)$, $\mathrm{Desp}(\chi) = (f_\chi, O, a_\chi)$, avec O l'origine du plan.

$$\begin{aligned}
[\mathrm{Desp}(\rho(\underline{\mu}))]_{\top} &= [\mathrm{Desp}(\rho(\chi\underline{\omega}))]_{\top} \\
&= [\mathrm{Desp}(\rho(\chi))]_{\top}.[\mathrm{Desp}(\rho(\underline{\omega}))]_{\top} \\
&\overset{\text{rec}}{=} [(\phi_2(e_\chi), \phi_1(a_\chi), O)]_{\top}.[(\phi_2(e_\omega), \phi_1(a_\omega), O)]_{\top} \\
&= [t'_{\phi_1(a_\omega)}(\phi_2(e_\chi), \phi_1(a_\chi), O).(\phi_2(e_\omega), \phi_1(a_\omega), O)]_{\top} \\
&= [(t_{\phi_1(a_\omega)}(\phi_2(e_\chi)), \phi_1(a_\chi + a_\omega), \phi_1(a_\omega)).(\phi_2(e_\omega), \phi_1(a_\omega), O)]_{\top} \\
&= [(t_{\phi_1(a_\omega)}(\phi_2(e_\chi)) \cup \phi_2(e_\mu), \phi_1(a_\mu), O)]_{\top} \\
&= [(\phi_2(e_\mu), \phi_1(a_\mu), O)]_{\top} \\
&= [\Phi((e_\mu, O, a_\mu))]_{\top} \\
&= [\Phi(\mathrm{Desp}(\mu))]_{\top}
\end{aligned}$$

□

2.2.2 Lemme du quadrant

Le lemme du quadrant caractérise certaines propriétés des mots optimaux. De manière informelle, le lemme énonce le fait suivant : lorsqu'un mot optimal commence par une lettre $x_i, i \in \mathbb{Z}/4\mathbb{Z}$, alors on ne peut sortir du quadrant (quart



Exemple de tracés interdits (tirets épais) et autorisé (trait plein épais).

FIG. 2.6 - *Illustration du lemme du quadrant.*

du plan $\mathbb{Z}^2$) délimité par x_i et x_{i-1} que par la frontière pointée par x_{i-1}, sinon le mot ne serait pas optimal (Fig. 2.6). La deuxième proposition du lemme précise que dans un dessin entièrement tracé dans ce même quadrant, quelque soit le pixel choisi alors le pixel immédiatement à gauche de celui obtenu en projetant cette position sur l'axe pointé par x_i est allumé (Fig. 2.7).

Lemme 2.2.9 (du quadrant) *Soit $x_i \in \Pi_p$ et $\omega = x_i\nu$ un mot optimal vérifiant :*

$$\forall \omega' \in FacG(\nu), |\omega'|_{x_i} \geq |\omega'|_{x_{i-2}}$$

Alors :

Prop. 1 : $|\omega|_{x_{i-1}} \geq |\omega|_{x_{i+1}}$

Prop. 2 : $\forall k \in [0, sup\{|\mu|_{x_i} - |\mu|_{x_{i-2}} \mid \mu \in FacG(\nu)\}]$, *il existe* $\omega_1 x_i \in FacG(\omega)$ *avec* $|\omega_1|_{x_{i+1}} = |\omega_1|_{x_{i-1}}$ *et* $|\omega_1|_{x_i} - |\omega_1|_{x_{i-2}} = k$.

Preuve. Par récurrence sur la longueur de ω.

Soit $\omega = x_i$, alors la prop. 1 est vérifiée car $|\nu|_{x_{i-1}} = |\nu|_{x_{i+1}} = 0$ et la prop. 2 est vérifiée aussi en prenant $\omega_1 = \varepsilon$. Supposons la prop. 1 et la prop. 2 vraies pour tout ω tel que $|\omega| \leq n$.

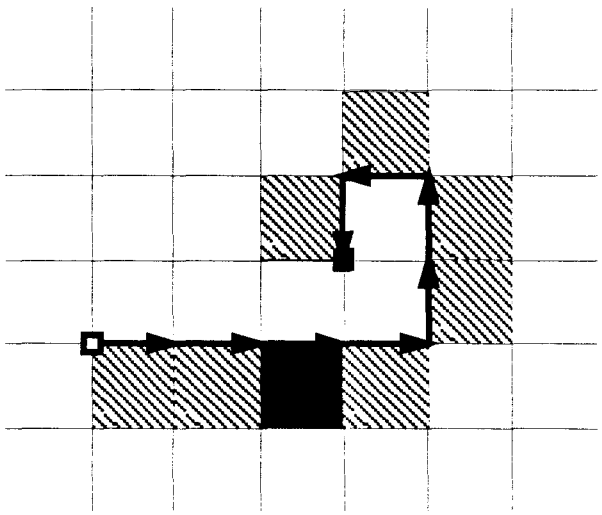


FIG. 2.7 - *En grisé, le pixel déjà allumé.*

Soit $\omega = x_i\nu' a$ vérifiant les prémisses du lemme, avec $|\nu'| = n-1$, $a \in \Pi_p$. D'après l'hypothèse de récurrence, la prop. 1 et la prop. 2 sont vérifiées pour le mot $x_i\nu'$.

Considérons chaque valeur possible pour a :

- Cas $a = x_{i+1}$. Par hypothèse de récurrence $|\nu'|_{x_{i-1}} \geq |\nu'|_{x_{i+1}}$. On ne peut pas avoir $|\nu'|_{x_{i-1}} = |\nu'|_{x_{i+1}}$ car alors (prop .2) $\omega_1 x_i$ allume le même pixel que $\omega_1 x_i \omega_2 a$ et ω ne serait pas optimal. Ce qui implique que $|\nu' a|_{x_{i-1}} \geq |\nu' a|_{x_{i+1}}$. La prop. 1 est donc vérifiée. Pour vérifier la prop. 2 il suffit de prendre les mêmes ω_1 que ceux qui existent pour $x_i\nu'$.

- Cas $a = x_{i-1}$. L'hypothèse implique $|\nu' a|_{x_{i-1}} \geq |\nu' a|_{x_{i+1}}$ et la prop. 1 est vérifiée. Pour vérifier la prop. 2 il suffit de prendre les mêmes ω_1 que ceux qui existent pour $x_i\nu'$.

- Cas $a = x_{i-2}$. Si $|\nu' a|_{x_i} < |\nu' a|_{x_{i-2}}$ alors la prémisse du lemme devient fausse. Sinon l'hypothèse implique $|\nu' a|_{x_{i-1}} \geq |\nu' a|_{x_{i+1}}$ et la prop. 1 est vérifiée. Pour vérifier la prop. 2 il suffit de prendre les mêmes ω_1 que ceux qui existent pour $x_i\nu'$.

- Cas $a = x_i$. L'hypothèse implique $|\nu' a|_{x_{i-1}} \geq |\nu' a|_{x_{i+1}}$ et la prop. 1 est vérifiée. Deux sous-cas sont à traiter pour la prop. 2 :

  - Si $|\nu'|_{x_{i-1}} = |\nu'|_{x_{i+1}}$ alors on prend $\omega_1 = x_i\nu'$.
  - Si $|\nu'|_{x_{i-1}} > |\nu'|_{x_{i+1}}$ alors $\exists \nu'_1, \nu'_2 \in \Pi_p^*$ tels que $\nu' a = \nu'_1 x_{i-1} \nu'_2$ avec ν'_1 le plus grand facteur gauche de ν' tel que $|\nu'_1|_{x_{i-1}} - |\nu'_1|_{x_{i+1}} = 0$. On a $|x_{i-1}\nu'_2 a| \leq n$, et $\forall \chi \in \mathrm{FacG}(\nu'_2 a)$, $|\chi|_{x_{i-1}} \geq |\chi|_{x_{i+1}}$, et on peut

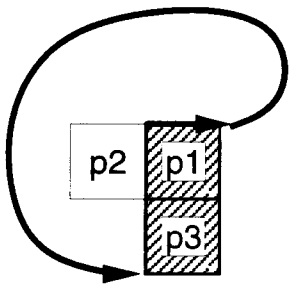
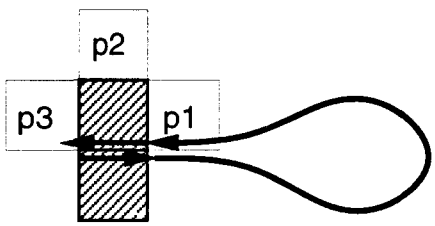


FIG. 2.8 - *Le pixel p_2 est dans un trou.*



Ce schéma illustre le cas $x_i^n = r$. La forme de la boucle correspondant au facteur α est fictive: la seule contrainte s'exerçant sur les pixels tracés par ce facteur résulte du fait qu'il est une boucle.

FIG. 2.9 - *Lemme de l'aller-retour.*

alors appliquer l'hypothèse de récurrence sur le mot $x_{i-1}\nu_2' a$ (on part ici avec x_{i-1}). Ce qui nous donne $|\nu_2' a|_{x_i} \leq |\nu_2' a|_{x_{i-2}}$. Donc il est clair qu'il existe $\zeta \in \mathrm{FacG}(\nu')$, tel que $|\zeta|_{x_i} - |\zeta|_{x_{i-2}} = |\omega|_{x_i} - |\omega|_{x_{i-2}}$. Par hypothèse de récurrence, pour ce facteur ζ il existe $\omega_1 x_i$ qui vérifie la prop. 2 et qui convient aussi. □

Dans la suite nous nous servirons fréquemment de ce lemme pour déduire qu'il existe un trou dans une figure donnée. Par exemple, soient un mot $\mu = \mu_1 r \mu_2 a \mu_3$, et les pixels $p_1 = \mathrm{Pix}(0,-1)$, $p_2 = \mathrm{Pix}(-1,-1)$ et $p_3 = \mathrm{Pix}(0,-2)$, tels que p_1 soit allumé dans $\mathrm{Des}(\mu)$ par la lettre r et p_3 soit allumé dans $\mathrm{Des}(\mu)$ par la lettre a (voir Fig. 2.8). Clairement si p_2 n'appartient pas à $\mathrm{Des}(\mu)$, alors il fait partie d'un trou de ce dessin.

2.2.3 Lemme de l'aller-retour

Nous énonçons ici une propriété partagée par les mots optimaux ayant une certaine forme (voir la Fig. 2.9 avec l'énoncé du lemme). Cette propriété nous sera utile lors de la démonstration du lemme du collage prolongeable.

Lemme 2.2.10 *Soit un mot optimal* $\omega = x_i^n \alpha x_{i-2}^n$ *avec* $\alpha \in \mathcal{B}$ *et* $n \geq 1$. *Soient* $e = Des(\omega)$ *le dessin associé à* ω, *et les pixels* $p_1 = Pix(1,0), p_2 = Pix(0,1), p_3 = Pix(-1,0)$, *alors :*

$$\{p_1, p_2, p_3\} \cap Ext_e \neq \emptyset \Rightarrow \alpha = x_i^m x_{i-2}^m, \text{ avec } m \in \mathbb{N}$$

Preuve. Pour simplifier, on admet, sans perte de généralité, que $x_i = r$, les autres cas étant traités de manière tout à fait symétrique. On procède par récurrence sur la taille de α. Les pixels p_1, p_2, p_3 sont appelés "pixels significatifs".

Le lemme est évidemment vrai si $|\alpha| = 0$. Comme α est une boucle sa longueur est paire. Supposons le lemme vrai pour tout $j \leq 2k$, $k \in \mathbb{N}$.

Soit α tel que $|\alpha| = 2k + 2$.

- Cas 1 : $\alpha = \gamma u$. On a $\omega = r^n \gamma u l^n$ et $\mathrm{Dec}(r^n\gamma) = (n, -1)$ (en effet on sera à l'origine après lecture de ω). Donc nous partons de l'origine pour atteindre $(n, -1)$ et le mot est optimal, aussi, d'après le lemme du quadrant, on entoure ou on trace les pixels significatifs : $\{p_1, p_2, p_3\} \cap \mathrm{Ext}_f = \emptyset$. Prémisses non vérifiées.

- Cas 2 : $\alpha = \gamma d$. On a $\omega = r^n \gamma d l^n$. Le mot n'est pas optimal : présence d'un facteur dl. Prémisses non vérifiées.

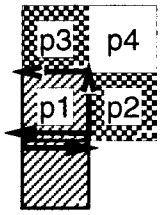
- Cas 3 : $\alpha = \gamma r$. On a $\omega = r^n \gamma r l^n$. Le mot n'est pas optimal : γr est une boucle qui finit par r et qui est située après un facteur finissant aussi par r. Prémisses non vérifiées.

- Cas 4 : $\alpha = \gamma l$. On examine alors par quelle lettre commence γ. Notons que nous sommes dans le cas $\{p_2, p_3\} \cap \mathrm{Ext}_f \neq \emptyset$.

  - Cas 4.1 : $\gamma = d\delta$. On a $\omega = r^n d\delta l l^n$. Le mot n'est pas optimal : présence d'un facteur rd. Prémisses non vérifiées.
  - Cas 4.2 : $\gamma = u\delta$. On a $\omega = r^n u\delta l l^n$. Le mot n'est pas optimal : $u\delta l$ est une boucle qui commence par u et finit par l. Prémisses non vérifiées.
  - Cas 4.3 : $\gamma = l\delta$. On a $\omega = r^n l\delta l l^n$. Le mot n'est pas optimal : $l\delta l$ est une boucle qui commence par l et qui est située devant un facteur commençant aussi par l. Prémisses non vérifiées.
  - Cas 4.4 : $\gamma = r\delta$. On a $\omega = r^{n+1}\delta l^{n+1}$. On peut alors appliquer l'hypothèse de récurrence sur δ. □

2.2.4 Lemme de la connexité

Nous examinons des propriétés caractéristiques liées à la connexité des polyominos pointés optimaux sans trous.



Le pixel p_1 est allumé par a, les pixels p_2 et p_3 sont allumés respectivement par x_i et x_{i-1}.

FIG. 2.10 - *Illustration du lemme de la connexité.*

Fait 2.2.11 *Soit un mot à pixel optimal $\mu =$ tel que $|\mu| \geq 2$ et $Des(\mu)$ n'est pas connexe. Alors il existe une factorisation $\mu = (\alpha x_i).(x_{i-1}\beta)$ avec $x_i, x_{i-1} \in \Pi_p$, $i \in \mathbb{Z}/4\mathbb{Z}$, telle que :*

$t_{c_1}(Des(x_i))$ et $t_{c_2}(Des(x_{i-1}))$, avec $c_1 = Dec(\alpha)$ et $c_2 = Dec(\alpha x_i)$, sont dans des composantes connexes différentes.

Preuve. Le mot μ est optimal et donc il ne peut contenir de facteur $x_i x_{i+1}$, et un mot qui ne contient que des facteurs $x_i x_i$ et $x_i x_{i-2}$ est clairement connexe. □

Nous appellerons *coupure* une telle factorisation.

Lemme 2.2.12 *Soit un mot à pixel $\mu = \alpha\beta$ tel que p-Fig(μ) soit un polyomino pointé optimal sans trous, $\alpha, \beta \in \Pi_p^*$, $|\alpha| \geq 1$, $|\beta| \geq 2$ et $Des(\beta)$ n'est pas connexe. Alors quelque soit la coupure $\beta = (\beta_1 x_i).(x_{i-1}\beta_2)$, $i \in \mathbb{Z}/4\mathbb{Z}$, il existe une factorisation $\alpha = \alpha_1 a \alpha_2$ avec $a \in \Pi_p$, telle que :*

$$t_{c_3}(Des(a)) = t_{c_4}(Des(x_{i-2})), \text{ avec } c_3 = Dec(\alpha_1) \text{ et } c_4 = Dec(\alpha\beta_1 x_i).$$

Nous donnons d'abord une explication intuitive du lemme précédent. Nous parlons, par abus de langage, du pixel allumé par une lettre. Le lemme affirme que si le dessin associé à β n'est pas connexe, alors quelque soit le facteur $x_i x_{i-1}$ de β où chacun des pixels associés à ces deux lettres appartient à une composante connexe différente, alors il y a un pixel allumé par une lettre de α qui est connexe avec chacun d'eux. Le lemme précise la position de ce pixel allumé dans α : c'est celui qui serait allumé si l'on remplaçait la deuxième lettre de ce facteur $x_i x_{i-1}$ par x_{i-2}. Une situation de ce type est présentée en Fig. 2.10.
Preuve. Nous notons $c_1 = \mathrm{Dec}(\alpha\beta_1)$, $c_2 = \mathrm{Dec}(\alpha\beta_1 x_i)$. Nous pouvons alors poser $p_1 = t_{c_2}(\mathrm{Des}(x_{i-2}))$, $p_2 = t_{c_1}(\mathrm{Des}(x_i))$, $p_3 = t_{c_2}(\mathrm{Des}(x_{i-1}))$ et $p_4 = t_{c_2}(\mathrm{Des}(x_i))$ (voir sur le schéma 2.10). Notons tout de suite que p_2 et p_3 appartiennent à $\mathrm{Des}(\mu)$.

- Cas 1 : $p_1 \in \mathrm{Des}(\mu)$. Si p_1 est allumé par une lettre de β, alors le facteur $x_i x_{i-1}$ ne vérifie pas la prop. 2.2.11, et les prémisses du lemme ne sont pas vérifiées. Donc p_1 est allumé par une lettre de α et le lemme est vérifié.

- Cas 2 : $p_4 \in \mathrm{Des}(\mu)$ et $p_1 \notin \mathrm{Des}(\mu)$. Nous raisonnons par l'absurde. Par une déduction semblable à celle du cas précédent, on déduit que p_4 est allumé par une lettre de α, que nous notons b. On a donc $\mu = \alpha'_1 b \alpha'_2 \beta_1 x_i x_{i-1} \beta_2$. Si p_4 est allumé par une lettre de l'ensemble $\{x_i, x_{i+1}, x_{i+3}\}$, alors, d'après le lemme du quadrant 2.2.9, le facteur $b\alpha'_2\beta_1 x_i$ entoure le pixel p_1 qui est alors dans un trou de $\mathrm{Des}(\mu)$, et les prémisses du lemme ne sont pas vérifiées. Si p_4 est allumé par la lettre x_{i-1} alors le facteur $\alpha'_2\beta_1$ commence par la lettre x_{i-2}, car μ est optimal, et donc p_1 appartient à $\mathrm{Des}(\mu)$. Contradiction.

- Cas 3 : $\{p_1, p_4\} \cap \mathrm{Des}(\mu) = \emptyset$. Alors soit p_1 ou p_4 appartiennent à des trous de $\mathrm{Des}(\mu)$, et les prémisses du lemme ne sont pas vérifiées. Soit p_1 et p_4 appartiennent à l'extérieur de $\mathrm{Des}(\mu)$, lequel n'est clairement pas connexe, et les prémisses du lemme ne sont pas vérifiées.□

2.2.5 Lemme du collage prolongeable

Le lemme suivant énonce que toute figure pointée dans $\mathcal{POSTPRO}$ de taille supérieure à 1 peut être obtenue par le collage de deux figures pointées dans $\mathcal{POSTPRO}$. La démonstration du lemme donne une méthode effective pour obtenir deux figures adéquates (il peut bien sûr exister plusieurs couples adéquats).

Lemme 2.2.13 (du collage prolongeable) *Soit un mot ω avec p-Fig$(\omega) \in \mathcal{POSTPRO}$ et $|\omega| \geq 2$, alors $\exists \omega_1, \omega_2$ avec p-Fig(ω_1), p-Fig$(\omega_2) \in \mathcal{POSTPRO}$ et $|\omega_1| \geq 1$, $|\omega_2| \geq 1$ tels que :*

$$\omega = \omega_1\omega_2 \text{ et } \textit{p-Fig}(\omega) = \textit{p-Fig}(\omega_1) \oplus \textit{p-Fig}(\omega_2)$$

Preuve. Nous ne travaillons pas directement sur les figures pointées, mais sur le représentant standard obtenu par Desp. Tous les dessins sont pointés, même lorsque nous ne le précisons pas. Nous parlons, par abus de langage, du pixel allumé par une lettre.

Nous admettons, sans perte de généralité, que :

- le mot optimal ω commence par la lettre r, les autres cas étant traités de manière similaire ;

- le point de départ de $\mathrm{Desp}(\omega)$ est à l'extérieur de la figure. En effet, $\mathrm{Desp}(\omega)$ est prolongeable, donc l'un au moins des points de départ ou d'arrivée se trouve à l'extérieur. Si c'est le point d'arrivée, on utilise la symétrie Φ présentée plus haut (sous-section 2.2.1, et on pose $\omega' = \rho(\underline{\omega})$. Alors $\mathrm{Desp}(\omega')$ est une figure avec point de départ à l'extérieur. Le découpage $\omega' = \omega'_1\omega'_2$ nous donnera donc, en appliquant à nouveau la symétrie, le découpage $\omega = \rho(\underline{\omega'}) = \rho(\underline{\omega'_2})\rho(\underline{\omega'_1})$.

Le principe de la preuve consiste à classer le dessin dans une parmi cinq catégories. Ces catégories, qui servent de "patron" pour le découpage, sont identifiées par l'appartenance ou pas, au dessin, de certains pixels précis. On note $f = \mathrm{Desp}(\omega)$, $f_1 = \mathrm{Desp}(\omega_1)$ et $f_2 = t'_{\mathrm{Dec}(\omega_1)}(\mathrm{Desp}(\omega_2))$. En coupant en deux ω, qui est optimal et dessine un polyomino, on sait que $f_1 \cap f_2 = \emptyset$ et que le tout est connexe : deux des conditions nécessaires au collage sont toujours respectées. Il est évident aussi que f_1 et f_2 sont optimaux et prolongeables. Il faudra donc montrer dans chaque cas que f_1 et f_2 sont des polyominos (condition de connexité), et sont sans trous. Comme f est connexe et sans trous, il suffit de montrer que f_1 ne peut assurer la connexité entre d'éventuelles composantes connexes de f_2, ni ne peut être un trou pour f_2, et, réciproquement, que f_2 ne peut assurer la connexité entre d'éventuelles composantes connexes de f_1, ni ne peut être un trou pour f_1. En ce qui concerne la connexité, nous scrutons les facteurs $x_i x_{i-1}$, qui doivent respecter le lemme 2.2.12.

Notons $p_1 = \mathrm{Pix}(0,-1)$, $p_2 = \mathrm{Pix}(-1,-1)$, $p_3 = \mathrm{Pix}(0,-2)$, $p_4 = \mathrm{Pix}(1,0)$, $p_5 = \mathrm{Pix}(1,-1)$ et $p_6 = \mathrm{Pix}(0,0)$ (voir Fig. 2.11).

Cas 1 : $p_2 \notin f$. On a $\omega = r\alpha$, et on prend comme découpage $\omega_1 = r$, $\omega_2 = \alpha$. Comme f est sans trous, on a $p_2 \in \mathrm{Ext}_f$, et d'après le lemme du quadrant, on ne peut tracer p_3 sans que p_2 devienne un trou dans f, donc on a $p_3 \notin f$. Il est évident que f_1, constituée du seul pixel p_1, est un polyomino sans trous. Montrons que f_2 est sans trous et connexe :

- Le pixel p_1 ne peut être un trou pour f_2 car il est connexe avec p_2 qui appartient à l'extérieur de f donc à l'extérieur de f_2. Par conséquent f_2 est sans trous.

- Supposons que f_2 ne soit pas connexe, alors il est constitué de deux composantes connexes reliées entre elles par p_1. Appelons C_5 celle qui contient p_5 et C_6 celle qui contient p_6. On déduit que p_4 n'appartient pas à f_2 qui sinon serait connexe. D'après le lemme du quadrant 2.2.9 et comme le dessin est optimal, si l'on trace C_5 puis C_6, alors p_4 est un trou pour f. Si l'on trace C_6 puis C_5 alors p_2 est un trou pour f. Contradiction.

Autres cas : Pour les cas différents du précédent nous renommons les pixels en : $p_1 = \mathrm{Pix}(0,-1)$, $p_2 = \mathrm{Pix}(0,0)$, $p_3 = \mathrm{Pix}(-1,0)$, $p_4 = \mathrm{Pix}(-1,-1)$ (voir Fig. 2.12). Nous avons par hypothèse $p_4 \in f$ et le point de départ de f doit être à l'extérieur, donc l'un au moins des pixels p_2 et p_3 doit être à l'extérieur. D'après le lemme du quadrant, on déduit que pour allumer p_4 nous avons $\omega = r\alpha l d\beta$, avec p_2 allumé par la lettre l et p_4 allumé par la lettre d, sinon ni p_2, ni p_3 ne sont à l'extérieur. D'après le lemme de l'aller-retour 2.2.10, α s'écrit obligatoirement $\alpha = r^n l^n$, avec $n \in \mathbb{N}$. On appelle alors $p_5 = \mathrm{Pix}(-1,-2)$, $p_6 = \mathrm{Pix}(0,-2)$, $p_7 = \mathrm{Pix}(n,-2)$, $p_8 = \mathrm{Pix}(n,-1)$, $p_9 = \mathrm{Pix}(n,1)$ et $p_{10} = \mathrm{Pix}(n-1,1)$.

Dans chaque cas détaillé ci-dessous, p_3 appartient à l'extérieur de f et est connexe à f_1, lequel ne peut donc jamais être un trou pour f_2.

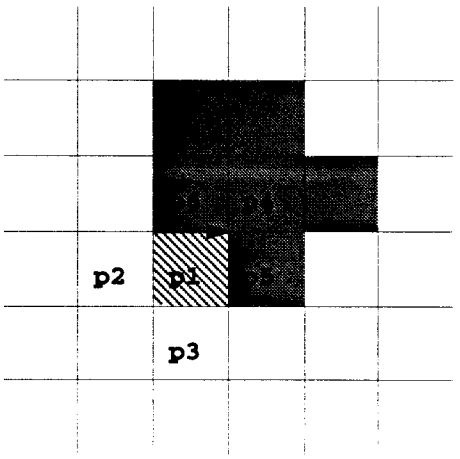


FIG. 2.11 - *Cas 1 : f_1 hachuré, f_2 en grisé.*

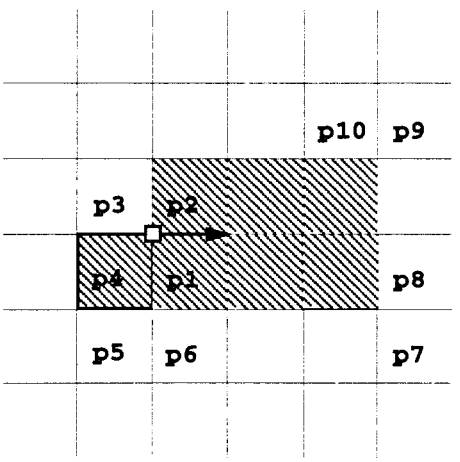


FIG. 2.12 - *Autres cas : numérotation des pixels.*

Cas 2 : $p_{10} \in f \wedge p_9 \notin f$. D'après le lemme du quadrant, pour que p_9 ne soit pas un trou, il faut nécessairement $\omega = \alpha u l \beta$, avec p_{10} tracé par la lettre l. On prend le découpage $\omega_1 = \alpha u$, $\omega_2 = l\beta$. Le dessin f_2 est connexe avec p_9, lequel appartient à Ext_f donc f_2 n'est pas un trou pour f_1. D'après le lemme du quadrant, il n'y a pas d'endroit où un facteur $x_i x_{i-1}$ dans ω_2 puissent être rendu connexe par un pixel de f_1, en laissant le point de départ de f à l'extérieur ni créer de trou, et réciproquement pour ω_1 et f_2.

Cas 3 : $p_8 \in f \wedge p_7 \notin f$, $\neg(p_{10} \in f \wedge p_9 \notin f)$. Pour que p_7 ne soit pas un trou, il faut nécessairement $\omega = \alpha r u \beta$, avec p_8 allumé par la lettre u. On prend le découpage $\omega_1 = \alpha r$, $\omega_2 = u\beta$. Le dessin f_2 est connexe avec p_7 qui appartient à Ext_f donc f_2 n'est pas un trou pour f_1. D'après le lemme du quadrant, il n'y a pas d'endroit où un facteur $x_i x_{i-1}$ dans ω_2 puissent être rendu connexe par un pixel f_1, en laissant le point de départ de f à l'extérieur ni créer de trou, et réciproquement pour ω_1 et f_2.

Cas 4 : $p_6 \in f \wedge p_5 \notin f$, $\neg(p_{10} \in f \wedge p_9 \notin f)$, $\neg(p_8 \in f \wedge p_7 \notin f)$. Pour que p_5 ne soit pas un trou, il faut nécessairement $\omega = \alpha d r \beta$, avec p_6 allumé par la lettre r. On prend le découpage $\omega_1 = \alpha d$, $\omega_2 = r\beta$. Le dessin f_2 est connexe avec p_5 qui appartient à Ext_f donc f_2 n'est pas un trou pour f_1. D'après le lemme du quadrant, il n'y a pas d'endroit où un facteur $x_i x_{i-1}$ dans ω_2 puissent être rendu connexe par un pixel f_1, en laissant le point de départ de f à l'extérieur ni créer de trou, et réciproquement pour ω_1 et f_2.

Cas 5 : $\neg(p_{10} \in f \wedge p_9 \notin f)$, $\neg(p_8 \in f \wedge p_7 \notin f)$, $\neg(p_6 \in f \wedge p_5 \notin f)$. On a $\omega = r^n l^n d \alpha$, avec p_4 tracé par la lettre d. On prend comme découpage $\omega_1 = r^n l^n$ et $\omega_2 = d\alpha$. Clairement f_1 est connexe et sans trous. D'après le lemme du quadrant, il n'y a pas d'endroit où un facteur $x_i x_{i-1}$ dans ω_2 puissent être rendu connexe par un pixel f_1, en laissant le point de départ de f à l'extérieur ni créer de trou. □

2.2.6 Conclusion

La démonstration du théorème 2.2.7 peut être complétée :

Preuve. Par récurrence sur la taille des figures. Clairement tous les polyominos de $\mathcal{POSTPRO}$ de taille inférieure ou égale à 2 sont obtenus par collage des carrés unitaires. On suppose que c'est aussi le cas pour tous ceux de taille inférieure à n. Soit un polyomino de $\mathcal{POSTPRO}$ de taille n, décrit par un mot μ, alors, d'après le lemme du collage prolongeable 2.2.13, il existe un découpage μ_1, μ_2 tel que p-Fig(μ) = p-Fig(μ_1) $\oplus$ p-Fig(μ_2), avec p-Fig(μ_1), p-Fig(μ_2) $\in \mathcal{POSTPRO}$. Nous avons p-Fig(μ_1) et p-Fig(μ_2) de taille inférieure à n et donc, par hypothèse de récurrence, pouvant être obtenus par collage itéré des carrés unitaires. Par conséquent c'est aussi le cas de p-Fig(μ). □

Conclusion

Sujet théorique, les langages de figures et de mots de figures ont déjà fait l'objet de nombreux travaux visant à explorer et caractériser leurs propriétés. Nous nous sommes quant à nous intéressés à la recherche de mots minimaux et de bornes de complexité descriptive pour les figures décrites par des mots avec lettres invisibles. Nous avons proposé un schéma pour la construction de preuve d'indécidabilité en utilisant le problème de Post. Nous avons étudié deux opérations de collage de figures, en proposant pour la dernière une sémantique nouvelle. Ce système inédit de description de figures nous semble intéressant, en particulier par la possibilité de renouveler l'approche des problèmes liés aux polyominos et aux pavages. Ainsi on peut se demander ce qu'il en est du collage pour d'autres classes de polyominos, ou encore ce que deviennent certains résultats de décidabilité dans cette sémantique.

Bibliographie

[AB94] P. Aigrin and D. Beauquier. Polyomino tilings, cellular automata and codicity. In *Proc. 3rd Int. Workshop on Polyminoes and Tilings*, pages 193–207. IRIT (CNRS-INP-UPS), 1994.

[Aut87] J.-M. Autebert. *Langages Algébriques.* Masson, Paris, 1987.

[BD93] F. J. Brandenburg and J. Dassow. Efficient reductions of picture words. *Theoretical Informatics and Applications*, 27(4):49–56, 1993.

[Bea91] D. Beauquier. An undecidable problem about rationnal sets and contour words of polyominoes. *Information Processing Letters*, 37:257–263, 1991.

[Ber79] J. Berstel. *Transductions and Context-free Languages.* Teubner, Stuttgart, 1979.

[BLS92] D. Beauquier, M. Latteux, and K. Slowinski. A decidability result about convex polyominoes. In *Lecture Notes in Computer Science*, volume 583, pages 32–45, 1992.

[BM94] M. Bousquet-Mélou. Polyominoes and polygons. *Contemporary Mathematics*, 178:55–70, 1994.

[BN91a] D. Beauquier and M. Nivat. Tiling pictures of the plane with two bars, a horizontal and a vertical one. Technical Report LITP 91.67, LITP, 1991.

[BN91b] D. Beauquier and M. Nivat. *Topology and Category Theory in Computer Science*, chapter Tiling the plane with one tile, pages 291–334. Oxford Univ. Press, 1991.

[BN92] D. Beauquier and M. Nivat. On translating one polyomino to tile the plane. *Discrete and Computational Geometry*, 1992.

[Bra89] F. J. Brandenburg. On minimal picture words. Technical Report MIP 8905, Univ. Passau, 1989.

[Das88] J. Dassow. Convexity and simplicity of chain code picture languages. *Rostock. Math. Kolloq.*, 34:53–60, 1988.

[Das89] J. Dassow. Graph-theoretical properties and chain code picture languages. *J. Inf. Process. Cybern.*, EIK 25:423–433, 1989.

[Das91] J. Dassow. On the connectedness of pictures in chain code picture languages. *Theoretical Computer Science*, 81:289–294, 1991.

[Del92] M. Delest. Physique, combinatoire et polyominos. In *Report on the Workshop Polyominoes and Tilings.* École Normale Supérieure de Lyon, 1992.

[DH91] J. Dassow and J. Hromkovic. On synchronized lindenmayer picture languages. Technical report, Technische Universität Magdeburg, 1991.

[DH93] J. Dassow and F. Hinz. Decision problems and regular chain code picture languages. *Discrete Applied Mathematics*, 1993.

[Eil74] S. Eilenberg. *Automata, Languages and Machines.* Academic Press, New York, London, 1974.

[EJ73] J. Edmonds and E. L. Johnson. Matching, euler tours and the chinese postman. *Mathematical Programming*, 5:88–124, 1973.

[ENRR87] H. Ehrig, M. Nagl, G. Rozenberg, and A. Rosenfeld. Third international workshop on graph grammars and their applications to computer science. *L.N.C.S.*, 291, 1987.

[Fre61] H. Freeman. On the encoding of arbitrary geometric configuration. *Ire Transactions on Electronic Computers*, 10:260–268, 1961.

[Fu74] K. S. Fu. *Syntactic Methods in Pattern Recognition.* Academic Press, New York, 1974.

[GM84] M. Gondran and M. Minoux. *Graphs and Algorithms.* Wiley, Chichester, UK, 1984.

[Gol65] S. Golomb. *Polyominoes.* Scribner, New York, 1965.

[Gut89] R. Gutbrod. A transformation system for generating description languages of chain code pictures. *Theoretical Computer Science*, 68:239–252, 1989.

[Gut90] R. Gutbrod. *Bildinvariante Transformationen und Verzweigungssprachen kettenkodierter Bilder.* PhD thesis, Bayerischen Julius-Maximilians-Universität Würzburg, 1990.

[Gut91] R. Gutbrod. Branch picture languages. Technical report, Aachen University, 1991.

[Har78] M. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley, reading, Mass, 1978.

[Hin86] F. Hinz. Regular chain code picture language of nonlinear descriptionnal complexity. *Lecture Notes in Computer Science*, 233:414–421, 1986.

[Hin88] F. Hinz. Questions of decidability for context-free chain code picture languages. In *IMYCS'88*, pages 135–144, 1988.

[Hin89] F. Hinz. Classes of picture languages that cannot be distinguished in the chain code concept and deletion of redundant retreats. *Lecture Notes in Computer Science*, 349:132–143, 1989.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA, 1979.

[HW88] F. Hinz and E. Welzl. Regular chain code picture languages with invisible lines. Technical Report 252, Graz University of Technology, 1988.

[Kim90] C. Kim. Complexity and decidability for restricted classes of picture languages. *Theoretical Computer Science*, 73, 1990.

[KS87] C. Kim and I. H. Sudborough. The membership and equivalence problems for picture languages. *Theoretical Computer Science*, 52:177–191, 1987.

[KS92] C. Kim and I. H. Sudborough. On reversal-bounded picture languages. *Theoretical Computer Science*, 104:185–206, 1992.

[Lee90a] J. Van Leeuwen, editor. *Handbook of Theoretical Computer Science.* Elsevier Science Publishers B. W., 1990.

[Lee90b] J. Van Leeuwen. *Handbook of Theoretical Computer Science*, volume Vol. A "Algorithm and Complexity", chapter Graph Algorithms, pages 525–631. Elsevier Science Publishers B. W., 1990.

[Lin68] A. Lindenmayer. Mathematical models for cellular interactions in development. *Journal of Theoretical Biology*, 18:280–315, 1968.

[LRRS95] M. Latteux, B. Ratoandromanana, D. Robilliard, and D. Simplot. Mots de figures composées de pixels. Technical Report IT–95–269, L.I.F.L., Univ. Lille I, 1995.

[LT90] M. Latteux and P. Turakainen. On characterizations of recursively enumerable languages. *Acta informatica*, 28:179–186, 1990.

[MRW82] H. A. Maurer, G. Rozenberg, and E. Welzl. Using string languages to describe picture languages. *Information and Control*, 54:155–185, 1982.

[Pap77] C. H. Papadimitriou. The euclidean traveling salesman problem is np-complete. *Theoretical Computer Science*, 4:237–244, 1977.

[Pău90] G. Păun. Five remarks (and two open problems) about the double segments and the invisible segments in picture description languages. Technical report, Institute of Mathematics of Bucarest, 1990. manuscript.

[Pol91] Université Paris XII – Val de Marne. *Journées Polyominos et Pavages*, 1991.

[PRS92] G. Păun, D. Robilliard, and K. Slowinski. Connected pictures and minimal words with blank moves. Technical Report IT-92-239, L.I.F.L., Univ. Lille I, 1992.

[Pru86] P. Prusinkiewicz. Graphical application of l-system. In *Proc. of Graphics Interface — Vision Interface*, pages 247–253, 1986.

[PS88] H.-O. Peitgen and D. Saupe, editors. *The Science of Fractal Images*. Springer-Verlag, 1988.

[PS89] V. Privman and N. M. Svrakic. Directed models of polymers, interfaces, and clusters : scaling and finite-size properties. *Physical Review Letters*, 60(123):1107–1109, 1989.

[Rem92] E. Remila. Un algorithme de pavage des figures sans pont par les pavés h_m et v_n. In *Report on the Workshop Polyominoes and Tilings*. École Normale Supérieure de Lyon, 1992.

[Rob94] D. Robilliard. Questions indécidables dans les langages de figures. Technical Report IT 262, L.I.F.L., Univ. Lille I, 1994.

[RR94] B. Ratoandromanana and D. Robilliard. Superposition in picture languages. *L.N.C.S.*, 787:322–334, 1994. Proceedings of CAAP'94.

[Sal81] A. Salomaa. *Jewels of Formal Language Theory*. Computer Science Press, 1981.

[Sch62] M. P. Schützenberger. Certain elementary families of automata. In *Proc. Symp. on Mathematical Theory of Automata*, pages 139–153. Polytechnic Institute of Brooklyn, 1962.

[Sim93] D. Simplot. Langages de figures 2d. Technical report, L.I.F.L., Univ. Lille I, 1993.

[Slo90] K. Slowinski. Improve the tracing of pictures with invisible lines. Technical Report IT 189, L.I.F.L., Univ. Lille I, 1990.

[Slo92] K. Slowinski. *Systèmes de Réécriture et Langages de Mots de Figure.* PhD thesis, Université des Sciences et Technologies de Lille, Flandres, Artois, France, 1992.

[Slo93] K. Slowinski. Picture words with invisible lines. *Theoretical Computer Science*, 108:357–363, 1993.

[SS90] P. Séébold and K. Slowinski. Minimizing picture words. In *LNCS*, volume 464, pages 234–243, 1990. Proc. IMYCS 1990.

[SS91] P. Séébold and K. Slowinski. The shortest way to draw a connected picture. *Computer Graphics Forum*, 10:319–327, 1991.

[SS94] P. Séébold and K. Slowinski. *Mathematical aspects of natural and formal languages*, chapter Redundant Retreat Free Word, pages 419–430. World Sci. Publ., Singapour, 1994.

[SW85] I. H. Sudborough and E. Welzl. Complexity and decidability for chain code picture languages. *Theoretical Computer Science*, 36:173–202, 1985.

[Vie92] X.G. Viennot. A survey of polyominoes enumeration. In *Actes du Colloque Séries Formelles et Combinatoire Algébrique*, Montréal, 1992.