USTL LIFL Laboratoire d'Informatique Fondamentale de Lille

Numéro d'ordre : 2008



# THÈSE

Nouveau Régime

Présentée à

L'UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE

Pour obtenir le titre de

DOCTEUR EN INFORMATIQUE

par

David SIMPLOT

# LANGAGES DE MOTS DE FIGURES, MONOÏDES INVERSIFS ET LANGAGES DE MOTS À DEUX DIMENSIONS

Thèse soutenue le 10 juin 1997, devant la Commission d'Examen :

| | | | |
|---|---|---|---|
| Président | : | Sophie TISON | Université de Lille 1 |
| Rapporteurs | : | Jean-Éric PIN | Université de Paris VII |
| | | Antonio RESTIVO | Universitá di Palermo |
| Examinateurs | : | Véronique BRUYÈRE | Université du Mons-Hainaut |
| | | Michel LATTEUX | Université de Lille 1 |
| | | Jacques MAZOYER | E.N.S. Lyon |

UNIVERSITE DES SCIENCES
ET TECHNOLOGIES DE LILLE

---

## DOYENS HONORAIRES DE L'ANCIENNE FACULTE DES SCIENCES

M. H. LEFEBVRE, M. PARREAU

## PROFESSEURS HONORAIRES DES ANCIENNES FACULTES DE DROIT ET SCIENCES ECONOMIQUES, DES SCIENCES ET DES LETTRES

MM. ARNOULT, BONTE, BROCHARD, CHAPPELON, CHAUDRON, CORDONNIER, DECUYPER, DEHEUVELS, DEHORS, DION, FAUVEL, FLEURY, GERMAIN, GLACET, GONTIER, KOURGANOFF, LAMOTTE, LASSERRE, LELONG, LHOMME, LIEBAERT, MARTINOT-LAGARDE, MAZET, MICHEL, PEREZ, ROIG, ROSEAU, ROUELLE, SCHILTZ, SAVARD, ZAMANSKI, Mes BEAUJEU, LELONG.

## PROFESSEUR EMERITE

M. A. LEBRUN

## ANCIENS PRESIDENTS DE L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

MM. M. PARREAU, J. LOMBARD, M. MIGEON, J. CORTOIS, A.DUBRULLE

## PRESIDENT DE L'UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

M. P. LOUIS

## PROFESSEURS - CLASSE EXCEPTIONNELLE

| | |
|---|---|
| M. CHAMLEY Hervé | Géotechnique |
| M. CONSTANT Eugène | Electronique |
| M. ESCAIG Bertrand | Physique du solide |
| M. FOURET René | Physique du solide |
| M. GABILLARD Robert | Electronique |
| M. LABLACHE COMBIER Alain | Chimie |
| M. LOMBARD Jacques | Sociologie |
| M. MACKE Bruno | Physique moléculaire et rayonnements atmosphériques |

| | |
|---|---|
| M. MIGEON Michel | EUDIL |
| M. MONTREUIL Jean | Biochimie |
| M. PARREAU Michel | Analyse |
| M. TRIDOT Gabriel | Chimie appliquée |

## PROFESSEURS - 1ère CLASSE

| | |
|---|---|
| M. BACCHUS Pierre | Astronomie |
| M. BIAYS Pierre | Géographie |
| M. BILLARD Jean | Physique du Solide |
| M. BOILLY Bénoni | Biologie |
| M. BONNELLE Jean Pierre | Chimie-Physique |
| M. BOSCQ Denis | Probabilités |
| M. BOUGHON Pierre | Algèbre |
| M. BOURIQUET Robert | Biologie Végétale |
| M. BRASSELET Jean Paul | Géométrie et topologie |
| M. BREZINSKI Claude | Analyse numérique |
| M. BRIDOUX Michel | Chimie Physique |
| M. BRUYELLE Pierre | Géographie |
| M. CARREZ Christian | Informatique |
| M. CELET Paul | Géologie générale |
| M. COEURE Gérard | Analyse |
| M. CORDONNIER Vincent | Informatique |
| M. CROSNIER Yves | Electronique |
| Mme DACHARRY Monique | Géographie |
| M. DAUCHET Max | Informatique |
| M. DEBOURSE Jean Pierre | Gestion des entreprises |
| M. DEBRABANT Pierre | Géologie appliquée |
| M. DECLERCQ Roger | Sciences de gestion |
| M. DEGAUQUE Pierre | Electronique |
| M. DESCHEPPER Joseph | Sciences de gestion |
| Mme DESSAUX Odile | Spectroscopie de la réactivité chimique |
| M. DHAINAUT André | Biologie animale |
| Mme DHAINAUT Nicole | Biologie animale |
| M. DJAFARI Rouhani | Physique |
| M. DORMARD Serge | Sciences Economiques |
| M. DOUKHAN Jean Claude | Physique du solide |
| M. DUBRULLE Alain | Spectroscopie hertzienne |
| M. DUPOUY Jean Paul | Biologie |
| M. DYMENT Arthur | Mécanique |
| M. FOCT Jacques Jacques | Métallurgie |
| M. FOUQUART Yves | Optique atmosphérique |
| M. FOURNET Bernard | Biochimie structurale |
| M. FRONTIER Serge | Ecologie numérique |
| M. GLORIEUX Pierre | Physique moléculaire et rayonnements atmosphériques |
| M. GOSSELIN Gabriel | Sociologie |
| M. GOUDMAND Pierre | Chimie-Physique |
| M. GRANELLE Jean Jacques | Sciences Economiques |
| M. GRUSON Laurent | Algèbre |
| M. GUILBAULT Pierre | Physiologie animale |
| M. GUILLAUME Jean | Microbiologie |
| M. HECTOR Joseph | Géométrie |
| M. HENRY Jean Pierre | Génie mécanique |
| M. HERMAN Maurice | Physique spatiale |
| M. LACOSTE Louis | Biologie Végétale |
| M. LANGRAND Claude | Probabilités et statistiques |

| | |
|---|---|
| M. LATTEUX Michel | Informatique |
| M. LAVEINE Jean Pierre | Paléontologie |
| Mme LECLERCQ Ginette | Catalyse |
| M. LEHMANN Daniel | Géométrie |
| Mme LENOBLE Jacqueline | Physique atomique et moléculaire |
| M. LEROY Jean Marie | Spectrochimie |
| M. LHENAFF René | Géographie |
| M. LHOMME Jean | Chimie organique biologique |
| M. LOUAGE Françis | Electronique |
| M. LOUCHEUX Claude | Chimie-Physique |
| M. LUCQUIN Michel | Chimie physique |
| M. MAILLET Pierre | Sciences Economiques |
| M. MAROUF Nadir | Sociologie |
| M. MICHEAU Pierre | Mécanique des fluides |
| M. PAQUET Jacques | Géologie générale |
| M. PASZKOWSKI Stéfan | Mathématiques |
| M. PETIT Francis | Chimie organique |
| M. PORCHET Maurice | Biologie animale |
| M. POUZET Pierre | Modélisation - calcul scientifique |
| M. POVY Lucien | Automatique |
| M. PROUVOST Jean | Minéralogie |
| M. RACZY Ladislas | Electronique |
| M. RAMAN Jean Pierre | Sciences de gestion |
| M. SALMER Georges | Electronique |
| M. SCHAMPS Joël | Spectroscopie moléculaire |
| Mme SCHWARZBACH Yvette | Géométrie |
| M. SEGUIER Guy | Electrotechnique |
| M. SIMON Michel | Sociologie |
| M. SLIWA Henri | Chimie organique |
| M. SOMME Jean | Géographie |
| Melle SPIK Geneviève | Biochimie |
| M. STANKIEWICZ François | Sciences Economiques |
| M. THIEBAULT François | Sciences de la Terre |
| M. THOMAS Jean Claude | Géométrie - Topologie |
| M. THUMERELLE Pierre | Démographie - Géographie humaine |
| M. TILLIEU Jacques | Physique théorique |
| M. TOULOTTE Jean Marc | Automatique |
| M. TREANTON Jean René | Sociologie du travail |
| M. TURRELL Georges | Spectrochimie infrarouge et raman |
| M. VANEECLOO Nicolas | Sciences Economiques |
| M. VAST Pierre | Chimie inorganique |
| M. VERBERT André | Biochimie |
| M. VERNET Philippe | Génétique |
| M. VIDAL Pierre | Automatique |
| M. WALLART Françis | Spectrochimie infrarouge et raman |
| M. WEINSTEIN Olivier | Analyse économique de la recherche et développement |
| M. ZEYTOUNIAN Radyadour | Mécanique |

## PROFESSEURS - 2ème CLASSE

| | |
|---|---|
| M. ABRAHAM Francis | |
| M. ALLAMANDO Etienne | Composants électroniques |
| M. ANDRIES Jean Claude | Biologie des organismes |
| M. ANTOINE Philippe | Analyse |
| M. BALL Steven | Génétique |
| M. BART André | Biologie animale |
| M. BASSERY Louis | Génie des procédés et réactions chimiques |
| Mme BATTIAU Yvonne | Géographie |
| M. BAUSIERE Robert | Systèmes électroniques |
| M. BEGUIN Paul | Mécanique |
| M. BELLET Jean | Physique atomique et moléculaire |
| M. BERNAGE Pascal | Physique atomique, moléculaire et du rayonnement |
| M. BERTHOUD Arnaud | Sciences Economiques |
| M. BERTRAND Hugues | Sciences Economiques |
| M. BERZIN Robert | Analyse |
| M. BISKUPSKI Gérard | Physique de l'état condensé et cristallographie |
| M. BKOUCHE Rudolphe | Algèbre |
| M. BODARD Marcel | Biologie végétale |
| M. BOHIN Jean Pierre | Biochimie métabolique et cellulaire |
| M. BOIS Pierre | Mécanique |
| M. BOISSIER Daniel | Génie civil |
| M. BOIVIN Jean Claude | Spectrochimie |
| M. BOUCHER Daniel | Physique |
| M. BOUQUELET Stéphane | Biologie appliquée aux enzymes |
| M. BOUQUIN Henri | Gestion |
| M. BROCARD Jacques | Chimie |
| Mme BROUSMICHE Claudine | Paléontologie |
| M. BUISINE Daniel | Mécanique |
| M. CAPURON Alfred | Biologie animale |
| M. CARRE François | Géographie humaine |
| M. CATTEAU Jean Pierre | Chimie organique |
| M. CAYATTE Jean Louis | Sciences Economiques |
| M. CHAPOTON Alain | Electronique |
| M. CHARET Pierre | Biochimie structurale |
| M. CHIVE Maurice | Composants électroniques optiques |
| M. COMYN Gérard | Informatique théorique |
| Mme CONSTANT Monique | Composants électroniques et optiques |
| M. COQUERY Jean Marie | Psychophysiologie |
| M. CORIAT Benjamin | Sciences Economiques |
| Mme CORSIN Paule | Paléontologie |
| M. CORTOIS Jean | Physique nucléaire et corpusculaire |
| M. COUTURIER Daniel | Chimie organique |
| M. CRAMPON Norbert | Tectolique géodynamique |
| M. CURGY Jean Jacques | Biologie |
| M. DANGOISSE Didier | Physique théorique |
| M. DE PARIS Jean Claude | Analyse |
| M. DECOSTER Didier | Composants électroniques et optiques |
| M. DEJAEGER Roger | Electrochimie et Cinétique |
| M. DELAHAYE Jean Paul | Informatique |
| M. DELORME Pierre | Physiologie animale |
| M. DELORME Robert | Sciences Economiques |
| M. DEMUNTER Paul | Sociologie |
| Mme DEMUYNCK Claire | Physique atomique, moléculaire et du rayonnement |
| M. DENEL Jacques | Informatique |
| M. DEPREZ Gilbert | Physique du solide - christallographie |

| | |
|---|---|
| M. DERIEUX Jean Claude | Microbiologie |
| M. DERYCKE Alain | Informatique |
| M. DESCAMPS Marc | Physique de l'état condensé et cristallographie |
| M. DEVRAINNE Pierre | Chimie minérale |
| M. DEWAILLY Jean Michel | Géographie humaine |
| M. DHAMELINCOURT Paul | Chimie physique |
| M. DI PERSIO Jean | Physique de l'état condensé et cristallographie |
| M. DUBAR Claude | Sociologie démographique |
| M. DUBOIS Henri | Spectroscopie hertzienne |
| M. DUBOIS Jean Jacques | Géographie |
| M. DUBUS Jean Paul | Spectrométrie des solides |
| M. DUPONT Christophe | Vie de la firme |
| M. DUTHOIT Bruno | Génie civil |
| Mme DUVAL Anne | Algèbre |
| Mme EVRARD Micheline | Génie des procédés et réactions chimiques |
| M. FAKIR Sabah | Algèbre |
| M. FARVACQUE Jean Louis | Physique de l'état condensé et cristallographie |
| M. FAUQUEMBERGUE Renaud | Composants électroniques |
| M. FELIX Yves | Mathématiques |
| M. FERRIERE Jacky | Tectonique - Géodynamique |
| M. FISCHER Jean Claude | Chimie organique, minérale et analytique |
| M. FONTAINE Hubert | Dynamique des cristaux |
| M. FORSE Michel | Sociologie |
| M. GADREY Jean | Sciences économiques |
| M. GAMBLIN André | Géographie urbaine, industrielle et démographie |
| M. GOBLOT Rémi | Algèbre |
| M. GOURIEROUX Christian | Probabilités et statistiques |
| M. GREGORY Pierre | I.A.E. |
| M. GREMY Jean Paul | Sociologie |
| M. GREVET Patrice | Sciences Economiques |
| M. GRIMBLOT Jean | Chimie organique |
| M. GUELTON Michel | Chimie physique |
| M. GUICHAOUA André | Sociologie |
| M. HAIMAN Georges | Modélisation,calcul scientifique, statistiques |
| M. HOUDART René | Physique atomique |
| M. HUEBSCHMANN Johannes | Mathématiques |
| M. HUTTNER Marc | Algèbre |
| M. ISAERT Noël | Physique de l'état condensé et cristallographie |
| M. JACOB Gérard | Informatique |
| M. JACOB Pierre | Probabilités et statistiques |
| M. JEAN Raymond | Biologie des populations végétales |
| M. JOFFRE Patrick | Vie de la firme |
| M. JOURNEL Gérard | Spectroscopie hertzienne |
| M. KOENIG Gérard | Sciences de gestion |
| M. KOSTRUBIEC Benjamin | Géographie |
| M. KREMBEL Jean | Biochimie |
| Mme KRIFA Hadjila | Sciences Economiques |
| M. LANGEVIN Michel | Algèbre |
| M. LASSALLE Bernard | Embryologie et biologie de la différenciation |
| M. LE MEHAUTE Alain | Modélisation,calcul scientifique,statistiques |
| M. LEBFEVRE Yannic | Physique atomique,moléculaire et du rayonnement |
| M. LECLERCQ Lucien | Chimie physique |
| M. LEFEBVRE Jacques | Physique |
| M. LEFEBVRE Marc | Composants électroniques et optiques |
| M. LEFEVRE Christian | Pétrologie |
| Melle LEGRAND Denise | Algèbre |
| M. LEGRAND Michel | Astronomie - Météorologie |
| M. LEGRAND Pierre | Chimie |
| Mme LEGRAND Solange | Algèbre |
| Mme LEHMANN Josiane | Analyse |
| M. LEMAIRE Jean | Spectroscopie hertzienne |

| | |
|---|---|
| M. LE MAROIS Henri | Vie de la firme |
| M. LEMOINE Yves | Biologie et physiologie végétales |
| M. LESCURE François | Algèbre |
| M. LESENNE Jacques | Systèmes électroniques |
| M. LOCQUENEUX Robert | Physique théorique |
| Mme LOPES Maria | Mathématiques |
| M. LOSFELD Joseph | Informatique |
| M. LOUAGE Francis | Electronique |
| M. MAHIEU François | Sciences économiques |
| M. MAHIEU Jean Marie | Optique - Physique atomique |
| M. MAIZIERES Christian | Automatique |
| M. MANSY Jean Louis | Géologie |
| M. MAURISSON Patrick | Sciences Economiques |
| M. MERIAUX Michel | EUDIL |
| M. MERLIN Jean Claude | Chimie |
| M. MESMACQUE Gérard | Génie mécanique |
| M. MESSELYN Jean | Physique atomique et moléculaire |
| M. MOCHE Raymond | Modélisation,calcul scientifique,statistiques |
| M. MONTEL Marc | Physique du solide |
| M. MORCELLET Michel | Chimie organique |
| M. MORE Marcel | Physique de l'état condensé et cristallographie |
| M. MORTREUX André | Chimie organique |
| Mme MOUNIER Yvonne | Physiologie des structures contractiles |
| M. NIAY Pierre | Physique atomique,moléculaire et du rayonnement |
| M. NICOLE Jacques | Spectrochimie |
| M. NOTELET Francis | Systèmes électroniques |
| M. PALAVIT Gérard | Génie chimique |
| M. PARSY Fernand | Mécanique |
| M. PECQUE Marcel | Chimie organique |
| M. PERROT Pierre | Chimie appliquée |
| M. PERTUZON Emile | Physiologie animale |
| M. PETIT Daniel | Biologie des populations et écosystèmes |
| M. PLIHON Dominique | Sciences Economiques |
| M. PONSOLLE Louis | Chimie physique |
| M. POSTAIRE Jack | Informatique industrielle |
| M. RAMBOUR Serge | Biologie |
| M. RENARD Jean Pierre | Géographie humaine |
| M. RENARD Philippe | Sciences de gestion |
| M. RICHARD Alain | Biologie animale |
| M. RIETSCH François | Physique des polymères |
| M. ROBINET Jean Claude | EUDIL |
| M. ROGALSKI Marc | Analyse |
| M. ROLLAND Paul | Composants électroniques et optiques |
| M. ROLLET Philippe | Sciences Economiques |
| Mme ROUSSEL Isabelle | Géographie physique |
| M. ROUSSIGNOL Michel | Modélisation,calcul scientifique,statistiques |
| M. ROY Jean Claude | Psychophysiologie |
| M. SALERNO Françis | Sciences de gestion |
| M. SANCHOLLE Michel | Biologie et physiologie végétales |
| Mme SANDIG Anna Margarette | |
| M. SAWERYSYN Jean Pierre | Chimie physique |
| M. STAROSWIECKI Marcel | Informatique |
| M. STEEN Jean Pierre | Informatique |
| Mme STELLMACHER Irène | Astronomie - Météorologie |
| M. STERBOUL François | Informatique |
| M. TAILLIEZ Roger | Génie alimentaire |
| M. TANRE Daniel | Géométrie - Topologie |
| M. THERY Pierre | Systèmes électroniques |
| Mme TJOTTA Jacqueline | Mathématiques |
| M. TOURSEL Bernard | Informatique |
| M. TREANTON Jean René | Sociologie du travail |

| | |
|---|---|
| M. TURREL Georges | Spectrochimie infrarouge et raman |
| M. VANDIJK Hendrik | |
| Mme VAN ISEGHEM Jeanine | Modélisation,calcul scientifique,statistiques |
| M. VANDORPE Bernard | Chimie minérale |
| M. VASSEUR Christian | Automatique |
| M. VASSEUR Jacques | Biologie |
| Mme VIANO Marie Claude | |
| M. WACRENIER Jean Marie | Electronique |
| M. WARTEL Michel | Chimie inorganique |
| M. WATERLOT Michel | géologie générale |
| M. WEICHERT Dieter | Génie mécanique |
| M. WERNER Georges | Informatique théorique |
| M. WIGNACOURT Jean Pierre | |
| M. WOZNIAK Michel | Spectrochimie |
| Mme ZINN JUSTIN Nicole | Algèbre |

*À Luc,*

*À mes parents.*

*Je remercie Sophie Tison de me faire l'honneur de présider le jury de cette thèse ; les cours qu'elle dispensait en D.E.A. ne sont pas étrangers à mon attrait pour l'informatique théorique.*

*Je suis très reconnaissant à Jean-Éric Pin et Antonio Restivo d'avoir accepté la tâche ingrate de rapporter mes travaux. Je suis très touché du soin avec lequel ils ont lu ce mémoire et de la diligence dont ils ont fait preuve afin que cette thèse puisse être soutenue.*

*Je tiens à exprimer toute ma gratitude à Véronique Bruyère et Jacques Mazoyer ; je suis très sensible à leur présence dans ce jury.*

*Je remercie tout particulièrement Michel Latteux qui, durant la préparation de cette thèse, a su me guider et m'aider constamment dans un domaine qui m'était jusqu'alors étranger. Je lui dois mon intérêt pour la théorie des langages formels — j'ai appris sous sa direction à ne pas me limiter à mon seul domaine de travail.*

*Mes remerciements s'adressent aussi à Boris Kokoszko sans qui cette thèse n'aurait pu être rédigée en d'aussi bonnes conditions, ainsi qu'à Isabelle Ryl pour son soutien qui m'a été précieux surtout pendant certaines périodes difficiles que j'ai connues. Je remercie également mon ami Denis Robilliard qui, par de nombreuses discussions, m'a énormément apporté aussi bien dans le domaine de la recherche que sur le plan personnel. Je remercie Marie-Suzanne Pailleux pour m'avoir supporté tout ce temps.*

*Je voudrais bien sûr remercier tous les membres du Laboratoire d'Informatique Fondamentale de Lille. Je pense tout particulièrement à Isabelle Biermann, Anne-Cécile Caron, Véronique Froidure, Catherine Herpin, Nathalie Revol, Denis Derencourt, Christian Lefebvre, Yves Roos et Julien Soula, ils ont tous une part de responsabilité dans la bonne ambiance dans laquelle j'ai pu travailler. Je remercie également Monsieur Wladislas Ryl qui a relu une bonne partie de ce mémoire abscons.*

# Table des matières

# Introduction

# Introduction

La synthèse d'images « virtuelles », utilisant des modeleurs 3D qui se servent des techniques de « lancer de rayon » ou encore de « radiosité », occupe le devant de la scène en matière de génération d'images par ordinateur — semblant[1] rendre obsolètes les travaux issus de la théorie des langages formels (algébriques ou syntaxiques pourrait-on dire) [Clo67, Sha69]. Le dual de la génération d'image est la reconnaissance de formes ou l'analyse d'images. Elle nécessite quant à elle le choix d'un formalisme adéquat aussi bien du point de vue mathématique que du point de vue algorithmique. Dans cette optique, les recherches s'appuient sur diverses théories : langages formels, géométrie discrète ou topologie [Kir64, KW67, Min71, Myl72, Ros79, Fu82, CM91].

L'objet de ce mémoire est d'étudier à l'aide de la théorie des langages formels deux formalismes permettant de décrire des ensembles de « figures » ou « images ». Nous traitons ces deux formalismes dans deux parties disjointes — les approches étant différentes vis-à-vis des langages formels. Dans le premier cas, nous tentons de décrire des figures, objets à deux dimensions, à l'aide de mots, objets unidimensionnels — un mot correspond à un parcours de la figure. C'est ce que nous appelons les langages de mots de figures ; nous essayons d'exhiber des propriétés sur les ensembles de figures à partir de ce que l'on connaît sur les langages de mots qui les décrivent. Le deuxième formalisme découle de la démarche inverse : les figures sont des mots à deux dimensions et nous étendons des notions bien connues dans les mots, comme la reconnaissabilité, au cas à deux dimensions.

## Mots de figures

Dans la première partie, nous nous proposons de décrire des figures à l'aide de mots. L'idée est d'assimiler le mot à une série d'ordres donnés à un traceur [Fre61, Fre62, Fed68, Fre74]. Cette méthode, utilisée à l'origine pour modéliser des courbes discrètes, utilise le « code de Freeman » qui comporte huit ordres élémentaires (voir Figure 1). Elle fût utilisée par H.A. Maurer et al. [MRW82, MRW83] dans une version simplifiée utilisant le « code de Freeman restreint » à quatre lettres (voir Figure 1) — néanmoins on sent bien que cette restriction n'est pas importante puisque les résultats restent pertinents que l'on utilise quatre ou huit lettres. Les auteurs de la « théorie des langages de figures » utilisent des langages sur l'alphabet $\Pi = \{u, r, d, l\}$ ($u$ pour up, $r$ pour right, $d$ pour down et $l$ pour

1. Ce n'est qu'apparence car des techniques « simples », comme les fractales, restent incontournables.

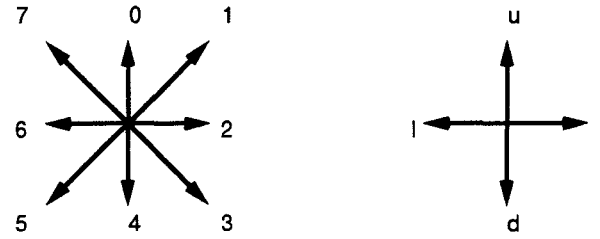



FIG. 1 - *Le code de Freeman et le code de Freeman restreint.*

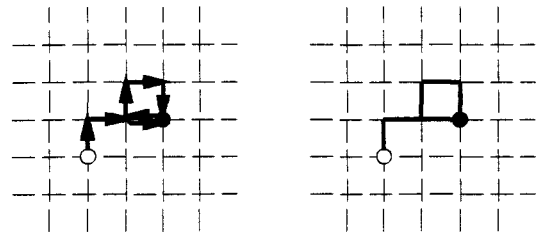



FIG. 2 - *Figure tracée par le mot ururdlr.*

left) pour décrire des figures composées d'un nombre fini de segments unitaires du plan cartésien $\mathbb{Z}^2$. Ainsi le mot *ururdlr* décrit la figure illustrée Figure 2 — notons qu'en plus des segments tracés nous retenons deux informations : le point de départ (désigné par un cercle blanc) qui correspond au point où se trouve le crayon avant de commencer la lecture du mot, et le point d'arrivée (désigné par un cercle noir) qui signale le point où se trouve le crayon après avoir lu le mot.

L'un des problèmes rencontrés dans ce formalisme est qu'il existe une infinité de mots décrivant une figure donnée. En effet, on ne compte pas le nombre de passages sur les segments, on ne retient que si le segment est tracé ou pas. Par exemple, tous les mots de la forme $(ud)^n ururdlr$ avec $n \in \mathbb{N}$ décrivent la figure illustrée précédemment. On aura donc à s'interroger sur le moyen de savoir si deux mots décrivent la même figure, ou comment trouver le plus petit mot qui décrit une figure donnée. Pour décrire un ensemble de figures, il suffit de considérer un langage de mots de $\Pi^*$. On peut se poser un certain nombre de problèmes d'algorithmique sur ces langages. Par exemple, pour quelles classes de langages (de la hiérarchie de Chomsky) peut-on décider de l'appartenance? Si l'on considère deux langages de mots rationnels sur $\Pi$, peut-on décider s'ils décrivent le même ensemble de figures?

La métaphore du traceur que nous avons utilisée nous montre clairement une limitation concernant les figures que l'on peut décrire. En effet, puisque le tracé se fait sans pouvoir lever le crayon, on ne peut tracer que des figures « connexes ». Il existe néanmoins des extensions à cette sémantique, la plus naturelle étant de permettre de bouger le crayon

sans tracer de segment. Ce sont les langages de figures avec traits invisibles [HW88] qui utilisent un alphabet $\Pi_i = \{u, r, d, l, u', r', d', l'\}$ où les lettres primées correspondent à des déplacements invisibles. Citons également les langages de figures avec branchements [Gut91a, RR94] où l'on considère que le traceur possède une pile où stocker la position courante du crayon afin de venir s'y replacer en dépilant.

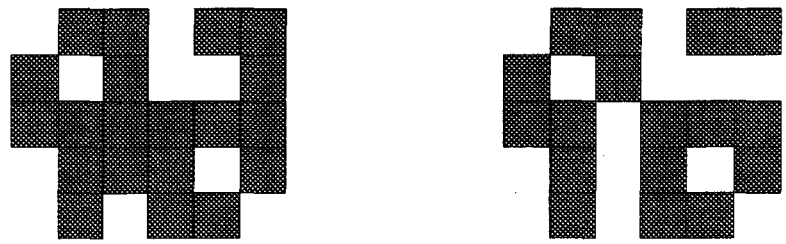
Dans une sémantique quelconque, nous disons que deux mots sont équivalents s'ils décrivent la même figure. Cette relation d'équivalence est en fait une congruence et l'on s'est intéressé aux règles de transformation qui permettent de réécrire un mot en un autre mot sans altérer la figure décrite. Cette base de la congruence est donnée sous la forme d'un système de réécriture dont la clôture réflexive et transitive est la congruence [Gut91b, SS91] (pour la sémantique « classique » sur $\Pi$) [Slo93] (pour la sémantique avec traits invisibles). Certaines des règles de ces systèmes de réécriture diminuent la longueur des mots. S'en inspirer permet d'optimiser la description des figures — *i.e.* trouver un mot le plus court possible qui décrit une figure — [BD89, Gut89, SS90, BD93, SS94]. Un résultat remarquable apparaît dans [SS90] où les auteurs donnent un algorithme polynomial permettant de trouver un mot minimal[2] à partir d'un mot décrivant la figure.

Certaines règles de réécriture qui diminuent la longueur de la description sont naturelles. Par exemple, les tracés redondants comme *rlr* ou *rudlru* peuvent être remplacés par *r* ou *ru*. Dans [SS94], P. Séébold et K. Slowinski montrent la confluence du système de réécriture constitué uniquement des règles supprimant ces redondances. Ceci amène naturellement à considérer des langages de mots afin de construire un langage de la même classe qui contient des descriptions plus courtes, par exemple des mots sans redondance [Hin86, Bra89a, Bra89b, Hin89].

Pour une figure décrite dans une sémantique donnée, la complexité descriptive désigne le rapport entre la longueur de ses mots minimaux et son nombre de segments. Cette complexité descriptive a pour but de mesurer la difficulté que l'on a à dessiner la figure dans la sémantique considérée. Ainsi, avec $\Pi$, on sait que pour une figure comprenant $n$ segments on peut toujours trouver un mot d'au plus $2.n$ lettres qui la décrit — on dit alors que la complexité descriptive est bornée par 2 [MRW82]. Pour la sémantique avec traits invisibles, il n'existe pas de telle borne puisque les segments peuvent être arbitrairement éloignés dans les figures non-connexes. Néanmoins, si l'on considère uniquement les figures connexes, mais en s'autorisant à lever le crayon, on peut diminuer de manière significative la longueur du tracé par rapport à la description sans lever de crayon mais on peut toujours atteindre la borne 2 [PRS93].

Parmi les problèmes d'algorithmique liés aux langages de figures, nous avons déjà cité le problème de l'appartenance. Il existe un algorithme qui prend en entrée un langage de mots algébrique et une figure et qui répond si oui ou non le langage contient une description de la figure. Le problème de l'appartenance est donc décidable pour les algébriques mais est un problème NP-Complet — même si l'on se restreint aux rationnels. Un autre problème naturel, celui de l'équivalence, *i.e.* savoir si deux langages de mots décrivent le même

2. Un mot est minimal s'il n'existe de mots strictement plus courts qui décrivent la même figure.



polyomino ensemble non 4-connexe

FIG. 3 - *Exemple et contre-exemple de polyomino*

ensemble de figures, est connu pour être indécidable même pour les langages rationnels. De même le problème de l'intersection, savoir s'il existe une figure décrite dans deux langages de mots, est indécidable pour les langages rationnels. On pourra trouver plus de précisions sur ces problèmes de décidabilité dans [MRW82, SW85, Hin88, KS87, Hin90, Kim90b, DH93, Kim96].

Ces résultats d'indécidabilité et l'aspect impraticable des résultats décidables ont poussé à considérer des classes restreintes de langages de figures où ces problèmes deviennent décidables ou polynomiaux. C'est le cas des langages-rubans où toutes les figures sont dessinées entre deux lignes parallèles fixées à l'avance ou encore des langages à trois directions où l'on s'interdit une des quatres lettres [SW85, Kim90a, Kim90b, KS92, Kim94, Hin].

Pour finir avec les problèmes de décidabilité, nous allons parler des propriétés existentielles dans les langages de figures, mais avant nous devons introduire des objets mathématiques que l'on peut décrire à l'aide des mots de figures : les polyominos. Un polyomino est une partie finie de $\mathbb{Z}^2$ 4-connexe [Gol66a] (voir Figure 3). Ces objets ont un intérêt tout particulier puisqu'ils sont utilisés en physique statistique pour modéliser des phénomènes de percolations ou des molécules de protéines. Le pavage de polyominos ou du plan par des polyominos est également un problème classique [Gol66b, Gol70, Rob71, Pen78, GS86, BN90, BN91].

Pour utiliser les polyominos dans des expériences de physique, il faut pouvoir les générer aléatoirement ce qui est proche de l'énumération — pour tout entier $n$ savoir combien il existe de polyominos possédant $n$ cellules. Ceci est un des plus importants problèmes non résolus, mais l'on a des résultats partiels qui concernent des classes particulières de polyominos. Les polygones sont des polyominos sans trous que l'on ne sait pas non plus énumérer. Un polyomino est dit horizontalement convexe (respectivement verticalement convexe) si chacune de ses lignes (resp. colonnes) est connexe. Un polyomino convexe est un polyomino qui est à la fois horizontalement et verticalement convexe (voir Figure 4). L'énumération de ces classes de polyominos a été résolue [Ell82, DV84, BM94].

Les polygones peuvent être décrits par des mots de figures, appelés mots de contour de polyominos. Ces mots sont des boucles où aucun facteur propre n'est une boucle (voir

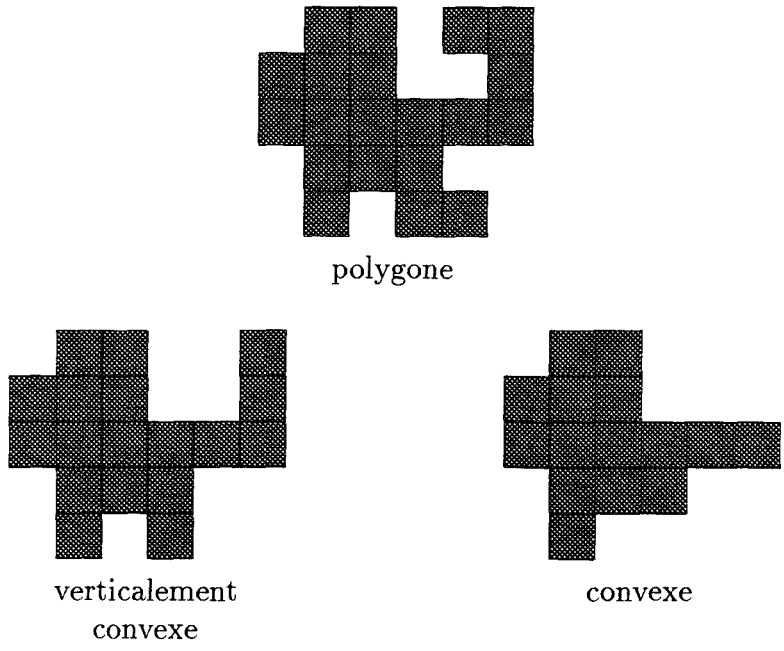



FIG. 4 - *Exemple des différentes classes de polyominos*

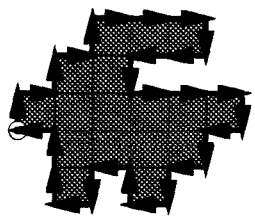


FIG. 5 - *Mot de contour de polyomino* $(ur)^3r^2dl^2dr^3(dl)^3uldlu^2l$

Figure 5).

Soient $p$ une propriété sur les mots de figures et $L$ un langage de mots de figures, on voudrait savoir s'il existe dans $L$ un mot qui vérifie $p$. Par exemple, la propriété $p$ peut être « est un mot de contour de polyomino » ou « est un mot minimal ». Un récapitulatif des résultats connus sur ces propriétés existentielles est donné Figure 6 reprenant les résultats de [Das88, Das89, Bea91, Das91, BLS92, Rob94, Rob96].

Toutes les sémantiques auxquelles nous avons fait allusion ci-dessus ne permettent de décrire que des structures filaires. Cette restriction ne permet pas de décrire simplement que des polyominos sans trous (voir Figure 5). Une solution, proposée par M. Nivat, est de parcourir le contour du polyomino par une boucle dans le sens anti-trigonométrique et d'y insérer le parcours des trous dans le sens trigonométrique (voir Figure 7). Une autre solution consiste à parcourir l'animal du polyomino, c'est-à-dire se déplacer du centre d'une

Langages de figures sans lever de crayon

| Propriété | Rat | Lin |
|---|---|---|
| est un mot de contour de polyomino | Indécidable | |
| est un mot de contour de polyomino verticalement convexe | Indécidable | |
| mot de contour de polyomino convexe | Décidable | Indécidable |
| décrit un arbre | Indécidable | |
| est un mot minimal | Indécidable | |
| est un mot optimal† | Indécidable | |

Langages de figures avec traits invisibles

| Propriété | Rat | Lin |
|---|---|---|
| décrit une figure connexe | Indécidable | |
| décrit une figure non connexe | ? | Indécidable |
| décrit une boucle | Décidable | Indécidable |

† Un mot est optimal s'il ne trace chaque segment qu'une seule fois.

FIG. 6 - *Résultats sur les propriétés existentielles*

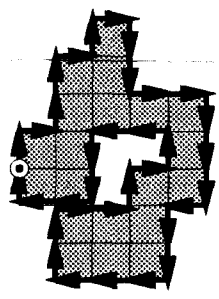


FIG. 7 - *Mot de contour de polyomino avec un trou*

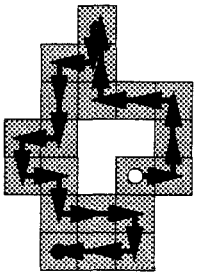


FIG. 8 - *Parcours de l'animal d'un polyomino*

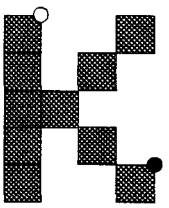


FIG. 9 - *Exemple de figure décrite par les langages de figures « à pixels »*

cellule au centre d'un voisin comme dans la Figure 8 [TT94].

Dans la première partie du mémoire, nous proposons une sémantique différente déjà évoquée dans [Rob96] qui permet de décrire des figures apparentées à la figure illustrée Figure 9. Dans le Chapitre 1, nous donnons les définitions de base de cette nouvelle sémantique. Nous utilisons toujours le principe des déplacements élémentaires induits par les lettres de l'alphabet $\Pi = \{u, r, d, l\}$ sur la grille cartésienne $\mathbb{Z}^2$, mais plutôt que de tracer le segment sous le déplacement, c'est le pixel à droite du déplacement qui est « allumé ». Hormis l'aspect ludique de ce choix, ceci nous a permis de nous interroger sur — et d'exhiber — les raisons pour lesquelles certains résultats s'étendent naturellement des langages de figures « classiques » aux langages de figures « à pixels » [Pin]. En effet, l'ensemble des figures à segments et l'ensemble des figures à pixels sont tous deux des monoïdes inversifs finiment engendrés et leur représentation par un monoïde libre ($\Pi^*$ en l'occurence) rentre dans le cadre des systèmes générateurs [Sak81].

Cette propriété algébrique de l'ensemble des figures à segments affleure d'ailleurs déjà dans l'étude qui en a été faite dans [CW83, Péc85, Bir91, SS94]. Il ne s'agit pas là d'une remarque anodine puisque certains résultats démontrés pour les langages de figures sont inhérents aux monoïdes inversifs. Par exemple le fait que les langages de mots récursivement énumérables n'ont pas plus de pouvoir d'expression que les langages de mots contextuels [MRW82] est une propriété pour toute représentation d'un monoïde inversif [Sil]. Ceci nous permet de mettre à profit les résultats déjà connus dans les monoïdes inversifs [Mun74, Pet84, MP84, MMS95, Wei].

Et à l'inverse, des propriétés qui semblaient propres aux langages de figures s'étendent aux monoïdes inversifs. Dans le chapitre 2, nous montrons que des systèmes de réécriture

très proches de celui donné dans [SS91] peuvent être utilisés pour les monoïdes inversifs.

Le Chapitre 3 concerne l'étude de la complexité descriptive des figures à pixels. Le Chapitre 4, quant à lui, considère la décidabilité, ou plutôt l'indécidabilité, de propriétés existentielles dans les langages de figures en reprenant la méthode exposée dans [Rob96] et l'applique aux langages de figures classiques et, finalement, aux langages de figures à pixels.

## Mots à deux dimensions

Parmi les extensions au cas à deux dimensions des notions rencontrées dans le monoïde libre, telles que la reconnaissabilité, plusieurs consistent à étudier la génération de figures à partir de grammaires, prolongement des différents types de grammaire que l'on trouve dans la hiérarchie de Chomsky [MR71, Pfa72, RM72, SSK73].

D'autres approches considèrent des machines à nombre fini d'états fonctionnant sur des mot à deux dimensions, tableaux rectangulaires contenant des symboles — on pourrait parler de « couleurs » choisies dans une « pallette », mais pour rester proche de la terminologie des langages formels nous garderons le vocabulaire « lettre » et « alphabet ». Par exemple, M. Blum et C. Hewitt considèrent les ensembles de figures reconnus par des automates à quatre directions (« four-way automata » en anglais). Il s'agit d'une variante des automates finis classiques où la tête de lecture se déplace dans les quatre directions lors de la transition [BH67]. K. Inoue et A. Nakamura introduisent dans [IN77b] des automates mosaïque en ligne qui sont un cas particulier d'automates cellulaires non-déterministes à deux dimensions.

Il est d'ailleurs intéressant de noter que la classe de langages de figures reconnaissables par automates mosaïque en ligne déterministes est proprement incluse dans celle des langages reconnaissables par des automates non-déterministes. De la même manière les automates à quatre directions non-déterministes sont strictement plus puissants que les déterministes. Ces deux types d'automates — à quatre directions ou mosaïque en ligne — ne sont néanmoins pas sans rapport puisque tout langage de figures accepté par un automate à quatre directions l'est également par un automate mosaïque en ligne. Parmi les propriétés remarquables, il est à signaler que la classe des langages reconnus par automates à mosaïque en ligne n'est pas close par complémentaire et que le vide n'y est pas décidable [IN77a, IN79, IT90].

D. Giammarresi et A. Restivo donnent dans [GR92b] une définition de reconnaissabilité directement issue de la théorie des langages formels. Il est bien connu que tout langage de mots reconnaissable est l'image par un homomorphisme lettre-à-lettre d'un langage de mots local. Un langage de mots local $L$ sur $\Sigma$ est complètement défini par un ensemble $\Delta \subseteq (\Sigma \cup \{\#\})^2$ où $\#$ est une lettre spéciale n'appartenant pas à $\Sigma$, le langage $L$ est l'ensemble des mots $\omega$ tels que les facteurs de longueur deux de $\#\omega\#$ sont tous dans $\Delta$ — cet ensemble de facteurs doit, en quelque sorte, « recouvrir » le mot. Pour étendre aux figures cette notion de langage local, les auteurs considèrent les sous-figures de taille $(2,2)$

à la place des facteurs de longueur deux. Un langage de figures $L$ sur $\Sigma$ est défini par un ensemble $\Theta$ de pavés $(2,2)$ sur $\Sigma \cup \{\#\}$ où $\# \notin \Sigma$, le langage $L$ est l'ensemble des figures $f$ telles que les sous-figures de taille $(2,2)$ de la figure $f$ entourée de $\#$ sont toutes dans $\Theta$.

Les langages de figures reconnaissables sont alors définis naturellement comme les images par projection, sorte d'homomorphisme lettre-à-lettre, d'un langage de figures local. La généralisation au cas à deux dimensions nous amène à définir deux produits de concaténation. La concaténation horizontale de deux figures consiste à mettre côte-à-côte les deux figures mais uniquement si elles ont même hauteur afin d'obtenir toujours un rectangle. De même pour la concaténation verticale, on place les figures l'une en-dessous de l'autre, mais seulement si elles ont même largeur. On peut définir alors l'étoile verticale et l'étoile horizontale qui correspondent à l'itération de la concaténation verticale ou horizontale. La classe des langages reconnaissables est ainsi close par union, intersection, concaténation horizontale ou verticale et étoile horizontale ou verticale.

La classe des langages de figures reconnaissables ainsi obtenue correspond exactement à la famille des langages de figures reconnaissables par automate mosaïque en ligne [IT91, GR96b]. Cette classe est également la classe des langages de figures définissables par des expressions existentielles dans la logique monadique du second-ordre [GRST96].

En fait les figures (comme les arbres ou les mots) sont un cas particulier d'ordres partiels et peuvent ainsi s'inscrire dans une théorie plus large. On pourra retrouver plus de précisions sur ceci dans [BMPS92, PST94, BDW95, Cou96, Tho97].

Parmi les études sur les langages de mots à deux dimensions reconnaissables menées depuis l'introduction de cette nouvelle définition, nous mentionnons une synthèse publiée dans le *handbook of formal languages* [GR96b] qui reprend, entre autres, les résultats déjà cités ainsi que ceux de [GR92a, GR96a]. Il est aussi à noter que si les propriétés de clotûre énoncées précédemment sont encourageantes, on ne sait pas définir les langages de figures reconnaissables à partir d'expressions régulières qui n'utilisent pas l'intersection et la projection. Par exemple la plus petite classe de langages de figures qui contient les langages finis et close par union, concaténation et étoile ne coïncide pas avec la classe des reconnaissables — on ne possède pas à l'heure actuelle d'équivalent au théorème de Kleene [GR96b, Mat97].

Toujours dans l'optique des comparaisons entre langages de mots et langages de figures, T. Wilke s'est intéressé aux langages de figures sans étoile — définissable uniquement à partir des singletons, des opérations booléennes et des concaténations — et montre dans [Wil97] que, contrairement à l'égalité dans les mots, les langages de figures sans étoile ont strictement moins de pouvoir que les expressions logiques dans la logique du premier ordre.

Enfin, citons [Gia93] où D. Giammarresi étudie les fonctions reconnaissables — une fonction $f$ est reconnaissable s'il existe un langage de figures reconnaissable où les figures de hauteur $n$ sont de taille $f(n)$.

L'auteur a appris récemment que d'une manière indépendante, K. Lindgren et al. ont définis dans [LMN97] la même classe de langages de figures reconnaissables dans le but d'étudier certains phénomènes dans les systèmes dynamiques.

La deuxième partie de cet ouvrage est consacrée à l'étude des langages de mots à deux dimensions. Dans le Chapitre 1, nous donnons, de manière formelle, les définitions de base de cette théorie.

Le Chapitre 2 étudie la possibilité de séparer contrôle horizontal et contrôle vertical. En effet avec les langages locaux ces contrôles sont faits simultanément puisque nous utilisons des pavés $(2,2)$. Ainsi nous définissons les langages *hv-locaux* où ces pavés sont remplacés par des dominos et nous montrons que, par projection, nous obtenons bien la classe des reconnaissables. Ainsi les langages de figures peuvent être définis à l'aide de deux langages de mots, l'un d'eux définissant les mots autorisés horizontalement et l'autre les mots autorisés verticalement — on dit que l'on fait le « croisement » des deux langages de mots. Ainsi les langages hv-locaux sont définis à l'aide de deux langages de mots locaux, ce qui autorise des manipulations plus aisés dans les preuves. Alors que dans leur définition les langages reconnaissables utilisent un recouvrement par un ensemble fini de pavés $(2,2)$, pour les locaux, ou des dominos, pour les hv-locaux, on peut définir des opérations de pavage qui correspondent plus à une partition d'une figure en composantes appartenant à un ensemble de figures donné. Le Chapitre 3 s'intéresse à la caractérisation des langages de figures reconnaissables par des intersections de pavages d'ensembles finis de figures.

Les chapitres 4 et 5 étudient les frontières — lignes supérieures — de différentes classes de langages de figures. Il est connu que la classe des frontières des langages d'arbres reconnaissables est la classe des langages algébriques. Dans le Chapitre 4, nous montrons que la classe des frontières des langages de figures reconnaissables est la classes des langages contextuels. Ceci signifie que le langage-frontière du « croisement » de deux langages de mots locaux est un langage contextuel. On étudie dans le Chapitre 5 les frontières des langages de figures obtenus par « croisement » de deux langages de mots suivant la hiérarchie de Chomsky. Les études précédentes, et surtout celles du Chapitre 4, semblent montrer que les langages de figures reconnaissables sont un modèle de calcul intéressant qui n'est pas sans rappeler les automates cellulaires.

# Références

[BD89] F. J. Brandenburg and J. Dassow. Reduction of picture words. Technical Report MIP 8905, Univ. Passau, Germany, 1989.

[BD93] F. J. Brandenburg and J. Dassow. Efficient reductions of picture words. *Informatique Théorique et Applications/Theoretical Informatics and Applications*, 27(1):49–56, 1993.

[BDW95] F. Bossut, M. Dauchet, and B. Warin. A Kleene Theorem for a class of planar acyclic graphs. *Information and Computation*, 117(2):251–265, 1995.

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

[BH67] M. Blum and C. Hewitt. Automata on two-dimensional tape. In *Proc. IEEE 8th Annual Symposium on Switching and Automata Theory*, IEEE Symposium on Switching and Automata Theory, pages 155–160, 1967.

[Bir91] J.-C. Birget. Strict local testability of the finite control of two-way automata and of regular picture description languages. *International Journal of Algebra and Computation*, 1(2):161–175, 1991.

[BLS92] D. Beauquier, M. Latteux, and K. Slowinski. A decidability result about convex polyominoes. In I. Simon, editor, *Proc. Latin American Theoretical INformatics*, volume 583 of *Lecture Notes in Computer Science*, pages 32–45, Sao Paulo, Brazil, 1992. Springer-Verlag, Berlin.

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

[BMPS92] P. Bonizzoni, G. Mauri, G. Pighizzini, and N. Sadadini. Recognizing sets of labelled acyclic graphs. In M. Nivat and A. Podelski, editors, *Tree Automata and Languages*, pages 201–224. Elsevier Science Publisher, Amsterdam, 1992.

[BN90] D. Beauquier and M. Nivat. Tiling the plane with one tile. In *Annual ACM Symposium on Computational Geometry*, 1990.

[BN91] D. Beauquier and M. Nivat. Tiling pictures of the plane with two bars, a horizontal and a vertical one. Technical Report 91.67, L.I.T.P., Univ. Paris 7, France, 1991.

[Bra89a] F. J. Brandenburg. Cancellations in linear context-free languages. Technical Report MIP 8904, Univ. Passau, Germany, 1989.

[Bra89b] F. J. Brandenburg. On minimal picture words. Technical Report MIP 8903, Univ. Passau, Germany, 1989.

[Clo67] M. Clowes. A generative picture grammar. In *Proc. Computing Research Section*, Melbourne, Australia, 1967. Commonwealth Scientific and Industrial Research Organization.

[CM91] J.-M. Chassery and A. Montanvert. *Géométrie Discrètes en Analyse d'Images.* Hermes, Paris, 1991.

[Cou96] B. Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg, editor, *Handbook of Graph Transformation*, volume I, Foundations. World Scientific, Singapore, 1996.

[CW83] K. Culik II and E. Welzl. Two-way finite state generators. In M. Karpinsky, editor, *Foundations of Computation Theory*, volume 158 of *Lecture Notes in Computer Science*, pages 106–114. Springer, Berlin, 1983.

[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. *Journal of Information Processing and Cybernetics EIK*, 25:423–433, 1989.

[Das91] J. Dassow. On the connectedness of pictures in chain code picture languages. *Theoretical Computer Science*, 81(2):289–294, 1991.

[DH93] J. Dassow and F. Hinz. Decision problems and regular chain code picture languages. *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 45, 1993.

[DV84] M.-P. Delest and X. Viennot. Algebraic languages and polyominoes enumeration. *Theoretical Computer Science*, 34(1/2):169–206, 1984.

[Ell82] D. Ellard. Polyominoes and enumeration. *Math. Gazette*, 66:130–314, 1982.

[Fed68] J. Feder. Languages of encoded line patterns. *Information and Control*, 13(3):230–244, 1968.

[Fre61] H. Freeman. On encoding arbitrary geometric configurations. *IRE Transactions on Electronic Computers*, 10:260–268, 1961.

[Fre62] H. Freeman. On the digital computer classification of geometric line patterns. In *Proc. 18th Nat. Elect. Conf.*, 1962.

[Fre74] H. Freeman. Computer processing of line-drawing images. *ACM Computing Surveys*, 6(1):57–97, 1974.

[Fu82] K. S. Fu. *Syntactic Pattern Recognition and Applications.* Prentice-Hall, Englewoods Cliffs, 1982.

[Gia93] D. Giammarresi. Two-dimensional languages and recognizable functions. In G. Rozenberg and A. Salomaa, editors, *Proc. Developments in Language Theory*, pages 290–301, Turku, Finland, 1993. World Scientific Publishing Co.

[Gol66a] S. W. Golomb. *Polyominoes.* Georges Allen and Unwin Ltd, London, 1966.

[Gol66b] S. W. Golomb. Tiling with polyominoes. *Journal of Combinatorial Theory*, 1:280–296, 1966.

[Gol70] S. W. Golomb. Tiling with sets of polyominoes. *Journal of Combinatorial Theory*, 9:60–71, 1970.

[GR92a] D. Giammarresi and A. Restivo. A new notion of recognizability for two-dimensional languages. In *Proc. Journées Internationales sur les Polyominos et Pavages*, pages 127–139, Lyon, France, 1992.

[GR92b] D. Giammarresi and A. Restivo. Recognizable picture languages. *International Journal Pattern Recognition and Artificial Intelligence*, pages 31–46, 1992. Special Issue on *Parallel Image Processing*. M. Nivat, A. Saoudi and P.S.P. Wangs (Eds.).

[GR96a] D. Giammarresi and A. Restivo. Two-dimensional finite state recognizability. *Fundamenta Informaticae*, 25(3/4):399–422, 1996. Special issue: Formal Language Theory.

[GR96b] D. Giammarresi and A. Restivo. Two-dimensional languages. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3, pages 215–267. Springer-Verlag, Berlin, 1996.

[GRST96] D. Giammarresi, A. Restivo, S. Seibert, and W. Thomas. Monadic second-order logic over rectangular pictures and recognizability by tiling systems. *Information and Computation*, 125(1):32–45, 1996.

[GS86] B. Grünbaum and G. C. Shepard. *Tilings and Patterns*. W. H. Freeman and Company, New York, 1986.

[Gut89] R. Gutbrod. A transformation system for generating description languages of chain code pictures. *Theoretical Computer Science*, 68(3):239–252, 1989.

[Gut91a] R. Gutbrod. Branch picture languages. Technical report, Techn. Univ. Aachen, Germany, 1991.

[Gut91b] R. Gutbrod. A transformation system for chain code picture languages: Properties and algorithms. Technical Report 10, Univ. Würzburg, Germany, 1991.

[Hin] F. Hinz. The equivalence problem for three-way picture languages. Manuscript.

[Hin86] F. Hinz. Regular chain code picture languages of nonlinear descriptional complexity. In J. Gruska, B. Rovan, and J. Wiedermann, editors, *Proc Mathematical Foundations of Computer Science*, volume 233 of *Lecture Notes in Computer Science*, pages 414–421, Bratislava, Czechoslovakia, 1986. Springer-Verlag, Berlin.

[Hin88] F. Hinz. Questions of decidability for context-free chain code picture languages. In *Proc. 5th International Meeting of Young Computer Scientists*, Smolenice Castle, Hungary, 1988.

[Hin89] F. Hinz. Classes of picture languages that cannot be distinguished in the chain code concept and deletion of redundant retreats. In *Proc. Annual Symposium on Theoretical Aspects of Computer Science*, volume 349 of *Lecture Notes in Computer Science*, pages 132–143. Springer-Verlag, Berlin, 1989.

[Hin90] F. Hinz. The membership problem for context-free chain code picture languages. In *Proc. Mathematical Foundations of Computer Science*, volume 452 of *Lecture Notes in Computer Science*, pages 329–336. Springer-Verlag, Berlin, 1990.

[HW88] F. Hinz and E. Welzl. Regular chain code picture languages with invisible lines. Technical Report 252, I.I.G., Techn. Univ. Graz, Austria, 1988.

[IN77a] K. Inoue and A. Nakamura. Nonclosure properties of two-dimensional on-line tesselation acceptors and one-way parallel sequential array acceptors. *Trans. of IECE of Japan*, 6:475–476, 1977.

[IN77b] K. Inoue and A. Nakamura. Some properties of two-dimensional on-line tesselation acceptors. *Information Sciences*, 13:95–121, 1977.

[IN79] K. Inoue and A. Nakamura. Two-dimensional finite automata and unacceptable functions. *Intern. J. Comput. Math.*, 7:207–213, 1979.

[IT90] K. Inoue and I. Takanami. A survey of two-dimensional automata theory. In J. Dassow and J. Kelemen, editors, *Proc. Aspects and Prospects of Theoretical Computer Science, 5th International Meeting of Young Computer Scientists*, volume 381 of *Lecture Notes in Computer Science*, pages 72–91. Springer-Verlag, Berlin, 1990.

[IT91] K. Inoue and I. Takanami. A characterization of recognizable picture languages. In *Proc. 2nd International Colloquium on Parallel Image Processing*, volume 654 of *Lecture Notes in Computer Science*, pages 133–143. Springer-Verlag, Berlin, 1991.

[Kim90a] C. Kim. Complexity and decidability for restricted classes of picture languages. *Theoretical Computer Science*, 73(3):295–311, 1990.

[Kim90b] C. Kim. Picture iteration and picture ambiguity. *Journal of Computer and System Sciences*, 40(3):289–306, 1990.

[Kim94] C. Kim. Retreat bounded picture languages. *Theoretical Computer Science*, 132(1/2):85–112, 1994.

[Kim96] C. Kim. Unambiguous description of chain code picture languages. *Information Processing Letters*, 58(2):75–79, 1996.

[Kir64] R. Kirsch. Computer interpretation of English text and pattern recognition. *IEEE Transactions on Electronic Computers*, 13:363–376, 1964.

[KS87] C. Kim and I. H. Sudborough. The membership and equivalence problems for picture languages. *Theoretical Computer Science*, 52(3):177–191, 1987.

[KS92] C. Kim and I. H. Sudborough. On reversal-bounded picture languages. *Theoretical Computer Science*, 104(2):185–206, 1992.

[KW67] P. J. Knoke and R. G. Wiley. A linguistic approach to mechanical pattern recognition. In *Proc. IEEE Comp. Conf.*, pages 142–144, 1967.

[LMN97] K. Lindgren, C. Moore, and M. Nordahl. Complexity of two-dimensional patterns. Technical Report 97–03–023, Santa Fe Institute, Unit. States, 1997.

[Mat97] O. Matz. Regular expressions and contextfree grammars for picture languages. In *Proc. STACS'97*, volume 1200 of *Lecture Notes in Computer Science*, pages 283–294, Lübeck, Germany, 1997. Springer-Verlag, Berlin.

[Min71] M. Minsky. *Perceptron.* M.I.T. Press, Cambridge, Mass., 1971.

[MMS95] S. Margolis, J. Meakin, and M. Sapir. Algorithmic problems in groups, semigroups and inverse semigroups. In *Proc. NATO Conference on Groups and Semigroups*, Foutain, pages 147–214, 1995.

[MP84] S. Margolis and J.-É. Pin. Languages and inverse semigroups. In *Proc. Annual International Colloquium on Automata, Languages and Programming*, 1984.

[MR71] D. L. Milgram and A. Rosenfeld. Array automata and array grammars. In *IFIP Congress 71*, volume Booklet TA2, pages 166–173. North-Holland, Amsterdam, 1971.

[MRW82] H. A. Maurer, G. Rozenberg, and E. Welzl. Using string languages to describe picture languages. *Information and Control*, 54(3):155–185, 1982.

[MRW83] H. A. Maurer, G. Rozenberg, and E. Welzl. Chain code picture languages. In H. Ehrig, M. Nagl, and G. Rozenberg, editors, *Graph Grammars and their Application to Computer Science*, volume 153 of *Lecture Notes in Computer Science*, pages 232–244. Springer, Berlin, 1983.

[Mun74] W. D. Munn. Free inverse semigroup. *Proceeding London Mathematical Society*, 29:385–404, 1974.

[Myl72] J. Mylopoulos. On the application of formal language and automata theory to pattern recognition. *Pattern Recognition*, 44:37–51, 1972.

[Péc85] J.-P. Pécuchet. Automates Boustrophédons, semi-groupe de Birget et monoïde inversif libre. *RAIRO Informatique Théorique et Applications/Theoretical Informatics and Applications*, 19(1):71–100, 1985.

[Pen78] R. Penrose. Pentaplexity. In *Eureka*, volume 39. 1978.

[Pet84] M. Petrich. *Inverse Semigroups.* Wiley, New York, 1984.

[Pfa72] J. L. Pfalz. Web grammars and picture description. *Computer Graphics and Image Processing*, 1:193–220, 1972.

[Pin] J.-É. Pin. Communications personnelles.

[PRS93] G. Păun, D. Robilliard, and K. Slowinski. Connected pictures and minimal words with blank moves. In *Proc. Automata and Formal Languages*, volume 48 (1996) of *Publicationes Mathematicae*, pages 375–387, Salgótarján, Hongrie, 1993. Institum Mathematicum Universitatis Debreceniensis.

[PST94] A. Potthoff, S. Seibert, and W. Thomas. Nondeterministic versus deterministic of finite automata over directed acyclic graphs. *Bulletin of the Belgian Mathematical Society - Simon Stevin*, 1:285–298, 1994.

[RM72] A. Rosenfeld and D. L. Milgram. Web automata and web grammars. *Machine Intelligence*, 7:307–324, 1972.

[Rob71] R. M. Robinson. Undecidability and non periodicity of tilings of the plane. *Inventione Mathematicae*, 12:177–209, 1971.

[Rob94] D. Robilliard. Questions indécidables dans les langages de figures. Technical Report it-262, L.I.F.L., Univ. Lille 1, France, June 1994.

[Rob96] D. Robilliard. *Langages de Figures.* PhD thesis, Univ. Lille 1, France, January 1996.

[Ros79] A. Rosenfeld. *Picture Languages – Formal Models of Picture Recognition.* Academic Press, New York, 1979.

[RR94] B. Ratoandromanana and D. Robilliard. Superposition in picture languages. In *Proc. CAAP*, volume 787 of *Lecture Notes in Computer Science*, pages 322–334, Edinburg, Scotland, 1994. Springer-Verlag, Berlin.

[Sak81] J. Sakarovitch. Description des monoïdes de type fini. *Journal of Information Processing and Cybernetics EIK*, 17(8):417–434, 1981.

[Sha69] A. C. Shaw. A formal picture description scheme as a basis for picture processing systems. *Information and Control*, 14(1):9–52, 1969.

[Sil] P. V. Silva. On free inverse monoid languages. *RAIRO Informatique Théorique et Applications/Theoretical Informatics and Applications.* à paraître.

[Slo93] K. Slowinski. Picture words with invisible lines. *Theoretical Computer Science*, 108(2):357–363, 1993.

[SS90] P. Séébold and K. Slowinski. Minimizing picture words. In J. Dassow and J. Kelemen, editors, *Proc. Aspects and Prospects of Theoretical Computer Science, 6th International Meeting of Young Computer Scientists*, volume 464 of *Lecture Notes in Computer Science*, pages 234–243. Springer-Verlag, Berlin, 1990.

[SS91] P. Séébold and K. Slowinski. The shortest way to draw a connected picture. *Computer Graphics Forum*, 10(4):319–327, 1991.

[SS94] P. Séébold and K. Slowinski. Redundant retreat-free word. In G. Păun, editor, *Mathematical Aspects of Natural Languages*, pages 419–430. World Sci. Publi., Singapour, 1994.

[SSK73] G. Siromoney, R. Siromoney, and K. Krithivasan. Picture languages with array rewriting rules. *Information and Control*, 22(5):447–470, June 1973.

[SW85] I. H. Sudborough and E. Welzl. Complexity and decidability for chain code picture languages. *Theoretical Computer Science*, 36(2/3):173–202, 1985.

[Tho97] W. Thomas. Automata theory on trees and partial orders. In *Proc. TAPSOFT'97*, volume 1214 of *Lecture Notes in Computer Science*, pages 20–34, Lille, France, 1997. Springer-Verlag, Berlin.

[TT94] J. Tatari and É. Tosan. Représentation de polyominos par des mélanges de mots parenthèsés. In *Proc. 3èmes Journées Internationales sur les Polyominos et Pavages*, pages 162–182, Toulouse, France, 1994.

[Wei] P. Weil. Communications personnelles.

[Wil97] T. Wilke. Star-free picture expressions are strictely weaker than first-order logic. In *Proc. 24th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science, Bologna, Italy, 1997. Springer-Verlag, Berlin. à paraître.

# Notations

# Notations

Nous supposons le lecteur familier avec la théorie des langages formels et nous ne précisons que quelques notations. Si ces précisions lui semblent toutefois incomplètes, il peut se reporter aux ouvrages de J. Berstel [Ber79], de S. Eilenberg [Eil74] ou de S. Ginsburg [Gin75].

L'ensemble des entiers naturels est noté $\mathbb{N}$, l'ensemble des entiers relatifs $\mathbb{Z}$ et l'ensemble des réels est noté $\mathbb{R}$.

Un *semi-groupe* est un ensemble $S$ muni d'une opération interne associative. Un *monoïde* est un semi-groupe possédant un élément neutre.

Un *alphabet* est un ensemble fini, éventuellement infini si précisé, et est désigné par une lettre grecque majuscule ($\Sigma$, $\Pi$, $\Psi$...). Les éléments d'un alphabet sont des *lettres* et sont désignés par des lettres minuscules (on notera par exemple $a \in \Sigma$...). Une chaîne finie sur un alphabet est un *mot* et est notée de préférence avec une lettre grecque minuscule ($\alpha$, $\beta$, $\omega$...). L'ensemble $\Sigma^*$ représente le monoïde libre engendré par $\Sigma$ et on notera $\varepsilon$ le mot vide, enfin $\Sigma^+$ représente le semi-groupe libre engendré par $\Sigma$.

Pour un mot $\omega \in \Sigma^*$, $|\omega|$ dénote sa longueur alors que $|\omega|_a$ représente le nombre d'occurences de la lettre $a$ dans $\omega$. On désigne par $\tilde{\omega}$ l'image miroir de $\omega$. Pour deux mots $\alpha, \beta \in \Sigma^*$, on note $\alpha \leq \beta$ le fait que $\alpha$ est un préfixe de $\beta$, c'est-à-dire qu'il existe $\alpha' \in \Sigma^*$ tel que $\beta = \alpha.\alpha'$. Pour deux mots $\alpha, \beta \in \Sigma^*$, $\alpha \sqcup \beta$ représente l'ensemble des mots obtenus par *shuffle*: $\alpha \sqcup \beta = \{\alpha_1.\beta_1 \dots \alpha_n\beta_n \mid \alpha_i, \beta_i \in \Sigma^* \quad \alpha_1 \dots \alpha_n = \alpha \quad \beta_1 \dots \beta_n = \beta\}$.

Un monoïde $M$ est *inversif* si, pour tout $x \in M$, il existe un et un seul élément noté $x^{-1}$ tel que $x.x^{-1}.x = x$ et $x^{-1}.x.x^{-1} = x^{-1}$ que l'on appelle inverse de $x$. Pour plus de renseignements, le lecteur pourra se reporter à l'ouvrage de M. Petrich [Pet84].

Soit $M$ un monoïde, un *système générateur* de $M$ est un couple $(\Sigma, h)$ où $\Sigma$ est un alphabet et $h$ un homomorphisme de $\Sigma^*$ dans $M$ tels que $h(\Sigma^*) = M$. Pour plus de précisions, nous invitons le lecteur à se reporter à l'article de J. Sakarovitch [Sak81]. La *congruence associée à un système générateur* $(\Sigma, h)$ est la congruence $\propto$ définie par : pour tout couple de mots $\omega, \omega' \in \Sigma^*$, $\omega \propto \omega'$ si et seulement si $h(\omega) = h(\omega')$.

Les langages de la hiérarchie de Chomsky sont notés de la manière suivante: $\mathrm{Loc}(\Sigma^*)$ désigne la classe des *langages locaux*; $\mathrm{Rat}(\Sigma^*) = \mathrm{Rec}(\Sigma^*)$ représentent respectivement les

*langages rationnels* de $\Sigma^*$ et les *langages reconnaissables*; Lin($\Sigma^*$) désigne les *langages linéaires*; Dét($\Sigma^*$) les *langages algébriques déterministes*; Alg($\Sigma^*$) les *langages algébriques*; CS($\Sigma^*$) (comme « context-sensitive ») les *langages contextuels* — notons que nous considérons conformément à [Gin75], que les langages contextuels ne contiennent pas le mot vide; r.é.($\Sigma^*$) les *langages récursivement énumérables*.

# Références

[Ber79] J. Berstel. *Tranductions and Context-Free Languages.* Teubner Studienbücher, Stuttgard, 1979.

[Eil74] S. Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, New York, 1974.

[Gin75] S. Ginsburg. *Algebraic and Automata-Theoretic Properties of Formal Languages.* North-Holland, Amsterdam, 1975.

[Pet84] M. Petrich. *Inverse Semigroups.* Wiley, New York, 1984.

[Sak81] J. Sakarovitch. Description des monoïdes de type fini. *Journal of Information Processing and Cybernetics EIK*, 17(8):417–434, 1981.

# Première partie

# Langages de mots de figures

# Chapitre 1

# Définitions de base

Nous allons décrire dans ce chapitre les objets graphiques que nous allons manipuler : dessins et figures. L'originalité de notre approche réside dans le fait qu'elle s'appuie sur des propriétés algébriques de l'ensemble des figures pointées (figures avec deux points distingués qui permettent de définir une concaténation) qui est un monoïde inversif finiment engendré. Plus qu'une adaptation des travaux de Maurer et al. [MRW82] nous recentrons l'étude des langages de figures dans un cadre où nous disposons d'outils bien connus.

Ce chapitre s'inspire pour une grande part des travaux publiés dans [LRS97].

## 1.1 Des dessins

Les objets graphiques que nous nous proposons d'étudier sont des ensembles finis de pixels, un pixel étant un carré unitaire $[i, i+1] \times [j, j+1]$ de $\mathbb{R}^2$ avec $(i, j) \in \mathbb{Z}^2$. L'ensemble des pixels est noté P. Il va de soi qu'un pixel peut être désigné sans ambiguïté par son coin inférieur gauche. On dénote $\text{pix}(i, j)$ avec $(i, j) \in \mathbb{Z}^2$ le pixel $[i, i+1] \times [j, j+1]$.

Un *dessin* est défini comme étant un ensemble fini de pixels (voir Figure 1.1). On note $\mathcal{D}$ l'ensemble des dessins.

Un *dessin pointé* est un dessin muni de deux points de $\mathbb{Z}^2$ distingués appelés respectivement *point de départ* et *point d'arrivée*. Ceci nous servira notamment à concaténer deux dessins comme nous le verrons plus loin. Un dessin pointé $q$ est un triplet $(p, d, a)$ où $p$ est un dessin, $d$ et $a$ sont deux points de $\mathbb{Z}^2$, $p$ est appelé la *base* du dessin pointé et est notée $\text{base}(q)$, $d$ est appelé le *point de départ* du dessin pointé et est noté $\text{dép}(q)$, enfin, $a$ est appelé *point d'arrivée* du dessin pointé et est noté $\text{arr}(q)$ (voir Figure 1.2). L'ensemble des dessins pointés est noté $\mathcal{D}_p$.

Par abus de langage, on dira qu'un pixel appartient à un dessin pointé alors qu'il faudrait dire que le pixel appartient à la base du dessin pointé.

Le *poids* d'un dessin $p$ est noté $||p||$ et correspond à son cardinal, c'est-à-dire au nombre de pixels qu'il contient. Le poids d'un dessin pointé $q$ est le poids de sa base. Soient deux dessins (éventuellement pointés) $p$ et $q$, on dit que $p$ est plus petit que $q$ si $||p|| \leq ||q||$.

Les points de départ et d'arrivée nous permettent de concaténer deux dessins pointés.

*La figure ci-dessous illustre le dessin $p \in \mathcal{D}$ avec :*

$$p = \{\text{pix}(0,-1), \text{pix}(0,0), \text{pix}(0,2), \text{pix}(2,0)\}$$

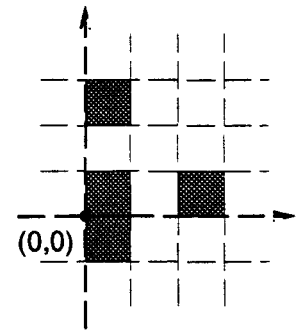
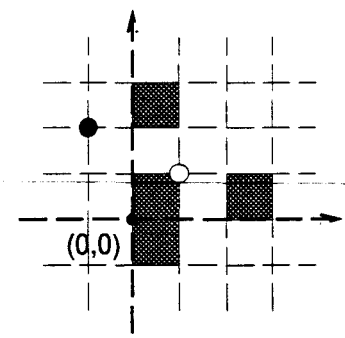



FIG. 1.1 - *Exemple de dessin*

*La figure ci-dessous montre le dessin pointé $q \in \mathcal{D}_p$ avec :*

$$q = (p, (1,1), (-1,2))$$




*Par convention, nous représentons le point de départ par un rond blanc et le point d'arrivée par un rond noir.*

FIG. 1.2 - *Exemple de dessin pointé*

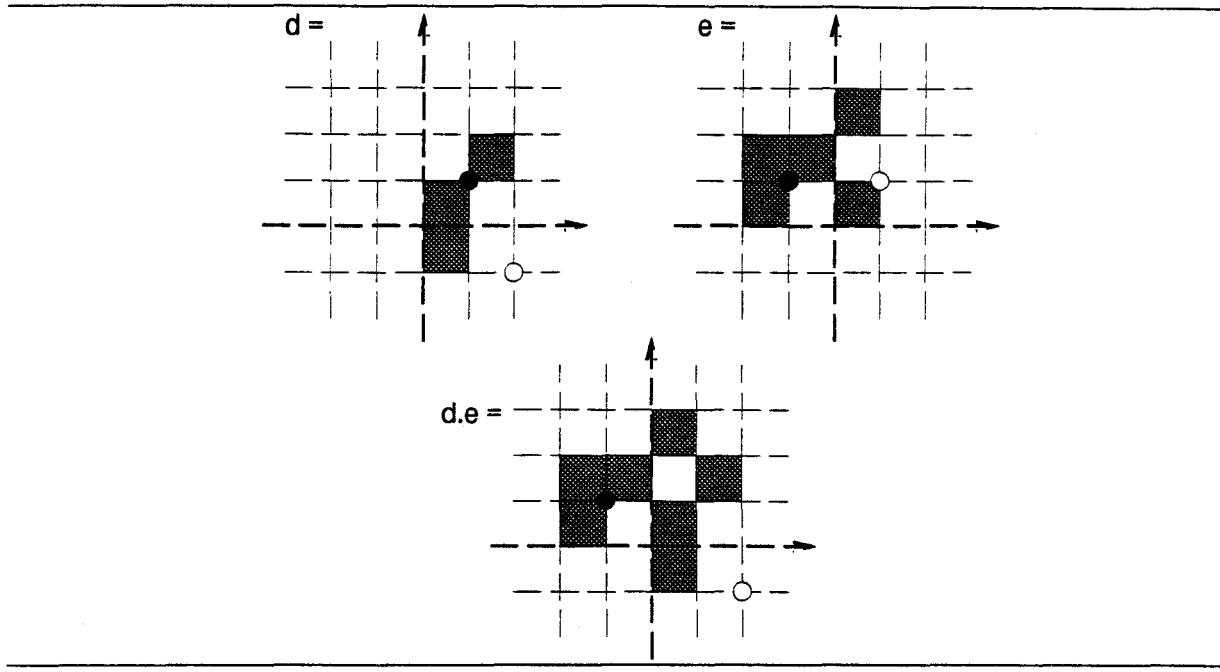



FIG. 1.3 - *Exemple de concaténation définie*

La concaténation de deux dessins pointés $p$ et $q$ n'est définie que si $\text{arr}(p) = \text{dép}(q)$, et dans ce cas elle correspond à l'union des bases. Un exemple de concaténation définie est donnée Figure 1.3.

**Définition 1.1** *Soient $p, q \in \mathcal{D}_p$, la concaténation de $p$ et $q$ notée $p.q$ n'est définie que si le point d'arrivée de $p$ correspond au point de départ de $q$: si* $\text{arr}(p) = \text{dép}(q)$ *alors* $p.q = (\text{base}(p) \cup \text{base}(q), \text{dép}(p), \text{arr}(q))$.

La *connexité* des dessins est une propriété importante. La définition formelle de la connexité est l'occasion pour nous d'introduire la notion d'armature et de voisinage qui, par ailleurs, sont des éléments souvent utiles dans les démonstrations.

L'*armature* d'un dessin $p$ est l'ensemble des points de la grille constituée par $\mathbb{Z}^2$ qui appartiennent à un pixel du dessin. L'armature d'un dessin pointé $q$ est l'armature de sa base.

**Définition 1.2** *Soit $p \in \mathcal{D}$, l'armature de $p$ est notée* $\text{Arm}(p)$ *et est définie par:*

$$\text{Arm}(p) = \bigcup_{\text{pix}(i,j) \in p} \{(i,j), (i+1,j), (i,j+1), (i+1,j+1)\}$$

*Soit $q \in \mathcal{D}_p$, l'armature de $q$ est définie par:*

$$\text{Arm}(q) = \text{Arm}(\text{base}(q))$$

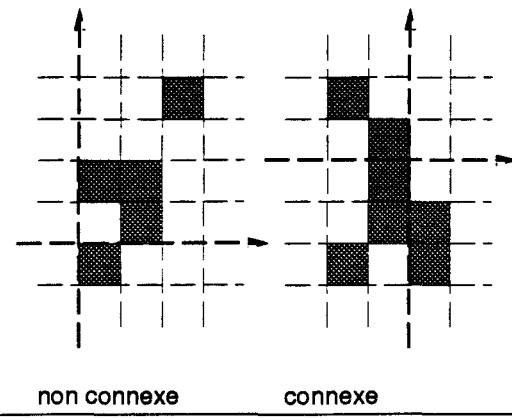



FIG. 1.4 - *Exemples de connexité*

Le *voisinage* d'un pixel $\text{pix}(i,j) \in \text{P}$ est désigné par $\text{Voi}(\text{pix}(i,j))$ et indique l'ensemble des 8 pixels qui l'entourent.

**Définition 1.3** *Le voisinage d'un pixel est défini comme suit :*

$$\text{Voi}(\text{pix}(i,j)) = \left\{ \text{pix}(k,l) \in \text{P} \;\middle|\; \begin{array}{l} i-1 \leq k \leq i+1 \\ j-1 \leq l \leq j+1 \end{array} \quad (k,l) \neq (i,j) \right\}$$

On peut maintenant définir formellement la notion de connexité.

**Définition 1.4** *On dit qu'un dessin $p \in \mathcal{D}$ est connexe si et seulement si :*

$$\forall a \neq b \in p \quad \exists c_1, c_2, \ldots, c_n \in p \quad \textit{tels que :} \begin{cases} c_1 = a \\ c_n = b \\ \forall 1 \leq i < n \quad c_{i+1} \in \text{Voi}(c_i) \end{cases}$$

Un dessin est connexe si de tout pixel du dessin on peut atteindre tout autre pixel du dessin en se déplaçant uniquement d'un pixel aux pixels qui lui sont voisins, c'est-à-dire qui le touchent au moins par un coin (voir Figure 1.4).

On définit aussi la connexité pour les dessins pointés en ajoutant une condition sur la position des points de départ et d'arrivée.

**Définition 1.5** *1. Soit $p \in \mathcal{D}_p$ un dessin pointé non vide ($||p|| \neq 0$), on dit que $p$ est connexe si et seulement si :*

*(a)* $\text{base}(p)$ *est connexe ;*

*(b)* $\text{dép}(p) \in \text{Arm}(p)$ *;*

*(c)* $\text{arr}(p) \in \text{Arm}(p)$.

2. *Soit $q \in \mathcal{D}_p$ un dessin pointé vide ($||q|| = 0$), on dit que $q$ est connexe si et seulement si* $\text{dép}(q) = \text{arr}(q)$.

L'ensemble des dessins connexes est noté $\mathcal{D}^c$ tandis que l'ensemble des dessins pointés connexes est noté $\mathcal{D}_p^c$.

Nous venons de définir des objets graphiques que nous appelons dessins. Mais plus que l'endroit où ils sont situés dans le plan, c'est leur forme qui nous intéresse.

## 1.2 Des figures

On définit une *figure* comme une classe d'équivalence de dessins modulo une translation. Plus formellement, nous donnons tout d'abord la définition d'une *translation* sur les pixels et sur les points de $\mathbb{Z}^2$. Soient $a = (x, y) \in \mathbb{Z}^2$ un point et $(i, j) \in \mathbb{Z}^2$, le translaté du point $a$ de $(i, j)$ est $\tau_{i,j}(a) = (x + i, y + j)$. Soient $b = \text{pix}(x, y) \in \text{P}$ un pixel et $(i, j) \in \mathbb{Z}^2$, le translaté du pixel $b$ de $(i, j)$ et $\tau_{i,j}(b) = \text{pix}(x + i, y + j)$.

Cette notion de translation s'étend naturellement à un ensemble de points ou de pixels, ce qui nous permet de définir la relation d'équivalence $\sim$ sur $\mathcal{D}$ :

$$\forall p, q \in \mathcal{D} \quad p \sim q \Leftrightarrow \exists (i, j) \in \mathbb{Z}^2 \quad \tau_{i,j}(p) = q$$

De même, on définit $\simeq$ sur $\mathcal{D}_p$ :

$$\forall p, q \in \mathcal{D}_p \quad p \simeq q \Leftrightarrow \exists (i, j) \in \mathbb{Z}^2 \text{ tels que } \begin{cases} \tau_{i,j}(\text{base}(p)) = \text{base}(q) \\ \tau_{i,j}(\text{dép}(p)) = \text{dép}(q) \\ \tau_{i,j}(\text{arr}(p)) = \text{arr}(q) \end{cases}$$

Une *figure* est une classe d'équivalence de $\sim$ tandis qu'une *figure pointée* est une classe d'équivalence de $\simeq$. Nous notons $\mathcal{F}$ l'ensemble des figures et $\mathcal{F}_p$ l'ensemble des figures pointées. Soient $p \in \mathcal{D}$ et $q \in \mathcal{D}_p$, on désigne par $[p]_\sim$ la figure contenant $p$ et $[q]_\simeq$ la figure pointée contenant $q$.

Comme nous le verrons par la suite, il est plus aisé de raisonner sur un représentant d'une figure que sur la figure elle-même. Aussi lorsqu'il est clair qu'une propriété est conservée par translation, nous utiliserons cette approche. Par exemple, la connexité est une propriété qui est conservée par translation, aussi, pour montrer qu'une figure est connexe, on considère l'un de ses représentants. De même, pour illustrer une figure par un schéma, nous donnons en fait un de ses représentants et nous omettons les axes.

On note $\mathcal{F}^c$ l'ensemble des figures connexes et $\mathcal{F}_p^c$ l'ensemble des figures pointées connexes.

On peut aisément étendre la notion de concaténation des dessins pointés aux figures pointées.

**Définition 1.6** *Soient $f, g \in \mathcal{F}_p$ deux figures pointées, on définit la concaténation de $f$ et $g$ notée $f.g$ :*

$$\forall f, g \in \mathcal{F}_p \quad f.g = [p.q]_\simeq \quad p \in f, q \in g, p.q \text{ définie}$$

On voit ainsi que, contrairement au cas des dessins, la concaténation de deux figures pointées est toujours définie. En effet, on peut toujours trouver $p$ et $q$ tels que $p.q$ est définie. Ceci vient du fait que si l'on se fixe un point $a \in \mathbb{Z}^2$ il existe toujours, pour une figure pointée $f$, un représentant $p$ tel que $\mathrm{arr}(p) = a$ et un représentant $q$ tel que $\mathrm{dép}(q) = a$. On vérifie ainsi clairement que la concaténation de deux figures est bien unique.

**Proposition 1.7** *L'ensemble des figures pointées muni de la concaténation est un monoïde.*

*Preuve.* On voit facilement que la concaténation est associative. De plus, elle possède bien un élément neutre que nous notons $f_\varepsilon = [(\emptyset, a, a)]_\simeq$ avec $a \in \mathbb{Z}^2$. Nous appelons $f_\varepsilon$ « la figure pointée vide » ou plus simplement la « figure vide » lorsqu'il n'y a pas d'ambiguïté entre figure et figure pointée. □

Soit $F$ un ensemble de figures pointées, on note $F^*$ le plus petit ensemble contenant $F$ et $f_\varepsilon$ et fermé par concaténation.

Dans la suite de cette partie, nous restreignons notre étude aux figures connexes, c'est-à-dire aux ensembles $\mathcal{F}^c$ et $\mathcal{F}_p^c$.

## 1.3 Le monoïde des figures connexes

Nous nous proposons ici de montrer le résultat ci-dessous et d'en déduire un certain nombre de propriétés.

**Théorème 1.8** *L'ensemble des figures pointées connexes muni de la concaténation est un monoïde inversif finiment engendré.*

Afin de montrer ce théorème, nous avons besoin d'un certain nombre de résultats intermédiaires. Notamment, nous devons montrer que la concaténation est bien une opération interne de $\mathcal{F}_p^c$.

**Proposition 1.9** *La concaténation de deux figures pointées connexes est une figure pointée connexe.*

*Preuve.* Si $f = f_\varepsilon$ ou $g = f_\varepsilon$, la propriété est évidente.

Si $f \neq f_\varepsilon$ et $g \neq f_\varepsilon$, nous raisonnons sur deux représentants $p \in f$ et $q \in g$ tels que $p.q$ est défini. On peut passer de tout pixel de $p$ à tout autre pixel de $p$ en n'utilisant que des points de l'armature de $p$. De même pour $q$. Puisque $\mathrm{arr}(p) = \mathrm{dép}(q)$ est un point commun aux deux armatures, $p.q$ est bien connexe. □

On en déduit, avec la Proposition 1.7 :

**Corollaire 1.10** *L'ensemble des figures connexes pointées muni de la concaténation est un monoïde.*

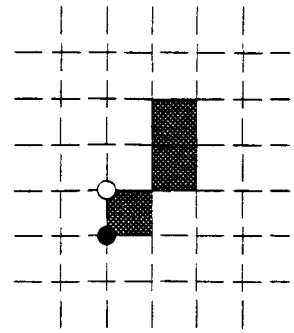


FIG. 1.5 - *Figure pointée connexe ne pouvant être « décomposée »*

On remarque que :

**Fait 1.11** *Tout dessin connexe de poids au moins deux peut être décomposé (au sens de l'union) en deux dessins connexes disjoints non vides.*

À l'opposé, une figure pointée connexe $f$ de poids au moins deux n'est pas toujours la concaténation de deux figures pointées connexes non vides strictement plus petites. Par exemple, la figure de la Figure 1.5 ne comporte pas ce type de décomposition. Notons que cette décomposition existe si l'on peut trouver deux pixels différents l'un ayant le point de départ sur son armature et l'autre le point d'arrivée (ceci nous servira notamment dans la Proposition 3.4). Par contre, on peut toujours trouver trois figures pointées connexes strictement plus petites dont $f$ est la concaténation. Ceci nous est montré par le lemme suivant :

**Lemme 1.12** *Soit $f \in \mathcal{F}_p^c$, une figure pointée connexe. Si $||f|| \geq 2$, il existe trois figures pointées connexes $f_1$, $f_2$ et $f_3$ toutes de poids strictement inférieur à $||f||$ telles que $f = f_1 \cdot f_2 \cdot f_3$.*

*Preuve.* Raisonnons sur un représentant de $f$ :

$$f = [(p, d, a)]_{\simeq}$$

Considérons le dessin $p$. D'après le Fait 1.11, on peut décomposer $p$ en deux dessins connexes disjoints $q_1$ et $q_2$ non vides.

Notons que l'on a $||q_1|| < ||p||$ et $||q_2|| < ||p||$. De plus, il est clair qu'il existe un point $e$ qui est à la fois dans l'armature de $q_1$ et dans l'armature de $q_2$ car $p = q_1 \cup q_2$ est connexe :

$$e \in \mathrm{Arm}(q_1) \cap \mathrm{Arm}(q_2)$$

Nous pouvons distinguer 4 cas selon la position de $d$ et $a$.

1. si $d \in \mathrm{Arm}(q_1)$ et $a \in \mathrm{Arm}(q_2)$ : on peut considérer les figures pointées suivantes :

$$f_1 = [(q_1, d, e)]_{\simeq} \quad f_2 = [(q_2, e, a)]_{\simeq} \quad f_3 = f_e$$

2. si $d \in \mathrm{Arm}(q_2)$ et $a \in \mathrm{Arm}(q_1)$ : cas similaire au précédent.

3. si $d \in \mathrm{Arm}(q_1)$ et $a \in \mathrm{Arm}(q_1)$ : on considère les figures pointées :

$$f_1 = [(q_1, d, e)]_{\simeq} \quad f_2 = [(q_2, e, e)]_{\simeq} \quad f_3 = [(q_1, e, a)]_{\simeq}$$

4. si $d \in \mathrm{Arm}(q_2)$ et $a \in \mathrm{Arm}(q_2)$ : cas similaire au précédent.

On peut donc toujours trouver $f_1$, $f_2$ et $f_3$ de poids strictement inférieur au poids de $f$ et avec $f = f_1.f_2.f_3$. □

Enfin, avant de montrer que $(\mathcal{F}_p^c, .)$ est inversif, nous allons avoir besoin de la définition d'un inverse pour les figures pointées :

**Définition 1.13** *Soit $f = [(p, d, a)]_{\simeq}$ une figure pointée, l'inverse de $f$, noté $f^{-1}$, est la figure $f^{-1} = [(p, a, d)]_{\simeq}$.*

*Preuve.* (Théorème 1.8) Le Corollaire 1.10 nous indique déjà que $(\mathcal{F}_p^c, .)$ est un monoïde. Il nous reste donc à montrer qu'il est finiment engendré et inversif.

Nous montrons tout d'abord qu'il est finiment engendré. Considérons l'ensemble $O$ qui contient toutes les figures pointées connexes de poids 1 :

$$O = \{f \in \mathcal{F}_p^c \mid ||f|| = 1\}$$

Il est clair que $O$ est fini : il ne contient en effet que 16 figures suivant la position du point de départ et du point d'arrivée. Nous montrons que $\mathcal{F}_p^c = O^*$. Pour ceci, il nous suffit que toute figure pointée $f$ de taille au moins 1 soit dans $O^*$ (la figure vide étant par définition dans $O^*$).

On raisonne par récurrence sur le poids de $f$. Si $f$ est de poids 1, la propriété est vraie puisque $f \in O$. Sinon, si $||f|| > 1$, on suppose la propriété vraie pour toute figure de poids strictement inférieur à $||f||$ et on la montre vraie pour $f$. La figure $f$ vérifie les conditions d'application du Lemme 1.12. Aussi, on peut trouver $f_1$, $f_2$ et $f_3$ dans $\mathcal{F}_p^c$ toutes de poids strictement inférieur à celui de $f$ et telles que $f = f_1.f_2.f_3$. Or, par hypothèse de récurrence, on a pour $i \in \{1, 2, 3\}$, $f_i \in O^*$ et donc $f \in O^*$.

Enfin, il nous faut montrer que $(\mathcal{F}_p^c, .)$ est inversif. Pour ceci, il suffit de montrer que pour toute figure pointée $f = [(p, d, a)]_{\simeq}$, $f^{-1} = [(p, a, d)]_{\simeq}$ est la seule figure $g = [(q, a, a')]_{\simeq}$ satisfaisant :

$$f.g.f = f \tag{1.1}$$
$$g.f.g = g \tag{1.2}$$

Ce qui est clair, puisque (1.1) impose que $a' = d$ et $q \subseteq p$ et que (1.2) contraint $p \subseteq q$. Il nous suffit maintenant de constater que l'inverse défini en 1.13 associe à une figure connexe pointée une figure connexe pointée. Ceci nous assure l'existence et l'unicité de l'inverse. □

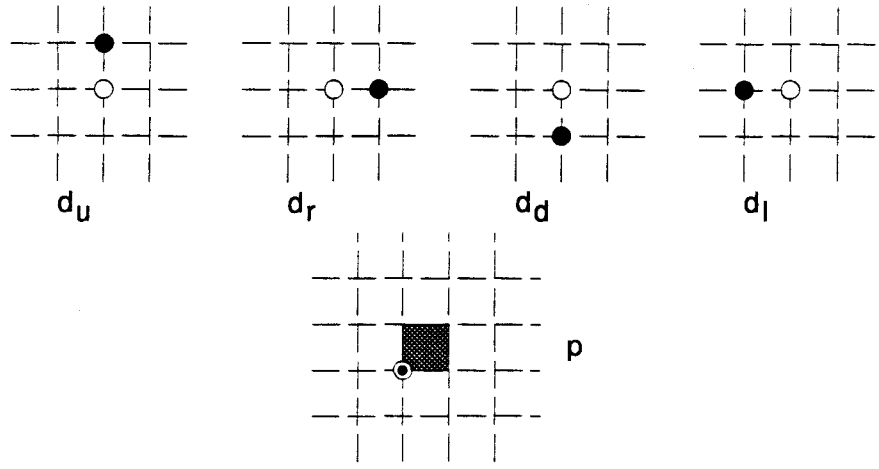



FIG. 1.6 - *Figures engendrant le monoïde* $(\mathcal{F}_p, .)$

Notons que l'ensemble des figures pointées $(\mathcal{F}_p, .)$ est aussi un monoïde inversif finiment engendré. Il est engendré par $A = \{d_u, d_r, d_d, d_l, p\}$ illustré Figure 1.6.

Nous avons vu dans la preuve précédente que le monoïde des figures pointées connexes était engendré par l'ensemble des figures pointées connexes de poids 1 (noté $O$ dans la preuve). Cet ensemble comporte 16 éléments, il nous a semblé intéressant de chercher à savoir s'il n'existe pas un ensemble de figures plus petit qui engendre $\mathcal{F}_p^c$.

**Corollaire 1.14** *Soit* $\mathcal{U} =$ {▀, ▛, ▜, ▟}, *on a* $\mathcal{F}_p^c = \mathcal{U}^*$.

Les symboles du type ▛, ▟,... désignent sans ambiguïté des figures de $\mathcal{F}_p^c$. Par exemple, le symbole ▛ représente la figure $[(\{\text{pix}(0,0)\}, (0,1), (1,1))]_{\simeq}$.

*Preuve.* Pour montrer ce corollaire, il suffit de montrer que toutes les figures pointées connexes de taille 1 peuvent être obtenues par concaténation des figures de $\mathcal{U}$. Nous n'examinons que les cas où le point de départ est dans le coin supérieur gauche, les autres cas étant symétriques. Il y a 4 cas possibles :

1. ▛ est bien dans $\mathcal{U}$
2. ▜ = ▛.▜
3. ▙ = ▛.▜.▟
4. ▚ = ▛.▜.▟.▙

□

Notons que cet ensemble de générateurs de $\mathcal{F}_p^c$ est de taille minimale. Il n'existe pas d'ensemble de taille strictement inférieure à 4 qui engendre $\mathcal{F}_p^c$.

Comme dans tout monoïde, nous pouvons distinguer les parties reconnaissables et les parties rationnelles de $\mathcal{F}_p^c$ [Ber79].

**Définition 1.15** *Une partie $F$ de $\mathcal{F}_p^c$ est reconnaissable s'il existe un monoïde fini $M$, un homomorphisme $h$ de $\mathcal{F}_p^c$ dans $M$ et une partie $N$ de $M$ tels que $F = h^{-1}(N)$. Nous notons* $\mathrm{Rec}(\mathcal{F}_p^c)$ *l'ensemble des parties reconnaissables de $\mathcal{F}_p^c$.*

**Définition 1.16** *La classe des parties rationnelles de $\mathcal{F}_p^c$, notée* $\mathrm{Rat}(\mathcal{F}_p^c)$, *est la plus petite classe des parties de $\mathcal{F}_p^c$ contenant les parties finies et close par union, concaténation et par étoile.*

On sait déjà grâce à MacKnight (1964) (voir proposition 2.4 p. 57 de [Ber79]), que l'on a $\mathrm{Rec}(\mathcal{F}_p^c) \subseteq \mathrm{Rat}(\mathcal{F}_p^c)$. Néanmoins $\mathcal{F}_p^c$ n'est pas « de Kleene » [Sak87, PS90], c'est-à-dire que $\mathrm{Rat}(\mathcal{F}_p^c) \neq \mathrm{Rec}(\mathcal{F}_p^c)$. Pour s'en convaincre, il suffit de considérer l'ensemble $L = \{\blacksquare\}^*$. L'ensemble $L$ est évidemment rationnel mais il est facile de montrer qu'il n'est pas reconnaissable. Nous verrons plus loin comment montrer ceci à l'aide des langages de mots de figures.

Les parties finies étant, par définition, rationnelles, on montre qu'elles sont également reconnaissables.

**Proposition 1.17** *Les parties finies de $\mathcal{F}_p^c$ sont rationnelles et reconnaissables.*

*Preuve.* Les reconnaissables étant clos par union, il nous suffit de montrer qu'un ensemble contenant une unique figure est reconnaissable (l'ensemble vide étant reconnaissable).

Soit $L = \{f\}$ avec $f \in \mathcal{F}_p^c$. Nous montrons que $L$ est reconnaissable en montrant que le nombre de classes d'équivalence de la congruence à droite induite par $L$ est fini (voir proposition 12.1 p. 68 de [Eil74]). Notons, pour $g \in \mathcal{F}_p^c$, $L/g = \{m \in \mathcal{F}_p^c \mid g.m \in L\}$. Deux figures $g$ et $g'$ de $\mathcal{F}_p^c$ sont en relation par la congruence si et seulement si $L/g = L/g'$. Remarquons que si $||g|| > ||f||$, alors $L/g = \emptyset$. Le nombre de figures connexes de poids inférieur ou égal à $f$ étant fini, le nombre de classes d'équivalence est, a fortiori, fini. □

On dit alors que $\mathcal{F}_p^c$ est « finitely recognizable » [dLV92]. Le fait que l'ensemble des figures pointées connexes est un monoïde finiment engendré nous pousse à nous interroger sur ses systèmes générateurs possibles ; c'est-à-dire représenter $\mathcal{F}_p^c$ par le biais d'un monoïde libre engendré par un alphabet fini [Sak81]. Nous consacrons la suite de cette partie à l'étude d'un système générateur de ce monoïde.

## 1.4 Des mots de figures

Nous cherchons donc, en étudiant un système générateur de $(\mathcal{F}_p^c, .)$, à modéliser les figures, qui sont de manière intrinsèque des objets à deux dimensions, à l'aide de mots. Il est en effet souvent malaisé de travailler sur des objets à deux dimensions alors que les mots sont des objets linéaires que nous savons bien manipuler.

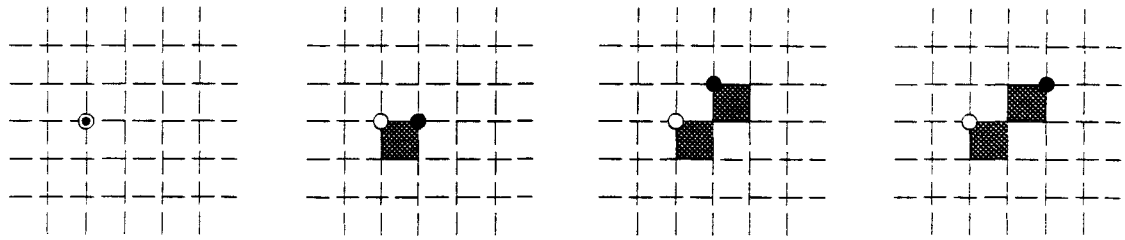



FIG. 1.7 - *Construction de* FigP($rur$)

Ici, cette approche nous est facilitée par le fait que $(\mathcal{F}_p^c, .)$ est un monoïde finiment engendré. Ainsi, nous avons vu dans le Corollaire 1.14 qu'il était engendré par l'ensemble $\mathcal{U}$ composé de 4 figures. Nous nous appuyons donc sur ce résultat pour construire un système générateur du monoïde.

**Proposition 1.18** *Soient l'alphabet* $\Pi = \{u, r, d, l\}$ *et l'homomorphisme* $\mathrm{FigP} : \Pi^* \to \mathcal{F}_p^c$ *défini par :*

$\mathrm{FigP}(u) =$ 
$\mathrm{FigP}(r) =$ 
$\mathrm{FigP}(d) =$ 
$\mathrm{FigP}(l) =$ 

*Le couple* $(\Pi, \mathrm{FigP})$ *est un système générateur de* $(\mathcal{F}_p^c, .)$.

*Preuve.* Immédiate d'après le Corollaire 1.14. □

Par exemple, on peut voir Figure 1.7, la construction pas-à-pas de la figure décrite par $rur$, à titre d'exemple le mot $ulldull$ décrit la figure pointée illustrée en 1.8. Le fait que $\mathcal{F}_p^c$ est inversif implique qu'il existe, pour une figure pointée non vide, une infinité de mots qui lui correspondent. En effet, pour une figure pointée $f$ non vide, si l'on considère les mots $\omega, \omega' \in \Pi^*$ tels que $\mathrm{FigP}(\omega) = f$ et $\mathrm{FigP}(\omega') = f^{-1}$, le mot $\omega\omega'\omega$ décrit également $f$, ainsi que les mots $(\omega\omega')^n\omega$ avec $n \in \mathbb{N}$. Nous notons $\equiv$ la congruence associée à ce système générateur :

**Définition 1.19** *La congruence associée au système générateur* $(\Pi, \mathrm{FigP})$, *notée* $\equiv$, *est définie par :*

$$\forall \omega, \omega' \in \Pi^* \quad \omega \equiv \omega' \Leftrightarrow \mathrm{FigP}(\omega) = \mathrm{FigP}(\omega')$$

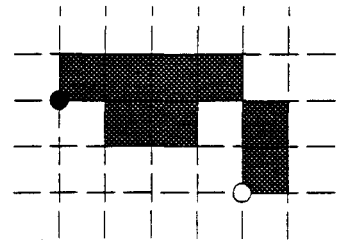


FIG. 1.8 - *Figure pointée décrite par uulldull*

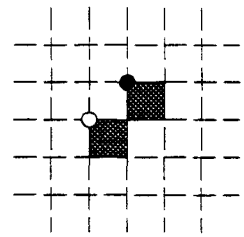


FIG. 1.9 - *Figure f*

Notons qu'ainsi nous pouvons considérer le monoïde $\mathcal{F}_p^c$ comme le quotient de $\Pi^*$ par la congruence $\equiv$. Pour décrire une figure, il est donc intéressant de prendre un mot le plus court possible, mais trouver ces « mots minimaux » est un problème difficile comme nous le verrons Chapitre 3. Nous nous proposons donc d'étudier les classes d'équivalence de cette congruence $\equiv$.

**Proposition 1.20** *Les classes d'équivalence de* $\equiv$ *sont des langages rationnels de* $\Pi^*$.

*Preuve.* Formellement, on veut montrer que l'on a, pour tout mot $\omega$ sur $\Pi$ : $[\omega]_\equiv \in \mathrm{Rat}(\Pi^*)$. Ceci est clair grâce à la Proposition 1.17. En effet, il suffit de constater que $[\omega]_\equiv$ est l'image par l'homomorphisme inverse $\mathrm{FigP}^{-1}$ de l'ensemble fini $\{\mathrm{FigP}(\omega)\}$. □

Cette propriété des classes d'équivalence de $\equiv$ a pour conséquence directe que pour tout ensemble de figures pointées $F \subseteq \mathcal{F}_p^c$ engendré par un langage algébrique $A$ sur $\Pi$ (c'est-à-dire que l'on a $A \in \mathrm{Alg}(\Pi^*)$ et $F = \mathrm{FigP}(A)$), le problème de l'appartenance à $F$ est décidable. Pour décider si une figure $f$ appartient à $F$, on est ramené à décider de la vacuité de $A \cap \mathrm{FigP}^{-1}(f)$. En effet, il est facile de voir que l'on peut construire effectivement, pour toute figure pointée $f \in \mathcal{F}_p^c$, un automate qui reconnaît $\mathrm{FigP}^{-1}(f)$.

Par exemple, pour la figure $f$ illustrée Figure 1.9, l'automate reconnaissant les mots de figures la décrivant est donné Figure 1.10.

Enfin, on peut, à l'aide de ce système générateur, caractériser les parties rationnelles et les parties reconnaissables de $\mathcal{F}_p^c$ :

**Fait 1.21** *1.* $F \in \mathrm{Rat}(\mathcal{F}_p^c) \Leftrightarrow \exists L \in \mathrm{Rat}(\Pi^*) \quad \mathrm{FigP}(L) = F$ ;

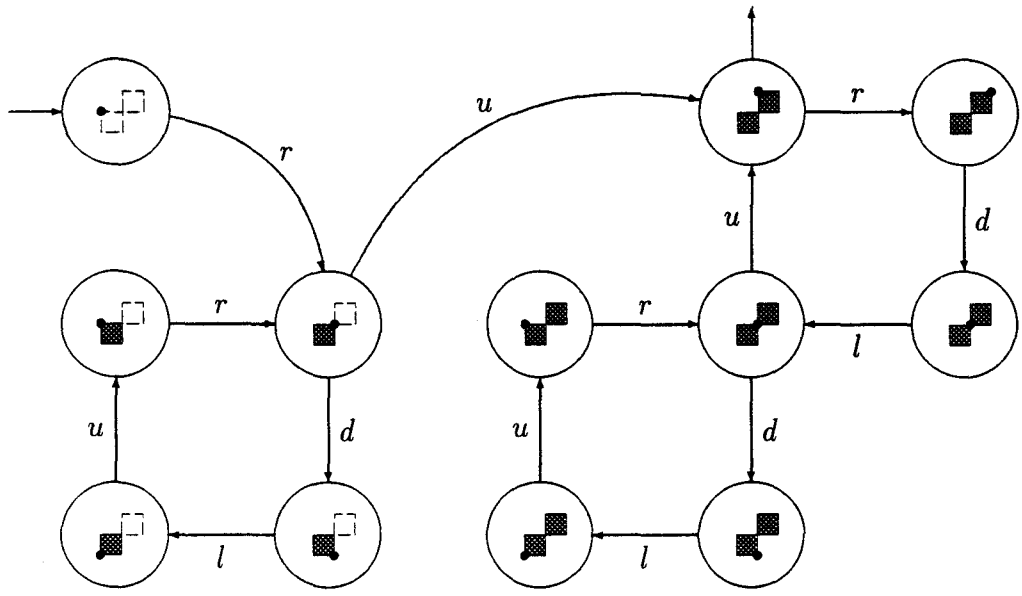



FIG. 1.10 - *Automate reconnaissant* $\mathrm{FigP}^{-1}(f)$

2. $F \in \mathrm{Rec}(\mathcal{F}_p^c) \Leftrightarrow \mathrm{FigP}^{-1}(F) \in \mathrm{Rat}(\Pi^*)$.

On peut maintenant facilement montrer que $\mathcal{F}_p^c$ n'est pas un monoïde de Kleene (voir page 12 après la Définition 1.16) en montrant que le langage $L = \{\blacksquare\}^*$ n'est pas reconnaissable. Si $L$ est reconnaissable, il en est de même pour $\mathrm{FigP}^{-1}(L)$. On utilise le lemme de l'étoile sur $K = \mathrm{FigP}^{-1}(L) \cap r^*dl^*ur^*$. On considère un mot de la forme $r^n dl^m ur^k$ avec $m$ supérieur au nombre d'états, on a forcément $m \leq n$ et $m = k$ or, il existe un mot de la forme $r^n dl^{m'} ur^k$ avec $m' > m$ qui appartient à $K$, toutefois ce mot ne décrit pas une figure de $L$, d'où contradiction : $L$ n'est pas reconnaissable.

Cette méthode de représentation de figures à l'aide d'un monoïde libre a déjà été appliquée par H.A. Maurer, G. Rozenberg et E. Welzl dans [MRW82] pour représenter, par des mots, des figures composées de segments unitaires. On peut en effet interpréter les mots sur $\Pi$ comme une série de commandes données à un traceur. La lettre $u$ provoque un déplacement unitaire vers le haut (up), la lettre $r$ un déplacement unitaire vers la droite (right), la lettre $d$ vers le bas (down) et $l$ vers la gauche (left). À chaque déplacement, on allume le pixel se trouvant à droite du déplacement.

Dans l'étude de la représentation proposée par H.A. Maurer et al., P. Séébold et K. Slowinski donnent dans [SS91] un système de réécriture qui génère tous les mots qui décrivent la même figure. En fait, la relation induite par le système de réécriture est la congruence associée au système générateur qu'ils utilisent. Nous montrons dans le chapitre suivant que des systèmes de réécriture similaires peuvent être utilisés pour des systèmes générateurs de monoïdes inversifs.

# Références

[Ber79] J. Berstel. *Tranductions and Context-Free Languages.* Teubner Studienbücher, Stuttgard, 1979.

[dLV92] A. de Luca and S. Varricchio. On finitely recognizable semigroups. *Acta Informatica*, 29(5):483–498, 1992.

[Eil74] S. Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, New York, 1974.

[LRS97] M. Latteux, D. Robilliard, and D. Simplot. Figures composées de pixels et monoïde inversif. In *Proc. Journées Montoises d'Informatique Théorique*, volume 4 of *Bulletin of the Belgian Mathematical Society - Simon Stevin*, pages 89–111, Mons, Belgium, 1994, 1997.

[MRW82] H. A. Maurer, G. Rozenberg, and E. Welzl. Using string languages to describe picture languages. *Information and Control*, 54(3):155–185, 1982.

[PS90] M. Pelletier and J. Sakarovitch. Easy multiplication II – extensions of rational semigroups. *Information and Computation*, 88(1):18–59, 1990.

[Sak81] J. Sakarovitch. Description des monoïdes de type fini. *Journal of Information Processing and Cybernetics EIK*, 17(8):417–434, 1981.

[Sak87] J. Sakarovitch. Easy multiplication I – the realm of Kleene's theorem. *Information and Computation*, 74(3):173–197, 1987.

[SS91] P. Séébold and K. Slowinski. The shortest way to draw a connected picture. *Computer Graphics Forum*, 10(4):319–327, 1991.

# Chapitre 2

# Équivalence des descripteurs dans un monoïde inversif

La relation qui relie tous les mots qui décrivent la même figure est une congruence. Dans ce chapitre, nous donnons un système de réécriture qui engendre cette congruence pour tout système générateur d'un monoïde inversif vérifiant une certaine condition [LRS97].

## 2.1 Étude de la congruence associée à un système générateur d'un monoïde inversif

Soit $M$ un monoïde admettant un système générateur $(\Sigma, h)$. Nous notons $X$ l'ensemble « généré » par $\Sigma$ : $X = h(\Sigma)$. On a alors $M = X^*$ – remarquons qu'ici $\Sigma$ peut être infini. Nous notons $\lambda$ l'élément neutre de la concaténation dans $M$. Nous notons $\asymp$ la congruence associée à $(\Sigma, h)$. Nous cherchons un système de réécriture $S$ qui « engendre » la congruence $\asymp$, c'est-à-dire tel que :

$$\forall \omega, \omega' \in \Sigma^* \quad \omega \asymp \omega' \Leftrightarrow \omega \xrightarrow[S]{*} \omega' \text{ et } \omega' \xrightarrow[S]{*} \omega$$

**Définition 2.1** *On dit qu'un élement $x$ de $M$ est un idempotent si et seulement si $x.x = x$. On note $I_M$ l'ensemble des idempotents de $M$.*

Dans le cas où $M$ est inversif, on a les propriétés suivantes :

1. $\forall x \in I_M \quad x = x^{-1}$ ;
2. $\forall x \in M \quad x.x^{-1} \in I_M$ ;
3. $\forall x, y \in I_M \quad x.y = y.x \in I_M$.

On supposera dans la suite que $M$ est inversif. Dans le monoïde $M$, nous disposons d'un inverse. Il est intéressant de pouvoir bénéficier d'un outil similaire dans le monoïde libre qu'on lui associe.

**Définition 2.2** *Une fonction inverse pour* $(\Sigma, h)$ *est une fonction* Inv *de* $\Sigma$ *dans* $\Sigma^*$ *telle que pour toute lettre* $a \in \Sigma$, *on a* $h(\mathrm{Inv}(a)) = h(a)^{-1}$. *On étend cette fonction sur les mots afin d'en faire un anti-morphisme de* $\Sigma^*$ *dans* $\Sigma^*$, *pour tout* $\omega \in \Sigma^*$ :

1. *si* $\omega = \varepsilon$, $\mathrm{Inv}(\omega) = \varepsilon$ ;
2. *si* $\omega = \omega' a$ *avec* $\omega \in \Sigma^*$ *et* $a \in \Sigma$, $\mathrm{Inv}(\omega) = \mathrm{Inv}(a)\mathrm{Inv}(\omega')$.

*On note aussi* $\mathrm{Inv}(\omega) = \overline{\omega}$.

On déduit immédiatement de cette définition les propriétés suivantes :

**Fait 2.3** 1. $\forall \alpha \in \Sigma^* \quad h(\overline{\alpha}) = h(\alpha)^{-1}$ ;

2. $\forall \alpha, \beta \in \Sigma^* \quad \overline{\alpha\beta} = \overline{\beta}\overline{\alpha}$.

On considère une fonction inverse et une première approche du problème nous donne le système de réécriture $R$ défini ci-dessous :

**Définition 2.4** *Soient* $M$ *un monoïde inversif,* $(\Sigma, h)$ *un système générateur de* $M$ *et* Inv *une fonction inverse de* $(\Sigma, h)$, *on définit le système de réécriture* $R$ :

$$R = R_1 \cup R_2$$

$$\text{avec} \quad \begin{cases} R_1 = \{a \leftrightarrow a\overline{a}a \mid a \in \Sigma\} \\ R_2 = \{\alpha \leftrightarrow \beta \mid \alpha, \beta \in \Sigma^* \text{ et } h(\alpha) = h(\beta) \in I_M\} \end{cases}$$

**Proposition 2.5** *La relation* $\underset{R}{\overset{*}{\to}}$ *coïncide avec la congruence* $\asymp$.

*Preuve.* On déduit d'abord de $R$ l'ensemble de règles $R_3$ :

$$R_3 = \{\alpha \leftrightarrow \alpha\overline{\alpha}\alpha \mid \alpha \in \Sigma^*\}$$

On montre par récurrence sur la longueur de $\alpha$ que l'on peut déduire la règle $\alpha \leftrightarrow \alpha\overline{\alpha}\alpha$ de $R$. Si $\alpha = \varepsilon$, on a $\alpha = \alpha\overline{\alpha}\alpha$. Si $|\alpha| = 1$, on utilise les règles de $R_1$. Sinon, on suppose la propriété vraie pour tout mot de longueur strictement inférieure à celle de $\alpha$ et on la montre vraie pour $\alpha$. On a $\alpha = \alpha' a$ avec $\alpha' \in \Sigma^*$ et $a \in \Sigma$. On a :

$$\alpha = \alpha' a \underset{\text{réc.}}{\leftrightarrow} \alpha'\overline{\alpha'}\alpha' a \underset{R_1}{\leftrightarrow} \alpha'\overline{\alpha'}\alpha' a\overline{a}a \underset{R_2}{\leftrightarrow} \alpha' a\overline{a}\overline{\alpha'}\alpha' a = \alpha\overline{\alpha}\alpha$$

L'application des règles de $R_2$ nous permet en effet d'intervertir deux mots représentant des idempotents. Ici, en l'occurrence, il s'agit de $\overline{\alpha'}\alpha'$ et $a\overline{a}$ que l'on permute.

Une fois cet ensemble de règles déduit, nous montrons que pour deux mots $\omega, \omega' \in \Sigma^*$, tels que $\omega \asymp \omega'$, on a $\omega \overset{*}{\underset{R}{\leftrightarrow}} \omega'$ :

$$\omega \underset{R_3}{\leftrightarrow} \omega\overline{\omega}\omega \underset{R_2}{\leftrightarrow} \omega'\overline{\omega'}\omega$$

Les mots $\overline{\omega'}\omega$ et $\overline{\omega'}\omega'$ représentent tous deux le même idempotent, on peut donc appliquer la règle $R_2$ :

$$\omega'\overline{\omega'}\omega \underset{R_2}{\leftrightarrow} \omega'\overline{\omega'}\omega' \underset{R_3}{\leftrightarrow} \omega'$$

Ceci nous assure que si deux mots représentent le même élément du monoïde $M$ alors on peut les dériver l'un de l'autre par $R$. De plus, les règles de $R$ nous indiquent qu'à chaque étape de dérivation, on ne change pas l'élément décrit. □

Néanmoins, ce résultat n'est pas étonnant puisque nous mettons dans $R$ énormément de règles, notamment la réécriture de tous les mots représentant le même idempotent. Nous allons montrer que sous certaines conditions, nous pouvons affiner ce résultat.

**Définition 2.6** *Soient $M$ un monoïde inversif, $(\Sigma, h)$ un système générateur de $M$ et* Inv *une fonction inverse de $(\Sigma, h)$, on définit le système de réécriture $S$ :*

$$S = S_1 \cup S_2 \cup S_3$$

$$\text{avec} \quad \begin{cases} S_1 = \{a \leftrightarrow a\overline{a}a \mid a \in \Sigma\} \\ S_2 = \{\alpha\alpha \rightarrow \alpha \mid h(\alpha) \in I_M\} \\ S_3 = \{\alpha \leftrightarrow \overline{\alpha} \mid h(\alpha) \in I_M\} \end{cases}$$

**Lemme 2.7** *Du système de réécriture $S$, on peut déduire les ensembles de règles :*

$$\begin{aligned} T_1 &= \{\alpha\beta \leftrightarrow \beta\alpha \mid h(\alpha), h(\beta) \in I_M\} \\ T_2 &= \{\alpha \leftrightarrow \alpha\overline{\alpha}\alpha \mid \alpha \in \Sigma^*\} \\ S_2' &= \{\alpha \rightarrow \alpha\alpha \mid h(\alpha) \in I_M\} \end{aligned}$$

*Preuve.* Les règles de $T_1$ sont dérivées des règles de $S_3$ :

$$\alpha\beta \underset{S_3}{\leftrightarrow} \overline{\alpha\beta} = \overline{\beta}\overline{\alpha} \overset{2}{\underset{S_3}{\leftrightarrow}} \beta\alpha$$

On montre que l'on peut déduire les règles de $T_2$ par récurrence sur la taille de $\alpha$. Si $\alpha = \varepsilon$, $\alpha = \alpha\overline{\alpha}\alpha$. Si $|\alpha| = 1$, on utilise les règles de $S_1$. Sinon, on suppose la propriété vraie pour tous les mots de taille strictement inférieure à celle de $\alpha$ et on la montre vraie pour $\alpha$. On a $\alpha = \alpha' a$ avec $\alpha' \in \Sigma^*$ et $a \in \Sigma$ :

$$\alpha = \alpha' a \underset{\text{réc.}}{\leftrightarrow} \alpha'\overline{\alpha'}\alpha' a \underset{S_1}{\leftrightarrow} \alpha'\overline{\alpha'}\alpha' a\overline{a}a \underset{T_1}{\leftrightarrow} \alpha' a\overline{a}\overline{\alpha'}\alpha' a = \alpha\overline{\alpha}\alpha$$

Les règles de $S_2'$ sont les règles réciproques de celles de $S_2$ et sont dérivées de la manière suivante :

$$\alpha \underset{T_2}{\leftrightarrow} \alpha\overline{\alpha}\alpha \underset{S_3}{\leftrightarrow} \alpha\alpha\alpha \underset{S_2}{\rightarrow} \alpha\alpha$$

□

La clôture transitive du système $S$ ne correspond pas toujours à la congruence $\asymp$. Par exemple, s'il existe un diviseur de l'élément neutre autre que $\lambda$, on a un mot non vide $\omega$ équivalent à $\varepsilon$, or aucune règle de $S$ ne permet de passer du mot vide à un mot non vide.

Notons que même si l'élement neutre n'a pas de diviseur, $S$ peut ne pas être complet. Considérons le sous-monoïde inversif généré par l'ensemble de figures $A$ :

$A =$ 

On représente les figures de $A^*$ grâce au système générateur $(X, h)$ avec l'alphabet $X = \{a, a', b, b'\}$ et $h$ qui fait correspondre les lettres de $X$ avec les figures de $A$ dans l'ordre ci-dessus. Les mots $bb$ et $aa'bb$ représentent tous deux la figure $f$ :

$f =$ 

Si l'on construit une fonction inverse Inv, on voit que $\mathrm{Inv}(b)$ et $\mathrm{Inv}(b')$ ne peuvent contenir que des $b$ et des $b'$. Toutes les règles de $S$ qui ne contiennent que des $b$ et des $b'$ d'un côté de la règle ne contiennent donc que des $b$ et des $b'$ de l'autre. On ne peut donc pas dériver $aa'bb$ à partir de $bb$.

**Théorème 2.8** *Soient $M$ un monoïde inversif, $(\Sigma, h)$ un système générateur de $M$,* Inv *une fonction inverse de $(\Sigma, h)$ et le système de réécriture $S$ défini ci-dessus. La relation $\overset{*}{\underset{S}{\rightarrow}}$ coïncide avec la congruence $\asymp$ associée à $(\Sigma, h)$ si et seulement si :*

$$\forall \alpha \in \Sigma^*, a \in \Sigma \quad h(\alpha) = h(a\overline{a}\alpha) \Rightarrow \alpha \overset{*}{\underset{S}{\leftrightarrow}} a\overline{a}\alpha \qquad (P)$$

Afin d'obtenir ce théorème, nous montrons les résultats intermédiaires suivants :

**Lemme 2.9** *Si la condition $(P)$ est respectée, on a alors :*

$$\forall \alpha, \beta \in \Sigma^* \quad h(\alpha) = h(\beta\overline{\beta}\alpha) \Rightarrow \alpha \overset{*}{\underset{S}{\leftrightarrow}} \beta\overline{\beta}\alpha$$

*Preuve.* Nous montrons cette propriété par récurrence sur la taille de $\beta$. Si $\beta = \varepsilon$, la propriété est immédiate. Si $|\beta| = 1$, elle se déduit directement de $(P)$. Sinon on suppose que la propriété est vraie pour tous les mots de longueur strictement inférieure à la longueur de $\beta$ et on la montre vraie pour $\beta$. On a $\beta = a\beta'$ avec $\beta' \in \Sigma^*$ et $a \in \Sigma$.

Notons que l'on a $h(a\overline{a}\alpha) = h(\alpha)$ et $h(\beta'\overline{\beta'}\overline{a}\alpha) = h(\overline{a}\alpha)$. En effet :

$$h(\alpha) = h(\beta\overline{\beta}\alpha) = h(a\beta'\overline{\beta}\alpha) = h(a).h(\beta'\overline{\beta}\alpha) = h(a).h(\overline{a}).h(a).h(\beta'\overline{\beta}\alpha) = h(a\overline{a}a\beta'\overline{\beta}\alpha) = h(a\overline{a}\alpha)$$

et

$$h(\overline{a}\alpha) = h(\overline{a}).h(\alpha) = h(\overline{a}).h(a\beta'\overline{\beta'}\overline{a}\alpha) = h(\overline{a}a).h(\beta'\overline{\beta'}).h(\overline{a}\alpha) = h(\beta'\overline{\beta'}).h(\overline{a}a).h(\overline{a}\alpha) = h(\beta'\overline{\beta'}).h(\overline{a}).h(a).h(\overline{a}).h(\alpha) = h(\beta'\overline{\beta'}\overline{a}\alpha)$$

On a donc, d'après $(P)$ :

$$\alpha \underset{S}{\overset{*}{\leftrightarrow}} a\overline{a}\alpha \underset{\text{réc.}}{\leftrightarrow} a\beta'\overline{\beta'}\overline{a}\alpha = \beta\overline{\beta}\alpha$$

□

**Lemme 2.10** *Si la condition $(P)$ est respectée, on a alors :*

$$\forall \alpha, \beta \in \Sigma^* \quad h(\alpha) = h(\beta\alpha) \text{ et } h(\beta) \in I_M \Rightarrow \alpha \underset{S}{\overset{*}{\leftrightarrow}} \beta\alpha$$

*Preuve.* Comme $\beta$ décrit un idempotent, on a :

$$h(\beta\overline{\beta}\alpha) = h(\beta).h(\overline{\beta}).h(\alpha) = h(\beta).h(\beta)^{-1}.h(\alpha) = h(\beta).h(\beta).h(\alpha) = h(\beta).h(\alpha) = h(\beta\alpha) = h(\alpha)$$

On a donc (d'après le Lemme 2.9) :

$$\alpha \underset{S}{\overset{*}{\leftrightarrow}} \beta\overline{\beta}\alpha \underset{S_3}{\leftrightarrow} \beta\beta\alpha \underset{S_2}{\rightarrow} \beta\alpha$$

et dans l'autre sens :

$$\beta\alpha \underset{T_2}{\leftrightarrow} \beta\overline{\beta}\beta\alpha \underset{S_3}{\leftrightarrow} \beta\overline{\beta}\ \overline{\beta}\alpha \underset{S_2}{\rightarrow} \beta\overline{\beta}\alpha \underset{S}{\overset{*}{\leftrightarrow}} \alpha$$

□

**Lemme 2.11** *Si la condition $(P)$ est respectée, on a :*

$$\forall \alpha, \beta \in \Sigma^* \quad h(\alpha) = h(\alpha\beta) \text{ et } h(\beta) \in I_M \Rightarrow \alpha \underset{S}{\overset{*}{\leftrightarrow}} \alpha\beta$$

*Preuve.* Il suffit de noter que si $h(\alpha) = h(\alpha\beta)$ alors on a $h(\overline{\alpha}) = h(\overline{\beta}\overline{\alpha})$. Ceci nous permet d'appliquer le Lemme 2.10 :

$$\alpha \underset{T_2}{\leftrightarrow} \alpha\overline{\alpha}\alpha \underset{S}{\overset{*}{\leftrightarrow}} \alpha\overline{\beta}\overline{\alpha}\alpha \underset{T_1}{\leftrightarrow} \alpha\overline{\alpha}\alpha\overline{\beta} \underset{T_2}{\leftrightarrow} \alpha\overline{\beta} \underset{S_3}{\leftrightarrow} \alpha\beta$$

□

*Preuve.* (Théorème 2.8) Si la clôture transitive du système de réécriture correspond à la congruence $\asymp$, il est trivial que $\alpha$ se dérive en $a\bar{a}\alpha$ s'ils décrivent le même élément. Pour l'autre sens, les règles de $S$ nous assurant qu'à chaque réécriture, l'élément de $M$ décrit n'est pas modifié, il nous suffit de montrer que:

$$\forall \omega, \omega' \in \Sigma^* \quad h(\omega) = h(\omega') \Rightarrow \omega \overset{*}{\underset{S}{\leftrightarrow}} \omega'$$

Si $h(\omega) = h(\omega')$, alors on a:

$$\begin{aligned} &h(\omega\bar{\omega}\omega') = h(\omega) = h(\omega') \\ &h(\omega\bar{\omega}) \in I_M \\ &h(\bar{\omega}\omega') \in I_M \end{aligned}$$

On a donc, en appliquant les Lemmes 2.10 et 2.11:

$$\omega \overset{*}{\underset{S}{\leftrightarrow}} \omega\bar{\omega}\omega' \overset{*}{\underset{S}{\leftrightarrow}} \omega'$$

□

La relation induite par le système de réécriture $S$ défini en 2.6 coïncide donc bien avec la congruence $\asymp$ si la condition $(P)$ est respectée. La propriété $(P)$ est équivalente à la suivante:

$$\forall \alpha \in \Sigma^*, a \in \Sigma \quad h(\alpha) = h(a\bar{a}\alpha) \Rightarrow \exists \beta, \gamma \in \Sigma^* \quad \alpha \overset{*}{\underset{S}{\leftrightarrow}} \beta a \gamma \quad \text{avec } h(\beta) \in I_M \qquad (P')$$

En effet, on peut alors dériver $\alpha$ de la manière suivante:

$$\alpha \overset{*}{\underset{S}{\leftrightarrow}} \beta a\gamma \underset{S_1}{\leftrightarrow} \beta a\bar{a}a\gamma \underset{T_1}{\leftrightarrow} a\bar{a}\beta a\gamma \overset{*}{\underset{S}{\leftrightarrow}} a\bar{a}\alpha$$

Cette condition signifie que l'on peut faire apparaître un $a$ dans $\alpha$ au « bon endroit », c'est-à-dire après un idempotent. En particulier, $(P')$ est vraie si:

$$\begin{aligned} &\forall \alpha \in \Sigma^*, a \in \Sigma \quad h(\alpha) = h(a\bar{a}\alpha) \Rightarrow \exists \beta, \gamma \in \Sigma^* \\ &\alpha\bar{a}\alpha = \beta a\gamma \quad \text{avec } h(\beta) \in I_M \end{aligned} \qquad (C)$$

Notons que la réciproque n'est pas vraie: $(P') \not\Rightarrow (C)$.

Nous allons maintenant appliquer ce résultat sur les systèmes de réécriture à notre système générateur du monoïde des figures pointées connexes qui vérifie cette dernière condition.

## 2.2 Équivalence entre mots de figures

Nous pouvons utiliser le système de réécriture $S$ pour notre système générateur de $\mathcal{F}_p^c$. Il nous suffit de définir la fonction inverse que l'on associe à $(\Pi, \mathrm{FigP})$.

**Définition 2.12** *La fonction inverse* Inv *que nous utilisons est définie par:*

$$\mathrm{Inv}(u) = rdl \quad \mathrm{Inv}(r) = dlu \quad \mathrm{Inv}(d) = lur \quad \mathrm{Inv}(l) = urd$$

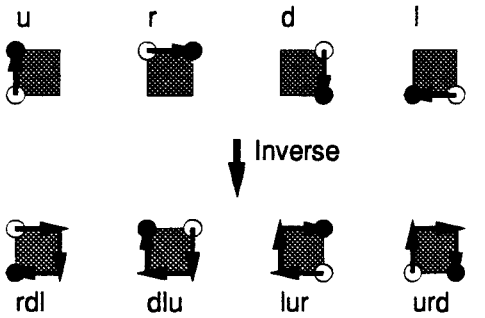



FIG. 2.1 - *Inversion des lettres de* $\Pi$

Nous donnons une illustration de l'inverse des lettres de $\Pi$ en Figure 2.1.

Afin de pouvoir utiliser le système de réécriture $S$, nous devons montrer que notre système générateur $(\Pi, \text{FigP})$ avec la fonction Inv définie ci-dessus vérifie la condition $(P)$ du Théorème 2.8. Nous pouvons aisément identifier les idempotents de $\mathcal{F}_p^c$ :

**Fait 2.13** *Soit* $f = [(p, d, a)]_{\simeq} \in \mathcal{F}_p^c$, $f \in I_{\mathcal{F}_p^c}$ *si et seulement si* $d = a$.

Lorsque nous parlerons des idempotents de $\mathcal{F}_p^c$, nous utiliserons le terme *boucles* puisque ce sont les figures pour lesquelles les points de départ et d'arrivée sont confondus. Nous notons $\mathcal{B}(= I_{\mathcal{F}_p^c})$ l'ensemble des boucles.

**Proposition 2.14** *Le monoïde* $\mathcal{F}_p^c$, *avec le système générateur* $(\Pi, \text{FigP})$ *et la fonction inverse* Inv *définie ci-dessus respecte la condition* $(P)$.

*Preuve.* On montre que la condition $(C)$ est vérifiée. Considérons $\alpha \in \Pi^*$ et $a \in \Pi$ avec $f = \text{FigP}(\alpha) = \text{FigP}(a\bar{a}\alpha)$. Raisonnons sur un représentant de $p$ de $f$ tel que dép$(p) = (0, 0)$. Supposons que $a = r$, les autres cas étant traités de manière similaire. On est sûr que $\alpha \neq \varepsilon$ car $\varepsilon$ est le seul représentant de $f_\varepsilon$. On peut donc noter $\alpha = a_1 \dots a_n$ avec pour tout $1 \leq i \leq n$, $a_i \in \Pi$.

À chaque lettre de $\alpha$, on peut associer le pixel de $p$ qu'elle « allume ». Pour ceci nous définissons les suites $(x_i)_{i\in[0,n]}$, $(y_i)_{i\in[0,n]}$ et $(p_i)_{i\in[1,n]}$ :

$$x_0 = 0$$
$$\forall 1 \leq i \leq n \quad x_k = \begin{cases} x_{k-1} + 1 & \text{si } a_k = r \\ x_{k-1} - 1 & \text{si } a_k = l \\ x_{k-1} & \text{sinon} \end{cases}$$

$$y_0 = 0$$
$$\forall 1 \leq i \leq n \quad y_k = \begin{cases} y_{k-1} + 1 & \text{si } a_k = u \\ y_{k-1} - 1 & \text{si } a_k = d \\ y_{k-1} & \text{sinon} \end{cases}$$

$$\forall 1 \leq i \leq n \quad p_k = \begin{cases} \text{pix}(x_{k-1}, y_{k-1}) & \text{si } a_k = u \\ \text{pix}(x_{k-1}, y_{k-1} - 1) & \text{si } a_k = r \\ \text{pix}(x_{k-1} - 1, y_{k-1} - 1) & \text{si } a_k = d \\ \text{pix}(x_{k-1} - 1, y_{k-1}) & \text{si } a_k = l \end{cases}$$

Le couple $(x_k, y_k)$ représente les coordonnées du point où l'on se trouve après avoir lu les $k$ premières lettres de $\alpha$. Le pixel $p_k$ désigne le pixel « allumé » par la $k$-ième lettre de $\alpha$.

Il est clair que :

$$\text{base}(p) = \bigcup_{i=1}^{n} p_i$$

Comme $\text{FigP}(r\overline{r}\alpha) = \text{FigP}(\alpha) = p$, il existe donc $1 \leq k_0 \leq n$ tel que $p(k_0) = \text{pix}(0, -1)$ ($\text{pix}(0, -1)$ étant l'unique pixel de $p$ allumé par $r\overline{r}$). Deux possibilités :

1. $a_{k_0} = r$, on voit facilement que l'on peut prendre $\beta = a_1 \dots a_{k_0 - 1}$ ;
2. $a_{k_0} \neq r$, on a alors $\text{Inv}(a_{k_0}) = \overline{a_{k_0}} = \delta r \delta'$ avec $\delta, \delta' \in \Pi^*$ et on pose $\beta = \alpha \overline{a_n} \dots \overline{a_{k_0+1}} \delta$.

La condition $(C)$ (et donc $(P)$) est bien vérifiée. □

On peut donc utiliser le système de réécriture $S$ comme base de la congruence. Néanmoins, dans le cas particulier du monoïde $\mathcal{F}_p^c$, nous allons pouvoir utiliser un système légèrement simplifié.

**Définition 2.15** *Le système de réécriture $S'$ est défini sur les mots de $\Pi^*$ :*

$$S' = S'_1 \cup S'_2 \cup S'_3$$

$$\text{avec} \quad \begin{cases} S'_1 = \{a \leftrightarrow a\overline{a}a \mid a \in \Pi\} \\ S'_2 = \{\alpha\alpha \rightarrow \alpha \mid \text{FigP}(\alpha) \in \mathcal{B}\} \\ S'_3 = \{\alpha \rightarrow \overline{\alpha} \mid \text{FigP}(\alpha) \in \mathcal{B}\} \end{cases}$$

La seule différence avec le système de réécriture $S$, c'est que les règles de l'ensemble $S'_3$ ne sont pas symétriques. Mais les règles réciproques des règles de $S'_3$ peuvent se déduire de $S'$.

**Proposition 2.16** *Du système de réécriture $S'$, nous pouvons déduire les ensembles de règles :*

$$\begin{aligned} T'_1 &= \{\alpha \leftrightarrow \overline{\overline{\alpha}} \mid \alpha \in \Pi^*\} \\ T'_2 &= \{\overline{\alpha} \rightarrow \alpha \mid \text{FigP}(\alpha) \in \mathcal{B}\} \end{aligned}$$

*Preuve.* Nous considérons les ensembles de règles un à un :

1. $T_1'$ : il suffit de montrer que les règles avec $\alpha$ ne comportant qu'une lettre peuvent être déduites de $S$ :

$$\forall a \in \Pi \quad a \overset{*}{\underset{S'}{\leftrightarrow}} \overline{\overline{a}}$$

Supposons que $a = r$, les autres cas se résolvant de manière symétrique. On a :

$$\overline{\overline{r}} = \overline{\overline{dlu}} = rdlurdlur = r\overline{r}r\overline{r}r \overset{2}{\underset{S_1'}{\leftrightarrow}} r$$

On voit facilement que l'on peut généraliser cette règle aux mots sur $\omega \in \Pi^*$, en effet si $\omega = a_1 \dots a_n$, on a $\overline{\overline{\omega}} = \overline{\overline{a_1}} \dots \overline{\overline{a_n}}$. Notons que si $\omega = \varepsilon$, on a $\overline{\overline{\omega}} = \varepsilon$.

2. $T_2'$ : il s'agit des règles réciproques de celles de $S_3'$. Soit $\alpha \in \Pi^*$ tel que $\mathrm{FigP}(\alpha) \in \mathcal{B}$, on a :

$$\overline{\alpha} \underset{S_3'}{\rightarrow} \overline{\overline{\alpha}} \underset{T_1'}{\leftrightarrow} \alpha$$

□

Ceci nous permet d'assurer que la relation $\overset{*}{\underset{S'}{\rightarrow}}$ est bien la congruence $\equiv$.

**Proposition 2.17** *La relation $\overset{*}{\underset{S'}{\rightarrow}}$ coïncide avec la congruence $\equiv$ :*

$$\forall \omega, \omega' \in \Pi^* \quad \omega \equiv \omega' \Leftrightarrow \omega \overset{*}{\underset{S'}{\leftrightarrow}} \omega'$$

*Preuve.* Immédiat d'après le Théorème 2.8 et les Propositions 2.14 et 2.16. □

Notre système de réécriture $S'$ contient un nombre infini de règles ($S_2'$ et $S_3'$) et il ne peut exister de système de réécriture fini possédant la propriété requise.

**Proposition 2.18** *Tout système de réécriture dont la relation induite est la congruence $\equiv$ est infini.*

*Preuve.* Par l'absurde. Supposons qu'il existe un système de réécriture $S''$ fini qui satisfait la propriété. Il existe donc $n$ la longueur maximale des parties gauches et droites des règles de ce système.

On considère le mot $\omega = dr^{n+1}ul^{n+1}$. Ce mot représente une figure comme celle de la Figure 2.2. Dans $\omega$, le pixel $p_1$ est allumé avant le pixel $p_2$.

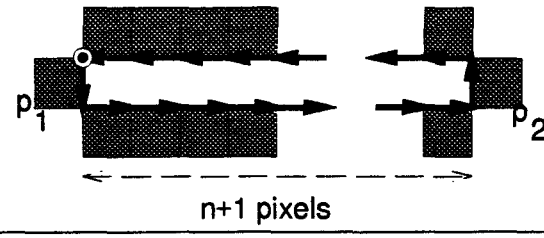



FIG. 2.2 - *Le mot $\omega$ ne pouvant être inversé*

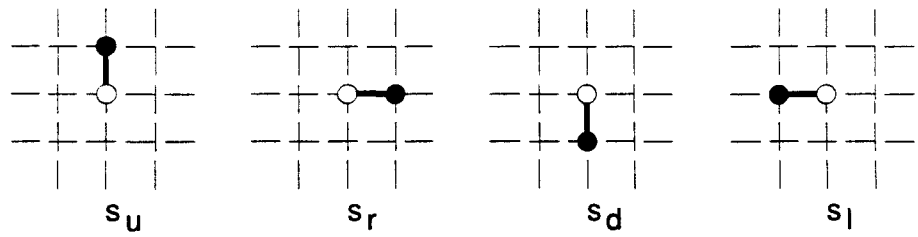



FIG. 2.3 - *Ensemble des segments unitaires*

Soit le mot $\omega' = ur^{n+1}drdldl^{n+1}ulur$. Les mots $\omega$ et $\omega'$ décrivent bien la même figure. Donc on a $\omega \overset{*}{\underset{S''}{\leftrightarrow}} \omega'$. Or dans $\omega'$, le pixel $p_2$ est allumé avant le pixel $p_1$. Il existe donc dans la dérivation $\omega_1$ et $\omega_2$ tels que $\omega_1$ se dérive en $\omega_2$ en une seule étape:

$$\omega \overset{*}{\underset{S''}{\rightarrow}} \omega_1 \underset{S''}{\rightarrow} \omega_2 \overset{*}{\underset{S''}{\rightarrow}} \omega'$$

et tels que dans $\omega_1$ le pixel $p_1$ soit allumé avant $p_2$, dans $\omega_2$ le pixel $p_2$ soit allumé avant $p_1$. Il existe donc dans $S''$ une règle $\alpha_1 \rightarrow \alpha_2$ et on a $\omega_1 = \beta\alpha_1\beta'$ et $\omega_2 = \beta\alpha_2\beta'$. Clairement, $p_1$ et $p_2$ sont tracés dans $\alpha_1$ et $\alpha_2$ sinon l'ordre ne serait pas inversé. Donc $\alpha_1$ trace $p_1$ et $p_2$. Or la distance entre $p_1$ et $p_2$ est strictement supérieure à $n$, donc la longueur de $\alpha_1$ est strictement supérieure à $n$, d'où contradiction. □

## 2.3 Langages de figures classiques

Comme nous l'avons déjà cité, le Théorème 2.8 est aussi une généralisation d'un résultat montré par P. Séébold et K. Slowinski [SS91]. En effet, dans la sémantique « classique » des langages de figures proposée par H.A. Maurer et al. dans [MRW82], les mots sur $\Pi$ décrivent des figures composées de segments.

Les dessins sont alors des ensembles finis de segments unitaires reliant des points voisins de $\mathbb{Z}^2$. Les dessins pointés sont des dessins munis d'un point de départ et d'un point d'arrivée. La concaténation est définie de manière identique à celle proposée à la Définition 1.1. Les figures sont également des classes d'équivalence de dessins modulo une translation.

L'ensemble étudié dans cette sémantique est l'ensemble des figures connexes qui sont engendrées par l'ensemble des segments unitaires illustré Figure 2.3 :

$$Q = \{s_u, s_r, s_d, s_r\}$$

Il s'agit également d'un monoïde inversif qui a comme système générateur le couple $(\Pi, \text{dpic})$ avec $\Pi = \{u, r, d, l\}$ et $\text{dpic}(u) = s_u$, $\text{dpic}(r) = s_r$, $\text{dpic}(d) = s_d$ et $\text{dpic}(l) = s_l$.

On montre de manière similaire à la Proposition 2.14 que la condition (P) est vérifiée par ce système générateur doté de la fonction inverse Inv : $\text{Inv}(u) = d$, $\text{Inv}(r) = l$, $\text{Inv}(d) = u$ et $\text{Inv}(l) = r$. Ceci nous permet d'utiliser le système de réécriture $S$ pour la congruence associée à $(\Pi, \text{dpic})$. Notons qu'ici, nous avons pour tout mot $\alpha$ sur $\Pi$ : $\overline{\overline{\alpha}} = \alpha$ et, ainsi, l'ensemble de règles $S_3$ est $\{\alpha \to \overline{\alpha} \mid h(\alpha) \in I_{Q^*}\}$. On retrouve donc exactement le système de réécriture de [SS91]. De même, R. Gutbrod a proposé dans [Gut89] un système de réécriture ayant les mêmes propriétés mais contenant $(S)$.

Dans [SS91], P. Séébold et K. Slowinski déduisent du système de réécriture un algorithme permettant de trouver un mot le plus court possible décrivant la même figure que le mot d'entrée. Nous verrons dans le chapitre suivant pourquoi un tel algorithme n'a pu être construit.

# Références

[Gut89] R. Gutbrod. A transformation system for generating description languages of chain code pictures. *Theoretical Computer Science*, 68(3):239–252, 1989.

[LRS97] M. Latteux, D. Robilliard, and D. Simplot. Figures composées de pixels et monoïde inversif. In *Proc. Journées Montoises d'Informatique Théorique*, volume 4 of *Bulletin of the Belgian Mathematical Society - Simon Stevin*, pages 89–111, Mons, Belgium, 1994, 1997.

[MRW82] H. A. Maurer, G. Rozenberg, and E. Welzl. Using string languages to describe picture languages. *Information and Control*, 54(3):155–185, 1982.

[SS91] P. Séébold and K. Slowinski. The shortest way to draw a connected picture. *Computer Graphics Forum*, 10(4):319–327, 1991.

# Chapitre 3

# Complexité descriptive

Nous disposons donc d'un système générateur $(\Pi, \mathrm{FigP})$ pour l'ensemble des figures connexes. Pour décrire une figure $f$ (différente de la figure vide $f_\epsilon$) du monoïde inversif $\mathcal{F}_p^c$, nous avons vu qu'il existait un ensemble infini, mais rationnel, de mots du monoïde libre $\Pi^*$ qui décrivent $f$ (voir Proposition 1.20). Il est donc intéressant de regarder quelle est la longueur des mots les plus courts qui décrivent $f$.

Dans la sémantique classique, Maurer et al. [MRW82] définissent la complexité descriptive comme étant le rapport entre la taille du plus petit mot et le poids de la figure. Rappelons que dans cette sémantique à segments, pour toute figure composée de $n$ segments il existe un mot sur $\Pi$ de longueur au plus $2.n$ qui la décrit. Nous cherchons à avoir une caractérisation identique des mots minimaux pour les pixels.

## 3.1 Définitions

Donnons d'abord quelques définitions sur la complexité descriptive et les mots minimaux.

**Définition 3.1** *1. Soit $\omega \in \Pi^*$, on dit que $\omega$ est minimal si et seulement si :*

$$\forall \omega' \in \Pi^* \quad \mathrm{FigP}(\omega') = \mathrm{FigP}(\omega) \Rightarrow |\omega'| \geq |\omega|$$

*2. Soit $f$ une figure pointée, on appelle $\mathrm{Min}(f)$, l'ensemble des mots minimaux qui décrivent $f$.*

$$\mathrm{Min}(f) = \{\omega \in \Pi^* \mid \mathrm{FigP}(\omega) = f \wedge \omega \textit{ minimal}\}$$

*3. On définit la longueur minimale de la description d'une figure pointée $f$ :*

$$\text{l-min}(f) = |\omega| \quad \omega \in \mathrm{Min}(f)$$

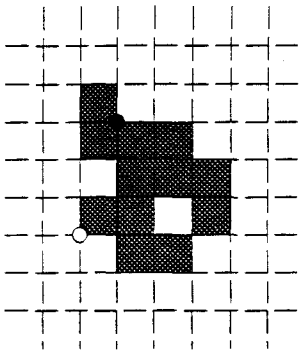


FIG. 3.1 - *La figure pointée $f$*

Pour mesurer la difficulté à décrire une figure pointée donnée par le système générateur que nous avons choisi, nous définissons la complexité descriptive en reliant la longueur de ses mots minimaux au poids de la figure.

**Définition 3.2** *Soit $f \in \mathcal{F}_p^c$, la complexité descriptive de $f$, notée* comp$(f)$, *est définie par:*

$$\text{comp}(f) = \frac{\text{l-min}(f)}{||f||}$$

Par exemple, si l'on considère la figure pointée $f$ illustrée Figure 3.1, l'ensemble Min$(f)$ est composé de 11 mots qui sont de longueur seize:

1. *urdrlrruuldululr*
2. *urdrruuldlrululr*
3. *urdudrruuldululr*
4. *urrdduruuldululr*
5. *urrdlrruuldululr*
6. *urrldrruudullulr*
7. *urrldrruuldululr*
8. *urrldrruulrllulr*
9. *urrudddurullulr*
10. *urruddlrruullulr*
11. *urrudldrruullulr*

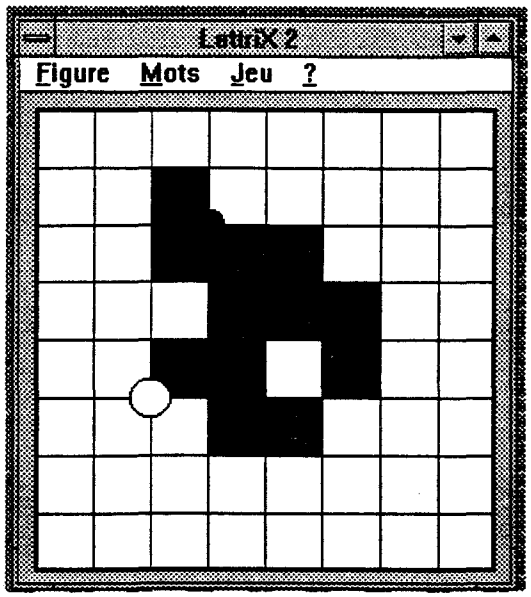



FIG. 3.2 - *Figure saisie avec* LattriX

On a donc l-min = 16 et, comme $||f|| = 12$, la complexité descriptive de $f$ est $\text{comp}(f) = \frac{16}{12} = \frac{4}{3} \simeq 1,333\ldots$. L'ensemble $\text{Min}(f)$ nous est fourni par un programme que nous avons écrit et qui permet entre autres de trouver les mots minimaux d'une figure pointée. Ce programme a été baptisé *Lattrix* et il suffit de lui donner la figure pointée (voir Figure 3.2) pour qu'il retourne la liste des mots minimaux et illustre le parcours de chaque mot (voir Figure 3.3 pour les mots *urdrlrruuldululr* et *urrdduruuldululr*).

Pour savoir s'il s'agit d'une figure compliquée à dessiner, il faut donc s'interroger sur le max de la complexité descriptive pour l'ensemble des figures pointées, c'est l'objet de la section suivante.

## 3.2 Complexité des figures pointées

Regardons d'abord un cas particulier de figures pointées que sont les boucles.

**Lemme 3.3** *Soit $f \in \mathcal{B}$ une boucle, on a* $\text{comp}(f) \leq 4$.

*Preuve.* On considère un représentant $p = (q, a, a) \in \mathcal{D}_p^c$ de $f$ tel que $f = [p]_{\simeq}$. On associe à $q$ un graphe défini par :

$$g = \bigcup_{(x,y)\in q} \left\{ \begin{array}{l} ((x,y),(x,y+1)),((x,y+1),(x+1,y+1)) \\ ((x+1,y+1),(x+1,y)),((x+1,y),(x,y)) \end{array} \right\}$$

Par exemple, si $q$ est le dessin de la Figure 3.4, le graphe associé est celui de la Figure 3.5. Le graphe $g$ est un graphe eulérien, on va donc pouvoir trouver un chemin qui parcourt

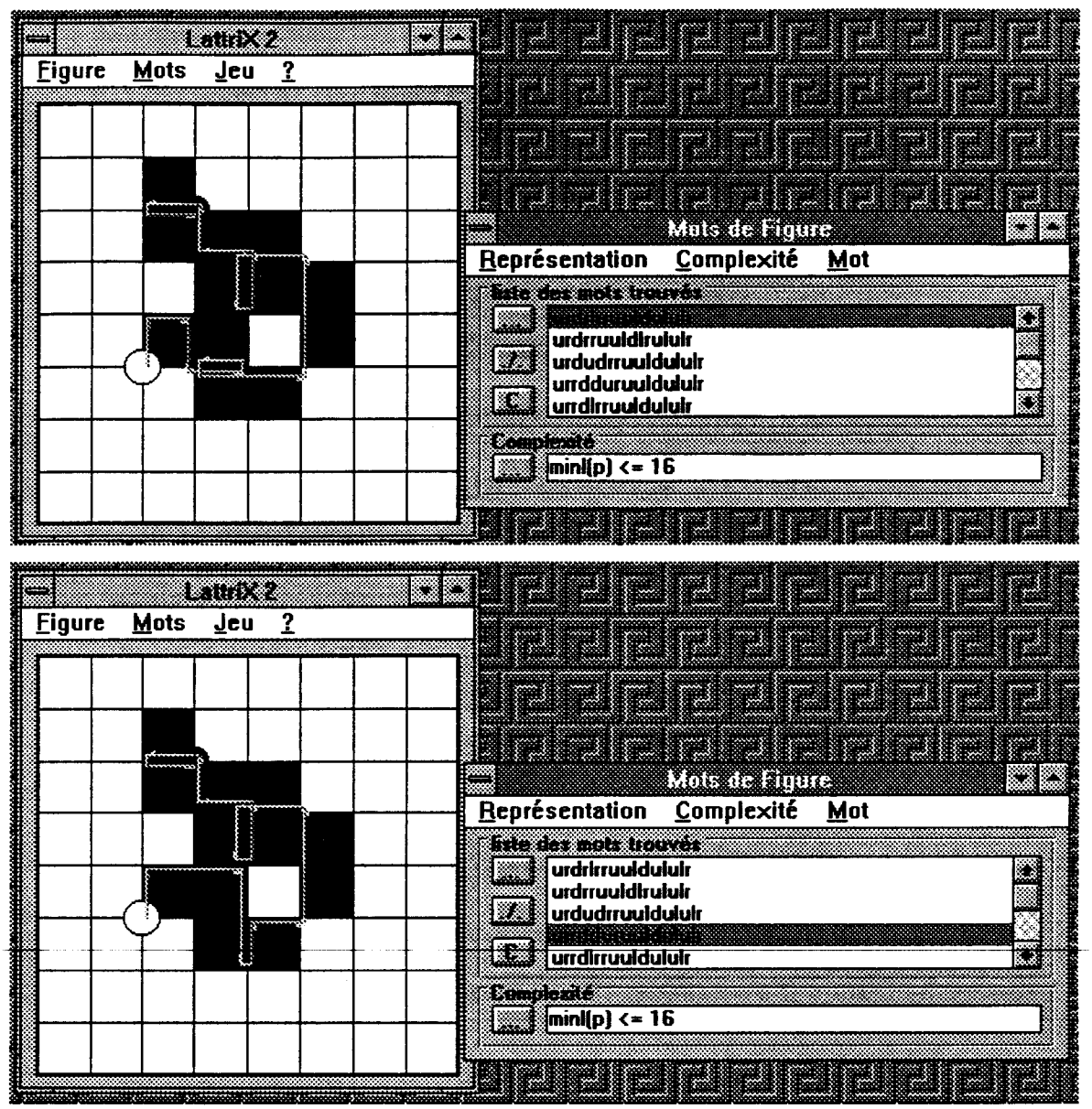



FIG. 3.3 - *Mots mimimaux trouvés par* LattriX *pour une figure pointée*

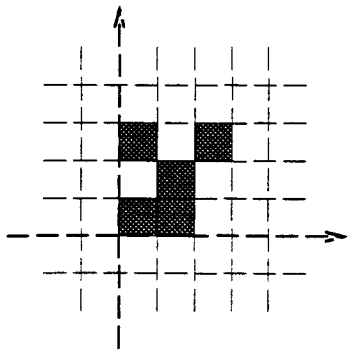


FIG. 3.4 - *Exemple de dessin q*

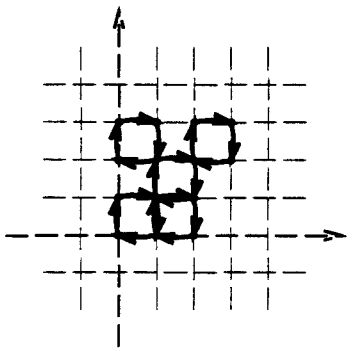


FIG. 3.5 - *Graphe eulérien associé à q*

chaque arc une seule fois qui part de $a$ et qui arrive à $a$. Le graphe comportant exactement 4 fois plus d'arcs que $f$ n'a de pixels, on a bien $\text{comp}(f) \leq 4$.

□

Notons que pour les boucles composées d'un unique pixel la complexité descriptive est exactement de 4. Ceci nous permet de borner la complexité descriptive des figures pointées quelconques.

**Proposition 3.4** *Soit $f$ une figure pointée, on a :* $1 \leq \text{comp}(f) \leq 5$.

*Preuve.* Pour allumer $n$ pixels, il faudra au moins utiliser $n$ lettres, on a donc bien $\text{comp}(f) \geq 1$.

On montre maintenant que pour toute figure pointée $f \in \mathcal{F}_p^c$, il existe un mot $\omega$ tel que $\text{FigP}\omega) = f$ avec $|\omega| \leq 5||f||$.

On raisonne par récurrence sur la taille de $f$.

Si $||f|| = 1$, on peut décrire $f$ avec au plus 4 lettres selon la position du point de départ et du point d'arrivée.

Sinon, on suppose que la propriété est vraie jusque $||f|| - 1$.

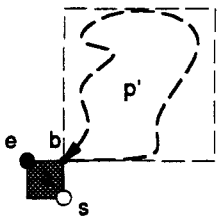



FIG. 3.6 - *Cas le pire où s et e appartiennent au même pixel*

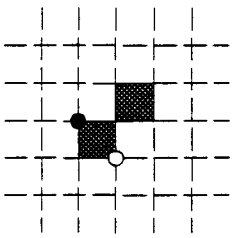


FIG. 3.7 - *Figure pointée de taille deux et de complexité cinq*

Prenons un représentant $p \in \mathcal{D}_p^c$ de $f$ tel que $f = [p]_{\simeq}$, deux cas peuvent se présenter :

1. On peut séparer $p$ en deux parties connexes $p_1$ et $p_2$ telles que $p = p_1.p_2$ et base$(p_1) \cap$ base$(p_2) = \emptyset$ (voir remarques sur la décomposition Section 1.3 page 9). Par hypothèse de récurrence, on sait qu'il existe $\omega_1$ (resp. $\omega_2$), un mot sur $\Pi$ décrivant $[p_1]_{\simeq}$ (resp. $[p_2]_{\simeq}$) tel que $|\omega_1| \leq 5.||p_1||$ (resp. $|\omega_2| \leq 5.||p_2||$). On pose $\omega = \omega_1\omega_2$ et on a bien un mot qui décrit $f$ avec $|\omega| \leq 5.||f||$.

2. On ne peut pas séparer $p$ en deux parties. C'est donc qu'il n'existe qu'un pixel $a$ sur l'armature duquel se trouvent le point de départ et le point d'arrivée. Dans ce cas, la situation la pire (en complexité) est celle de la Figure 3.6.

   Le dessin $p'$ est le dessin base$(p)$ à laquelle on a retiré le pixel de départ $a$. La boucle $(p', b, b)$ peut être décrite par un mot $\omega$ avec $\omega = 4.||p'|| = 4.(||f|| - 1)$ (Lemme 3.3).

   On peut donc décrire $f$ par $\omega' = lur\omega dlu$, c'est-à-dire en $3+4(||f||-1)+3 = 4.||f||+2$. Comme $||f|| > 1$, on a bien $|\omega'| \leq 5.||f||$.

□

Cette borne est élevée et est atteinte par la figure de taille 2 illustrée Figure 3.7 — le seul mot minimal la décrivant est en effet *lururdldlu*. On peut néanmoins constater en énumérant les figures de taille 2, 3, 4... que cette limite n'est plus atteinte. On peut donc se demander ce que devient la complexité descriptive lorsque la taille augmente. Pour notre part, nous pensons que la complexité ne peut que tendre vers une borne inférieure ou égale à quatre, même si c'est par une convergence supérieure.

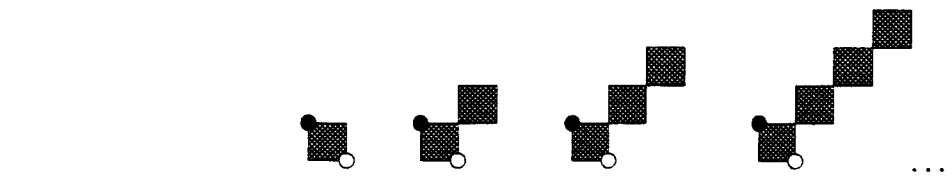


FIG. 3.8 - *Famille de figure dont la complexité descriptive tend vers quatre*

Il est facile de trouver une famille de figures pointées dont la complexité tend vers quatre (en restant supérieure). On prend par exemple les figures de la forme :

$$f_n = \mathrm{FigP}(lur(ur)^n(dl)^n dlu)$$

qui sont illustrées Figure 3.8. Néanmoins si l'on s'intéresse uniquement à la base des figures, c'est-à-dire si les points de départ et d'arrivée peuvent être placés où l'on désire, la complexité descriptive des figures de la famille précédente passe en dessous de 2 — alors que pour la figure pointée de complexité cinq (illustrée Figure 3.7) elle devient 1 (sa base est décrite par $ru$). On se propose donc d'étudier la complexité descriptive des figures (non pointées).

## 3.3 Complexité des figures

Se donner la possibilité de placer les points de départ et d'arrivée où l'on veut nous permet bien sûr de trouver des mots plus courts car on a une contrainte en moins. Pour une figure, on définit la complexité descriptive de la même manière que pour les figures pointées :

**Définition 3.5** *1. Soit $\omega \in \Pi^*$, on note* $\mathrm{Fig}(\omega)$ *la figure (non pointée) décrite par $\omega$ :* $\mathrm{Fig}(\omega) = \mathrm{base}(\mathrm{FigP}(\omega))$.

*2. Soit $\omega \in \Pi^*$, on dit que $\omega$ est b-minimal si et seulement si :*

$$\forall \omega' \in \Pi^* \quad \mathrm{Fig}(\omega') = \mathrm{Fig}(\omega) \Rightarrow |\omega'| \geq |\omega|$$

*3. Soit $f \in \mathcal{F}^c$ une figure, on appelle* $\mathrm{Min}(f)$, *l'ensemble des mots b-minimaux qui décrivent $f$.*

$$\mathrm{Min}(f) = \{\omega \in \Pi^* \mid \mathrm{Fig}(\omega) = f \wedge \omega \textit{ b-minimal}\}$$

*4. On définit la longueur minimale de la description d'une figure $f \in \mathcal{F}^c$ :*

$$\text{l-min}(f) = |\omega| \quad \omega \in \mathrm{Min}(f)$$

*5. Soit $f \in \mathcal{F}^c$, la complexité descriptive de $f$, notée* comp($f$), *est définie par:*

$$\text{comp}(f) = \frac{\text{l-min}(f)}{||f||}$$

Si l'on reprend la figure $f$ de la Figure 3.1, on note $g = \text{base}(f)$, l'ensemble Min($g$) contient deux mots de longueur treize (rappelons que $f$ ne pouvait être dessinée en moins de seize lettres) :

$$\text{Min}(g) = \{udrruuldulluu, udrruuldululr\}$$

La complexité descriptive de $g$ est donc $\text{comp}(g) = \frac{13}{12} \simeq 1,083\ldots$ (contre $1,333\ldots$ pour $f$). Ces mots b-minimaux de $g$ nous sont également fournis par *LattriX* (voir Figure 3.9).

On voit donc facilement que :

**Proposition 3.6** *Soit $\omega \in \Pi^*$, on a $\omega$ b-minimal $\overset{\Rightarrow}{\not\Leftarrow}$ $\omega$ minimal.*

*Preuve.*

$\Rightarrow$ Supposons que $\omega$ ne soit pas minimal. Il existe donc $\omega' \in \Pi^*$ tel que $\text{FigP}(\omega) = \text{FigP}(\omega')$ et $|\omega| > |\omega'|$. On a $\text{Fig}(\omega) = \text{Fig}(\omega)$, donc $\omega$ n'est pas b-minimal, d'où contradiction.

$\not\Leftarrow$ Il suffit de prendre l'exemple de $\omega = ur$ qui est minimal mais pas b-minimal.

□

Le Lemme 3.3 nous permet d'énoncer la proposition suivante.

**Proposition 3.7** *Soit $f \in \mathcal{F}^c$ une figure, on a $1 \leq \text{comp}(f) < 4$.*

*Preuve.* Il suffit de considérer une boucle dont $f$ est la base. D'après la preuve du Lemme 3.3, on sait que l'on peut trouver un mot de longueur $4.||f||$ qui « allume » chaque pixel quatre fois. Pour avoir l'inégalité stricte, il suffit donc de retirer la dernière lettre. □

Cette borne ne peut donc pas être atteinte, mais peut-on s'en rapprocher autant que l'on veut ?

**Proposition 3.8** *Il existe une famille de figures $(f_n)_{n\in\mathbb{N}}$ telle que pour tout $i > 0$ on ait $||f_{i-1}|| < ||f_i||$ on a :*

$$\lim_{n\to\infty}(\text{comp}(f_n)) = 4$$

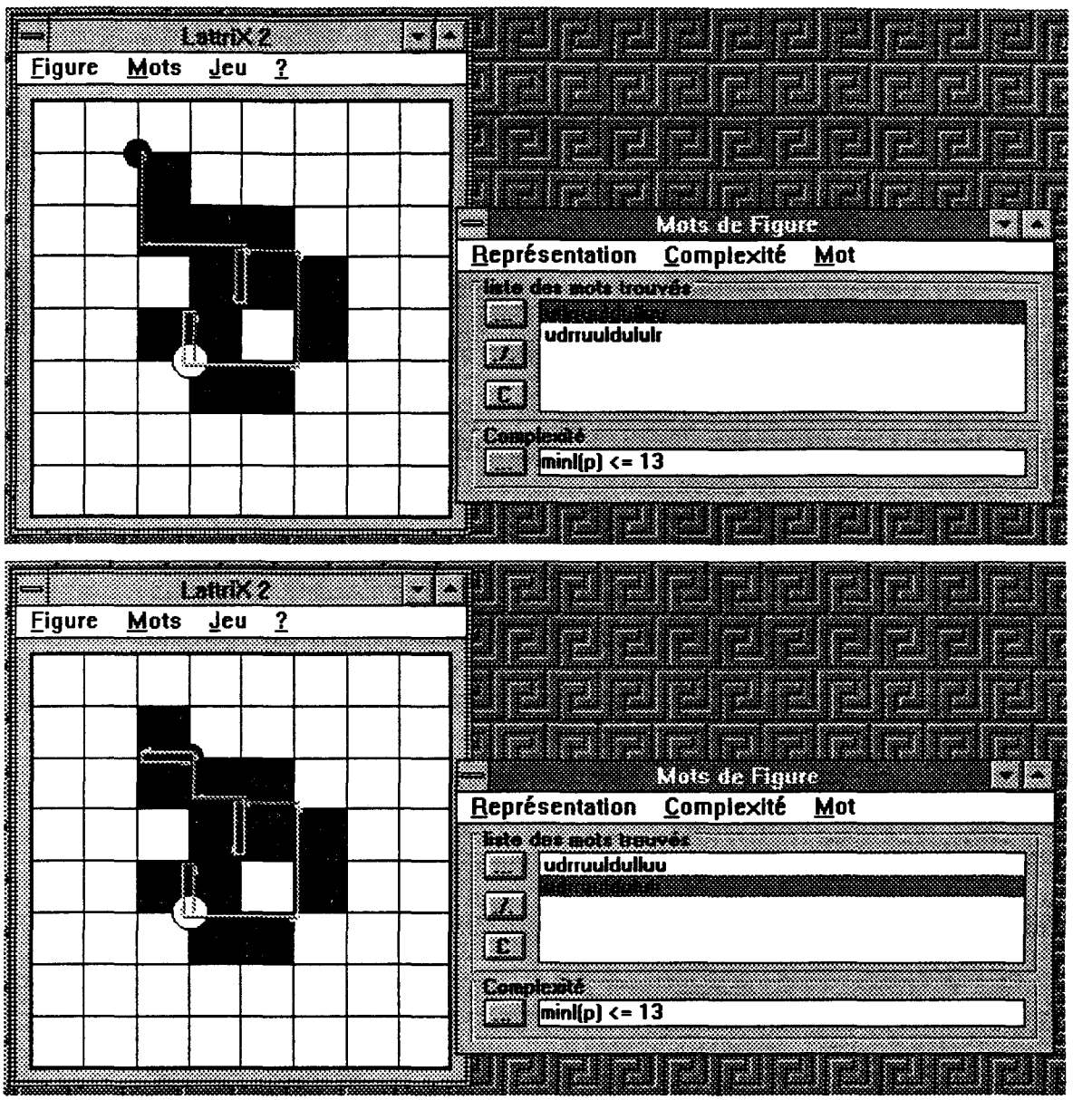



FIG. 3.9 - *Mots mimimaux trouvés par* LattriX *pour une figure*

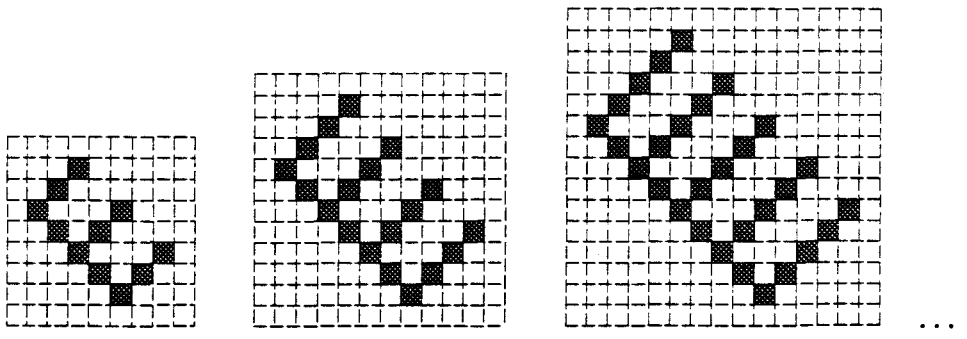


FIG. 3.10 - *Famille de figures dont la complexité tend vers quatre*

Nous donnons cette famille qui est décrite Figure 3.10 et par :

$$f_n = \mathrm{Fig}((ld)^{n+1}(rdr(ur)^{n+1}(dl)^{n+1}d)^n rd(ru)^{n+1})$$

Nous allons montrer qu'il s'agit bien d'une famille de figures dont la complexité tend vers quatre quand $n$ tend vers l'infini. Mais pour ceci, nous avons besoin de quelques lemmes techniques.

**Définition 3.9** *Le diamètre d'un dessin $p \in \mathcal{D}^c$ est noté $\delta(p)$ et est défini par :*

$$\delta(p) = \max\{\min\{|\omega| \mid \omega \in \Pi^* \ \mathrm{FigP}(\omega) = [(g,p,q)]_{\simeq} \wedge g \subseteq p\} \mid p,q \in \mathrm{Arm}(p)\}$$

Le diamètre mesure la distance la plus grande qu'il faut parcourir entre deux points de l'armature du dessin tout en restant dans le dessin. La notion de diamètre est étendue naturellement aux figures et aux figures pointées :

$$\forall f = [p]_{\sim} \in \mathcal{F}^c \quad \delta(f) = \delta(p)$$
$$\forall f \in \mathcal{F}_p^c \quad \delta(f) = \delta(\mathrm{base}(f))$$

**Lemme 3.10** *Soit $f \in \mathcal{F}_p^c$ une figure pointée, on a :*

$$\text{l-min}(\mathrm{base}(f)) \leq \text{l-min}(f) \leq \text{l-min}(\mathrm{base}(f)) + 2.\delta(f)$$

*Preuve.* Le fait que l-min(base($f$)) $\leq$ l-min($f$) est déduit directement de la Proposition 3.6.

Nous montrons maintenant que l-min($f$) $\leq$ l-min(base($f$)) + $2.\delta(f)$. Raisonnons par l'absurde : supposons que l-min(base($f$)) + $2\delta(f)$ < l-min($f$). Prenons $(p,s,e)$ un représentant de $f$, il existe un mot $\omega \in \mathrm{Min}(\mathrm{base}(f))$ tel que $\mathrm{FigP}(\omega) = [(p,s',e')]_{\simeq}$ et donc si $\omega' \in \mathrm{Min}(f)$, on a :

$$|\omega| + 2.\delta(f) < |\omega'|$$

Construisons $\omega'' = \alpha\omega\beta$ où $\alpha$ est un chemin dans le dessin $p$ de $s$ à $s'$ et où $\beta$ est un chemin dans le dessin $p$ de $e'$ à $e$. On a :

$$|\omega''| = |\alpha| + |\omega| + |\beta| \leq \delta(f) + |\omega| + \delta(f) = |\omega| + 2.\delta(f)$$

Or on a $|\omega'| > |\omega| + 2.\delta(f)$ d'où contradiction puisque $\text{FigP}(\omega'') = f$. □

D'un autre point de vue, ceci nous permet également de borner la taille minimale d'un mot décrivant la base de $f$ en fonction l-min($f$) et de $\delta(f)$ :

$$\text{l-min}(f) - 2.\delta(f) \leq \text{l-min}(\text{base}(f)) \leq \text{l-min}(f)$$

Il est en effet souvent plus facile de calculer $\delta(f)$ et l-min($f$) que l-min(base($f$)). Par exemple, pour les figures pointées, on dispose du lemme suivant :

**Lemme 3.11** *Soit $f = [(p, s, e)]_{\simeq} \in \mathcal{F}_p^c$ une figure pointée. Si $g = [(q, s, r)]_{\simeq}$ et $g' = [(q', r, e)]_{\simeq}$ sont deux figures pointées telles que $g.g' = f$, $q \cap q' = \emptyset$ et $\text{Arm}(q) \cap \text{Arm}(q') = \{r\}$, alors on a :*

$$\forall \omega, \omega' \in \Pi^* \quad \omega \in \text{Min}(g) \wedge \omega' \in \text{Min}(g') \Rightarrow \omega.\omega' \in \text{Min}(f)$$

*Preuve.* Soit $\gamma \in \text{Min}(f)$, on a :

$$\gamma = \alpha_0\beta_1\alpha_1\beta_2 \ldots \alpha_{n-1}\beta_n$$

avec pour tout $1 \leq i \leq n-1 \quad \alpha_i, \beta_i \in \Pi^+$ et $\alpha_0, \beta_n \in \Pi^*$ tels que les $\alpha_j$ ne tracent que des pixels de $q$ et les $\beta_j$ que des pixels de $q'$. On note :

$$\begin{aligned}
\text{FigP}(\alpha_0) &= [(a_0, s, x_0)]_{\simeq} \\
\text{FigP}(\beta_1) &= [(b_1, x_0, y_1)]_{\simeq} \\
\text{FigP}(\alpha_1 &= [(a_1, y_1, x_1)]_{\simeq} \\
\text{FigP}(\beta_2) &= [(b_2, x_1, y_2)]_{\simeq} \\
&\vdots \\
\text{FigP}(\alpha_{n-1}) &= [(a_{n-1}, y_{n-1}, x_{n-1})]_{\simeq} \\
\text{FigP}(\beta_n) &= [(b_n, x_{n-1}, e)]_{\simeq}
\end{aligned}$$

Clairement, puisque $r$ est le seul « point de passage » entre $q$ et $q'$, on a :

$$x_0 = y_1 = x_1 = y_2 = \ldots = y_{n-1} = x_{n-1} = r$$

Donc $\beta_1$, $\alpha_1$, $\beta_2$,..., $\alpha_{n-1}$ sont des mots qui décrivent des boucles. On a vu que le système de réécriture $S'$ (voir Définition 2.15 Section 2.2 du chapitre précédent) pouvait

intervertir deux boucles consécutives (règles $T_1$ Lemme 2.7). On peut donc obtenir la dérivation suivante :

$$\begin{aligned}\gamma = \alpha_0\beta_1\alpha_1\beta_2\ldots\alpha_{n-1}\beta_n &\xrightarrow[T_1]{} \alpha_0\alpha_1\beta_1\beta_2\ldots\alpha_{n-1}\beta_n \\ &\xrightarrow[T_1]{*} \alpha_0\alpha_1\ldots\alpha_{n-1}\beta_1\beta_2\ldots\beta_n\end{aligned}$$

On pose alors $\alpha = \alpha_0\alpha_1\ldots\alpha_{n-1}$ et $\beta = \beta_1\beta_2\ldots\beta_n$. Il est clair que $\alpha$ décrit $g$ et que $\beta$ décrit $g'$. De plus, on voit facilement que $\alpha\beta \in \mathrm{Min}(f)$ car $\mathrm{FigP}(\alpha\beta) = f$ ($S'$ conserve les figures) et $|\alpha\beta| = |\gamma|$.

Mieux, $\alpha \in \mathrm{Min}(g)$ car sinon, il existe $\alpha' \in \mathrm{Min}(g)$ avec $|\alpha'| < |\alpha|$ et donc $\mathrm{FigP}(\alpha'\beta) = f$ avec $|\alpha'\beta| < |\gamma|$ d'où $\gamma \notin \mathrm{Min}(f)$, ce qui est en contradiction avec les hypothèses. De même, on montre que $\beta \in \mathrm{Min}(g')$.

Par $S'$, on peut dériver $\alpha$ et $\beta$ en tout mot minimal de $g$ et $g'$ respectivement, d'où le lemme. □

Pour trouver un mot minimal d'une figure, on peut donc la décomposer en deux parties qui ne se touchent que par un point et travailler séparément sur ces deux composantes.

**Proposition 3.12** *Soit $(f_n)_{n\in\mathbb{N}^*}$ la famille de figures (non pointées) définie par :*

$$f_n = \mathrm{Fig}((ld)^{n+1}(rdr(ur)^{n+1}(dl)^{n+1}d)^n rd(ru)^{n+1})$$

*On a :*

$$\lim_{n\to\infty}(\mathrm{comp}(f_n)) = 4$$

*Preuve.* Posons $\omega_n = (ld)^{n+1}(rdr(ur)^{n+1}(dl)^{n+1}d)^n rd(ru)^{n+1}$. Grâce au Lemme 3.11, il est facile de voir que : $\omega_n \in \mathrm{Min}(\mathrm{FigP}(\omega_n))$. La figure pointée décrite par $w_3$ est illustrée Figure 3.11. Pour ceci, il suffit de décomposer la figure pixel par pixel et de faire la concaténation. On a donc $\text{l-min}(\mathrm{FigP}(\omega_n)) = |\omega_n| = 4n^2 + 12n + 6$.

Sans perte de généralités, on peut dire que le diamètre est la longueur du chemin qui relie le point le plus en haut à gauche au point le plus à droite en bas (voir Figures 3.10 et 3.11). On obtient $\delta(f_n) = 8n + 10$. Enfin, on sait que $||f_n|| = n^2 + 5n + 5$. D'après le Lemme 3.10 et la Proposition 3.7, on a :

$$\begin{gathered}\text{l-min}(\mathrm{FigP}(\omega_n)) - 2\delta(\mathrm{FigP}(\omega)) \leq \text{l-min}(\mathrm{Fig}(\omega)) \\ 4n^2 + 12n + 6 - 2(8n + 10) \leq \text{l-min}(f_n) \\ \tfrac{4n^2-4n-14}{n^2+5n+5} \leq \mathrm{comp}(f_n) < 4\end{gathered}$$

On a donc bien $lim_{n\to\infty}(\mathrm{comp}(f_n)) = 4$. □

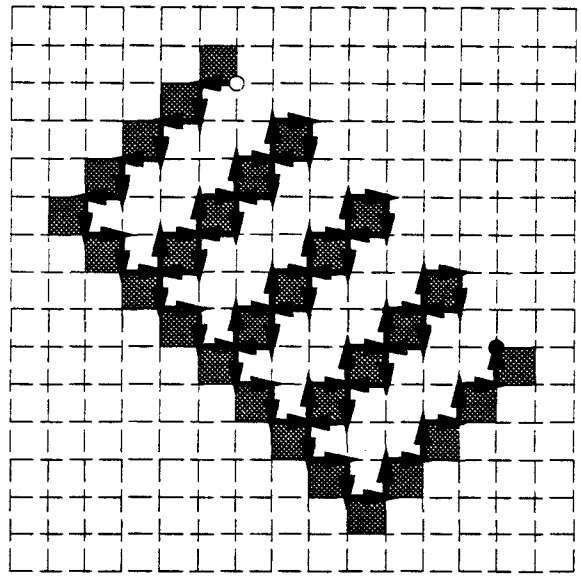


FIG. 3.11 - *Figure pointée décrite par* $w_3 = (ld)^4(rdr(ur)^4(dl)^4d)^3rd(ru)^4$

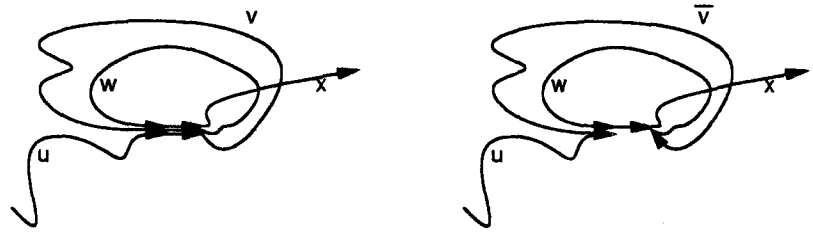



FIG. 3.12 - *Parcours avec un segment tracé trois fois*

## 3.4 Remarques

On aurait aimer pouvoir utiliser le système de réécriture $S'$ que nous décrivons dans le Chapitre 2 pour déduire un algorithme efficace pour trouver les mots minimaux d'une figure à l'instar de P. Séébold et K. Slowinski [SS91] dans les figures à segments. Malheureusement, cet algorithme (dans les segments) utilise une propriété des mots minimaux qui n'a pas d'équivalent dans les pixels. Les auteurs de cet algorithme utilisent le fait que dans un mot de figure minimal aucun segment ne peut être parcouru plus de deux fois. En effet, si un segment est parcouru trois fois, une manipulation permet de réduire la taille du mot.

Supposons que pour décrire une figure à segments on ait un mot du type $u.r.v.r.w.r.x$ avec $u, v, w, x \in \Pi^*$ tels que les trois $r$ tracent le même segment. C'est-à-dire que l'on est dans la situation de la Figure 3.12 : $r.v$ et $r.w$ décrivent des boucles. On peut donc décrire la même figure avec le mot $u.\overline{v}.w.r.x$. Comme dans les segments $\overline{v}$, l'inverse de $v$, a la même taille que $v$, on diminue bien la taille de mot.

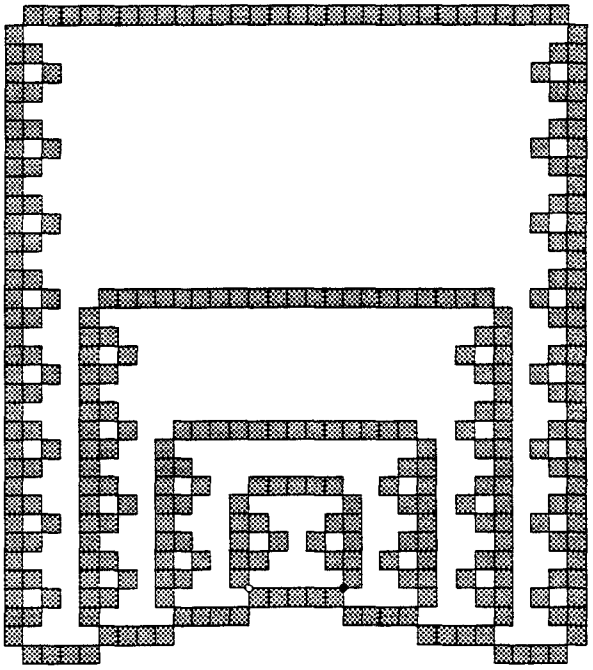


FIG. 3.13 - *Figure où l'on passe 5 fois sur les pixels centraux*

Or ceci n'est pas le cas dans les pixels. Pire, dans les pixels pour une figure pointée $f$ dont la description minimale est $n$, on n'est pas sûr de pouvoir trouver un mot de taille $n$ décrivant $f^{-1}$ (prendre par exemple la figure décrite par $ru$, le mot minimal décrivant son inverse est $rdldlu$) — le fait de tracer le pixel à droite du déplacement induit une certaine « anti-symétrie ». Dans le cas général d'un monoïde inversif $M$ ayant un système générateur $(\Sigma, h)$ et une fonction inverse Inv, si un élément $x \in M$ peut être décrit par un mot de taille $n$, la seule chose que l'on peut dire, c'est que l'élément $x^{-1}$ peut être décrit avec un mot de taille bornée par $k.n$ où $k$ est la longueur maximale de $\mathrm{Inv}(a)$ pour $a \in \Sigma$. Pour les segments, on est dans le cas idéal où $|\mathrm{Inv}(a)| = 1$ pour tout $a \in \Pi$.

Aussi, dans $\mathcal{F}_p^c$, on ne peut borner le nombre de « passages » sur un pixel donné dans un mot minimal. Par exemple, dans les figures, les pixels centraux compris entre le point de départ et le point d'arrivée de la figure illustrée Figure 3.13 sont toujours tracés cinq fois dans les mots minimaux. Ajouter une boucle ajoute systématiquement un passage. Les motifs sur les boucles ont été étudiés pour qu'il soit moins coûteux de repasser sur les pixels centraux et de les tracer dans le sens trigonométrique que de les parcourir dans l'autre sens.

# Références

[MRW82] H. A. Maurer, G. Rozenberg, and E. Welzl. Using string languages to describe picture languages. *Information and Control*, 54(3):155–185, 1982.

[SS91] P. Séébold and K. Slowinski. The shortest way to draw a connected picture. *Computer Graphics Forum*, 10(4):319–327, 1991.

# Chapitre 4

# Indécidabilité de propriétés existentielles dans les langages de figures rationnels

Avoir la possibilité de décrire un ensemble de figures nous pousse à nous poser un certain nombre de problèmes d'algorithmique. Nous nous intéressons à la description d'un ensemble de figures par des langages (de mots) classifiés selon la hiérarchie de Chomsky. Ainsi l'on sait que le problème de l'appartenance pour les langages de figures classiques (et donc également dans les langages à pixels) — c'est-à-dire savoir si une figure donnée est représentée dans un langage de $\Pi^*$ — est décidable mais NP-complet pour les langages rationnels [KS87]. On sait également décider si un langage algébrique décrit un nombre fini de figures [MRW82].

Dans cette lignée de problèmes de décision, nous nous intéressons dans ce chapitre à des propriétés existentielles. Soit $p$ un prédicat sur les mots de figures, pouvant donc porter sur la figure que représente le mot. On cherche à savoir si l'on peut décider du problème :

> *Soit $L$ un langage de mots de figures, existe-t-il dans $L$ un mot qui vérifie la propriété $p$ ?*

Nous donnons ici une méthode pour montrer l'indécidabilité de ce genre de problèmes pour différentes propriétés $p$ telles que « est un mot auto-évitant », « est un mot de contour de polyomino » pour les langages de figures rationnels ainsi que pour différentes sémantiques : segments [MRW82], mais aussi segments avec déplacements invisibles [HW88] et pixels. Cette méthode a été présentée dans un formalisme légèrement différent dans la thèse de D. Robilliard [Rob96]. Ici, nous la précisons et nous montrons de nouveaux résultats sur les langages de figures classiques dans un premier temps et nous montrons comment on peut l'appliquer aux langages de figures à pixels. Une version reprenant les résultats de [Rob96] et les résultats présentés ici peut être trouvée dans [RS96].

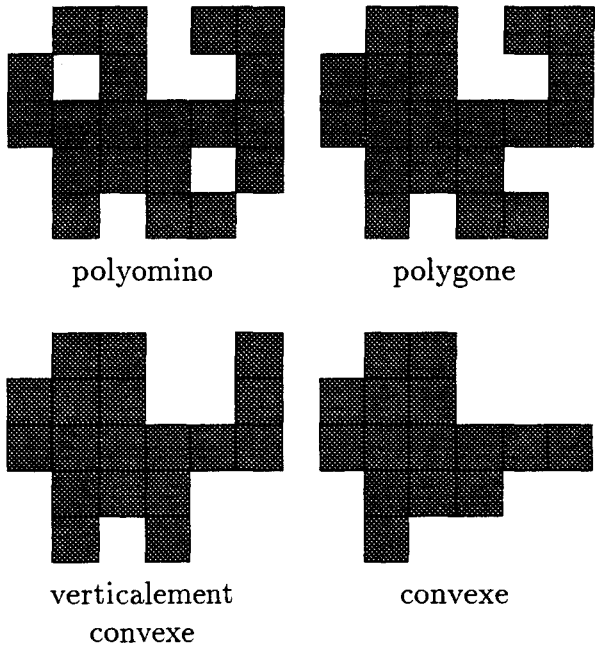



FIG. 4.1 - *Exemple des différentes classes de polyominos*

## 4.1 Langages linéaires

La méthode que nous développons dans les Sections 4.2 et suivantes, s'inspire d'une méthode simple à mettre en œuvre dans les langages linéaires. Ici, nous illustrons cette méthode, qui utilise le problème de correspondante de Post, sur les linéaires pour montrer qu'il est indécidable de savoir si un langage linéaire sur Π contient un mot de contour de polyomino convexe. Ce résultat est à mettre en rapport avec celui de D. Beauquier, M. Latteux et K. Slowinski qui montrent que cette question est décidable pour les rationnels [BLS92], ainsi qu'avec les résultats d'indécidabilité de J. Dassow sur les linéaires [Das89].

Nous donnons d'abord quelques définitions.

### Polyomino

Un *polyomino* est une partie finie de $\mathbb{Z}^2$ 4-connexe. Un polyomino sans trous est appelé *polygone*. Un polyomino est dit *horizontalement convexe* (respectivement verticalement convexe) si chacune de ses lignes (resp. colonnes) est connexe. Un *polyomino convexe* est un polyomino qui est à la fois horizontalement et verticalement convexe (voir Figure 4.1).

Les polyominos sont des objets combinatoires couramment utilisés comme modèles en physique car ils permettent par exemple de représenter des objets où un liquide peut circuler. Les problèmes classiques pour les polyominos sont l'énumération et la génération aléatoire qui sont reliés [BM94, BPS94, DV84] et le pavage [BN90, BN91, Gol70].

## Problème de correspondance de Post

La donnée du *problème de correspondance de Post* (noté PCP), est un couple $(U, V)$ avec $U = (u_1, \ldots, u_k)$ et $V = (v_1, \ldots, v_k)$ avec $k > 1$ et pour $1 \leq i \leq k$ $u_i, v_i \in \{0,1\}^+$. Une solution du PCP est une suite d'indices $i_1, \ldots, i_n \in \{1, \ldots, k\}$ avec $n \geq 1$ telle que :

$$u_{i_1} \ldots u_{i_n} = v_{i_1} \ldots v_{i_n}$$

Une variante du PCP (notée VPCP) consiste à trouver une suite d'indices $i_1, \ldots, i_n \in \{1, \ldots, k\}$ avec $n \geq 1$ telle que :

$$u_{i_1} \ldots u_{i_n} = v_{i_1} \ldots v_{i_n}$$

et

$$\forall 1 \leq j \leq n \quad |u_{i_1} \ldots u_{i_j}| \geq |v_{i_1} \ldots v_{i_j}|$$

Cette condition impose en fait que les $u_i$ soient toujours « en avance » sur les $v_i$. Le PCP et le VPCP sont bien connus pour être des problèmes indécidables [Pos47, LT90].

## Résultat sur les linéaires

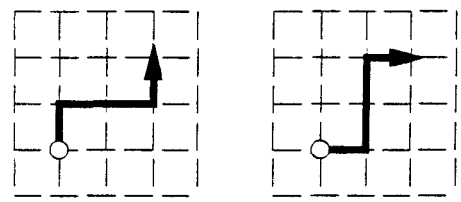
**Théorème 4.1** *Soit $G$ une grammaire linéaire sur $\Pi$. Il est indécidable de savoir si $L(G)$ contient un mot de contour de polyomino convexe.*

*Preuve.* Nous montrons que si ce problème était décidable, alors il en serait de même pour le PCP. Ainsi, pour chaque instance du PCP, on donne un langage linéaire qui contient un mot de contour de polyomino convexe si et seulement si le PCP a une solution. Soit $(U, V)$ une instance du PCP avec $U = (u_1, \ldots, u_k)$ et $V = (v_1, \ldots, v_k)$, nous construisons une grammaire linéaire $G$ sur $\Pi$ :

$$G = \begin{cases} S \to uAl \\ A \to \varphi(u_i) A \overline{\varphi(v_i)} & i \in \{1, \ldots, k\} \\ A \to \varphi(u_i) r d \overline{\varphi(v_i)} & i \in \{1, \ldots, k\} \end{cases}$$

où $\varphi$ est l'homomorphisme de $\{0,1\}^*$ dans $\Pi^*$ défini par :

$$\varphi(1) = ur^2u \qquad \varphi(0) = ru^2r$$



Remarquons que $sh(\varphi(1)) = sh(\varphi(0)) = (2,2)$ (et que symétriquement $sh(\overline{\varphi(1)}) = sh(\overline{\varphi(0)}) = (-2,-2)$).

Un mot $\omega$ de $L(G)$, d'après la définition de la grammaire, correspond à la donnée d'une suite $i_1, \ldots, i_j \in \{1, \ldots, k\}$ avec $j \geq 1$ déduite de la dérivation, et on a :

$$\omega = u.\varphi(u_{i_1} \ldots u_{i_j}).rd.\overline{\varphi(v_{i_1} \ldots v_{i_j})}.l$$

Notons $\alpha = u_{i_1} \ldots u_{i_j} = a_1 \ldots a_n$ et $\beta = v_{i_1} \ldots v_{i_j} = b_1 \ldots b_m$ avec les $a_p, b_p \in \{0,1\}$. On a :

$$\omega = u.\varphi(a_1) \ldots \varphi(a_n).rd.\overline{\varphi(b_m)} \ldots \overline{\varphi(b_1)}.l$$

On voit facilement que :

$$sh(\omega) = (2n - 2m, 2n - 2m) \tag{4.1}$$

Ainsi, si la suite $(i_1, \ldots, i_j)$ correspondant à $\omega$ est une solution du PCP, on a $n = m$ (c'est-à-dire $|\alpha| = |\beta|$) et, réciproquement, si $\omega$ est un mot de contour de polyomino convexe, il décrit une boucle et on a d'après (4.1) :

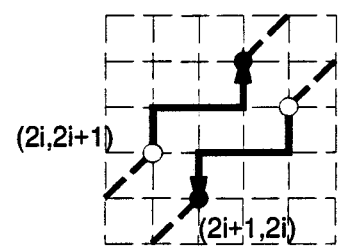
$$sh(\omega) = (0,0) \iff (2n - 2m, 2n - 2m) = (0,0) \iff n = m$$

Nous ne considérons que ce cas ; on peut donc écrire :

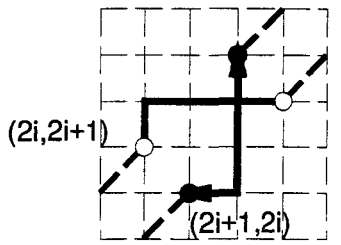
$$\begin{aligned} sh(u.\varphi(a_1) \ldots \varphi(a_i)) &= (2i, 2i+1) \\ sh(u.\varphi(a_1) \ldots \varphi(a_n).rd.\overline{\varphi(b_m)} \ldots \overline{\varphi(b_i)}) &= (2i+1, 2i) \end{aligned}$$

D'après la construction, on voit qu'il ne peut y avoir de cycles (c'est-à-dire d'intersections) strictement inclus dans $\omega$ qu'entre $\varphi(a_i)$ et $\overline{\varphi(b_i)}$ avec $i \in \{1, \ldots, n\}$ :

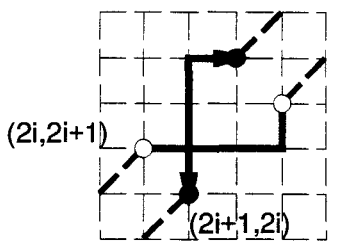
1. Cas $a_i = 1, b_i = 1$ : il n'y a pas d'intersections :



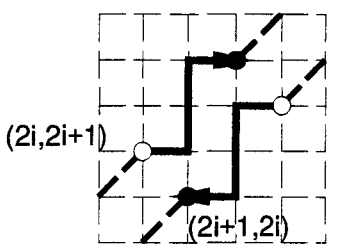

2. Cas $a_i = 1, b_i = 0$ : il y a intersection :




3. Cas $a_i = 0, b_i = 1$ : il y a intersection :




4. Cas $a_i = 0, b_i = 0$ : il n'y a pas d'intersections :




Ceci nous permet de conclure : si $(i_1, \ldots, i_j)$ est une solution du PCP alors $\omega$ décrit une boucle et ne possède pas de facteurs propres qui décrivent une boucle. De plus, le fait que $\omega$ soit dans $(u + r)^+(d + l)^+$ nous permet d'affirmer que $\omega$ est un mot de contour de polyomino convexe [BLS92]. Réciproquement, si $\omega$ est un mot de contour de polyomino, on a $|\alpha| = |\beta|$ et pour tout $1 \leq i \leq |\alpha|$ $a_i = b_i$, d'où $\alpha = \beta$. Si l'existence d'un mot de contour de polyomino convexe était décidable, le PCP le serait aussi, d'où contradiction. □

Avant de considérer les langages rationnels, nous donnons un exemple correspondant à la preuve précédente.

*Exemple. Soit le PCP* $((100, 11, 01), (10, 001, 001))$. *On voit que 1,3 est une solution mais que 2,1 n'en est pas une. La figure dessinée par le mot de G correspondant à la solution 1,3 est donnée Figure 4.2, il s'agit bien d'un mot de contour de polyomino convexe ; le mot de G est :*

$$\omega_{1,3} = u.\underbrace{ur^2u.ru^2r.ru^2r}_{\varphi(u_1)}.\underbrace{ru^2r.ur^2u}_{\varphi(u_3)}.rd.\underbrace{dl^2d.ld^2l.ld^2l}_{\overline{\varphi(v_3)}}.\underbrace{ld^2l.dl^2d}_{\overline{\varphi(v_1)}}.l$$

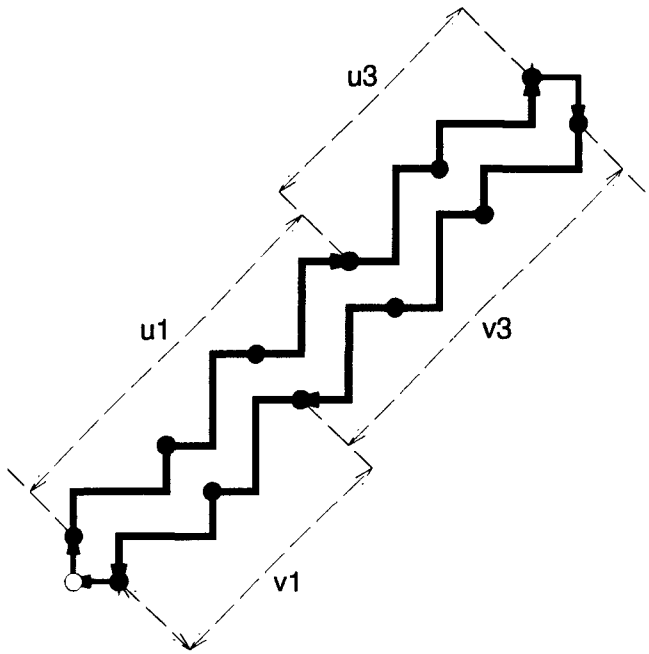



FIG. 4.2 - *Figure associée à la suite d'indices 1,3*

*Par contre, la suite d'indices 2,1 donne la figure illustrée Figure 4.3 qui n'est pas un contour de polyomino. Le mot de G associé à 2,1 est :*

$$\omega_{2,1} = u.\underbrace{ur^2u.ur^2u}_{\varphi(u_2)}.\underbrace{ur^2u.ru^2r.ru^2r}_{\varphi(u_1)}.rd.\underbrace{ld^2l.dl^2d}_{\overline{\varphi(v_1)}}.\underbrace{dl^2d.ld^2l.ld^2l}_{\overline{\varphi(v_2)}}.l$$

On remarquera que le langage utilisé dans la preuve est un « stripe language » ou « langage ruban » [SW85] c'est-à-dire que toutes les figures peuvent être tracées entre deux lignes parallèles données. La plupart des problèmes indécidables ou impraticables dans le cas général des langages de figures deviennent décidables ou polynomiaux dans les langages ruban. Or ici, on voit bien que cette restriction ne suffit pas à rendre décidable le problème de l'existence d'un polyomino convexe dans les langages ruban linéaires (exception faite des langages ruban horizontaux ou verticaux).

## 4.2 Méthode et exemple dans les rationnels

Passer des linéaires aux rationnels nous prive de la possibilité de générer des palindromes, ce qui semble a priori nécessaire pour le codage du PCP dans un langage. Nous donnons ici une méthode, pour les rationnels, qui repose sur la variante du PCP notée VPCP.

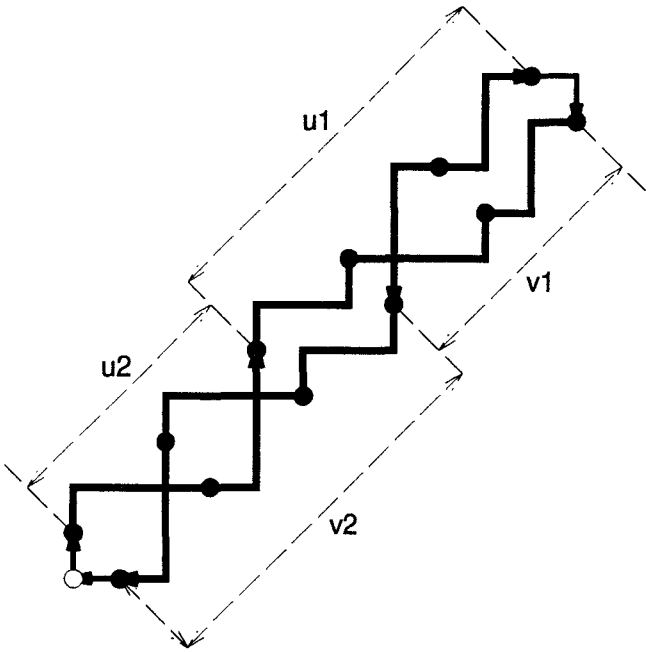



FIG. 4.3 - *Figure associée à la suite d'indices 2,1*

Nous illustrons cette méthode sur un exemple et nous ne la donnerons formellement qu'à la fin de la section. L'exemple que nous traitons est celui des chemins auto-évitants.

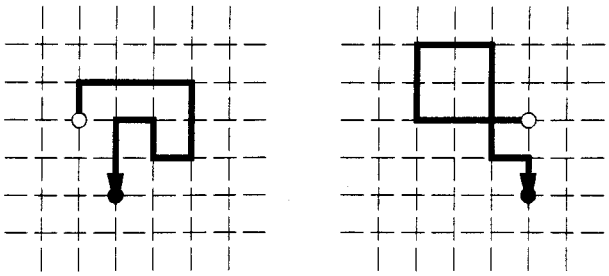
Un mot est dit « auto-évitant » s'il ne contient pas de boucles, c'est-à-dire si aucun de ses facteurs non vides n'est une boucle. Un exemple et un contre-exemple sont donnés Figure 4.4.

**Proposition 4.2** *Soit $L$ un langage rationnel sur $\Pi$. Il est indécidable de savoir si $L$ contient un mot auto-évitant.*

*Preuve.* Pour obtenir ce résultat, nous montrons que si ce problème était décidable, alors on pourrait décider du VPCP. Ainsi, pour chaque instance du VPCP, nous construisons un langage régulier de $\Pi^*$ qui contient un chemin auto-évitant si et seulement si l'instance du VPCP a une solution.

Soit $(U, V)$ une instance du VPCP avec les ensembles de mots $U = (u_1, \ldots, u_k)$ et $V = (v_1, \ldots, v_k)$. Notons $K$ le langage rationnel que nous construisons. Un mot $\omega$ de $K$ est une tentative de mise en correspondance pour une suite d'indices $i_1, \ldots, i_n \in \{1, \ldots, k\}$ avec $n \geq 1$. En donnant la construction de $K$, nous montrons que la seule possibilité d'obtenir un chemin auto-évitant est que $i_1, \ldots, i_n$ soit une solution du VPCP. Par construction, il sera facile de voir qu'à une solution on peut associer un mot auto-évitant de $K$. Les mots de $K$ seront de la forme :

$$\omega = \mu_0.\varphi(u_{i_1}).\lambda_1.\psi(v_{i_1}).\mu_1 \ldots \mu_{n-1}.\varphi(u_{i_n}).\lambda_n.\psi(v_{i_n}).\rho$$



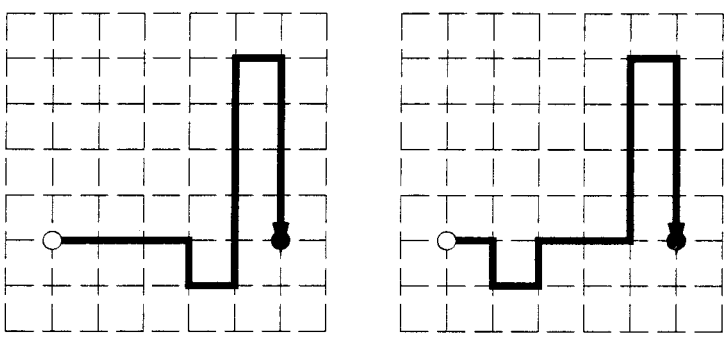
$ur^3d^2luld^2$
auto-évitant

$l^3u^2r^2d^3rd$
non-auto-évitant

FIG. 4.4 - *Exemple et contre-exemple de mots auto-évitants*

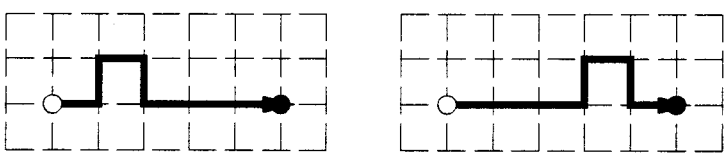
avec $\varphi$ et $\psi$ deux homomorphismes de $\{0,1\}^*$ dans $\Pi^*$ :

$\varphi(1) = r^3dru^5rd^4$ $\qquad$ $\varphi(0) = rdrur^2u^4rd^4$



et

$\psi(1) = rurdr^3$ $\qquad$ $\psi(0) = r^3urdr$



et les $\mu_i$, $\lambda_i$ et $\rho$ des mots de $\Pi^*$.

Les motifs $\varphi(0)$, $\varphi(1)$, $\psi(0)$ et $\psi(1)$ ont été élaborés de manière à ce que l'on puisse placer un motif $\psi(0)$ (respectivement $\psi(1)$) « en dessous » (décalé de deux unités vers le bas) d'un motif $\varphi(0)$ (resp. $\varphi(1)$) sans qu'il y ait intersections et que dans les autres cas il y ait intersections. Notons que la partie significative, qui « code » l'information 0 ou 1,

des motifs $\varphi(0)$ et $\varphi(1)$ se trouve en dessous de l'axe alors que la partie supérieure sert de contrôle comme nous le verrons par la suite, et que pour les motifs $\psi(0)$ et $\psi(1)$ la partie significative se trouve au-dessus.

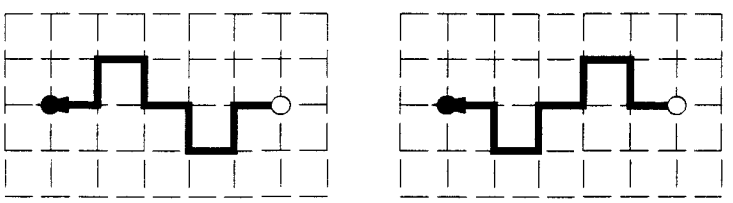
Ainsi, si l'on place $\psi(v_{i_1})$ sous (décalé de deux unité) $\varphi(u_{i_1})$, on n'obtiendra un mot sans intersections que si $v_{i_1}$ est un préfixe de $u_{i_1}$ ou vice-versa. Le rôle de $\lambda_1$ dans $\omega$ va donc être de venir « placer le crayon » en dessous de $\varphi(u_{i_1})$. Le mot $\lambda_1$ va être de la forme :

$$\lambda_1 \in \delta.(\zeta_0 + \zeta_1)^*.\eta$$

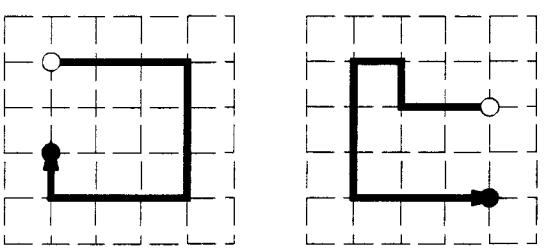
où $\delta$, $\zeta_0$, $\zeta_1$ et $\eta$ sont des mots sur $\Pi$.

Comme il est impossible de revenir placer le crayon au bon endroit sans perturber la comparaison, on recopie le motif de $\varphi(u_{i_1})$ grâce à $\zeta_0$ et $\zeta_1$.

$\zeta_1 = ldlululdl$ $\qquad$ $\zeta_0 = luldldlul$

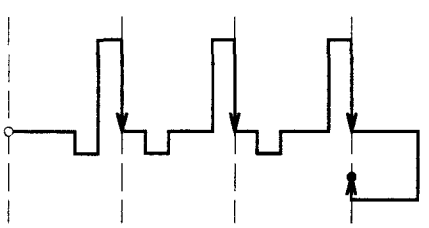


$\delta = r^3d^3l^3u$ $\qquad$ $\eta = l^2uld^3r^3$

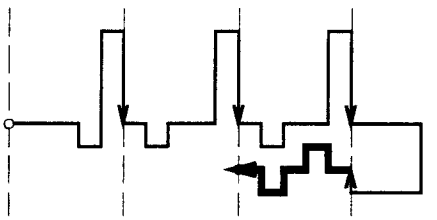


Si l'on considère l'homomorphisme $h$ de $\{0,1\}^*$ dans $\Pi^*$, qui associe à 0 le mot $\zeta_0$ et à 1 le mot $\zeta_1$, on veut montrer qu'il n'y a pas d'intersections dans $\mu_0.\varphi(u_{i_1}).\lambda_1$ que si $\lambda_1 = \delta.h(\widetilde{u_{i_1}}).\eta$.

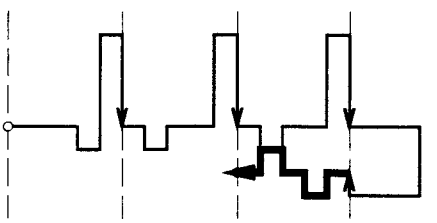
Prenons, par exemple, $u_{i_1} = 100$, la figure décrite par $\varphi(u_{i_1}).\delta$ est la suivante :



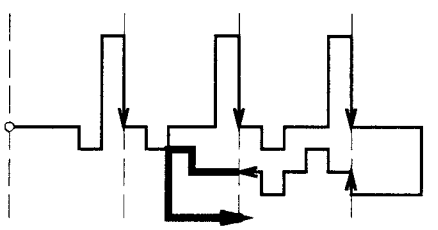
Les motifs $\zeta_0$ et $\zeta_1$ ont été conçus pour ne pas créer de cycles lorsqu'ils sont placés en dessous d'un 0 et un 1 respectivement. Ainsi, $\varphi(u_{i_1}).\delta.\zeta_0$ ne contient pas de cycles :



alors que $\varphi(u_{i_1}).\delta.\zeta_1$ en contient un :

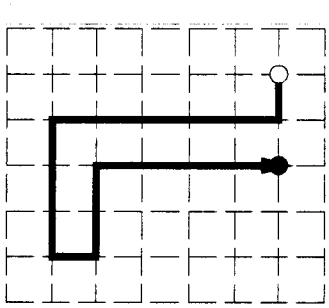


Ceci nous assure bien que la « recopie », c'est-à-dire la portion $(\zeta_0 + \zeta_1)^*$ du mot $\lambda_1$, correspond bien au motif créé par le $\varphi$. Il faut encore s'assurer qu'elle ne s'arrête pas trop tôt et qu'elle n'est pas trop longue. On peut être sûr qu'elle n'est pas trop courte car sinon le motif $\eta$ viendrait créer un cycle avec le motif non recopié :

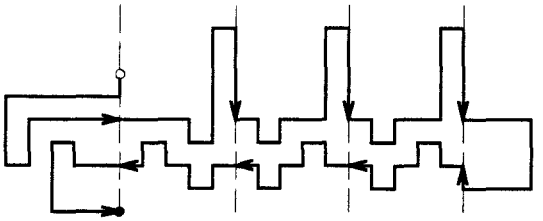


Pour s'assurer que la recopie n'est pas trop longue, on place avant $\varphi(u_{i_1})$, en posant $\mu_0 = \beta$ un motif qui créera un cycle. Ce motif $\beta$ est :

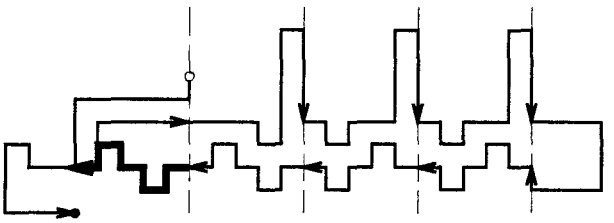
$$\beta = dl^5d^3ru^2r^4$$



Ainsi, si dans $\lambda_1$ la recopie est correcte et de bonne longueur, il n'y a pas d'intersections dans $\mu_0.\varphi(u_{i_1}).\lambda_1$ :

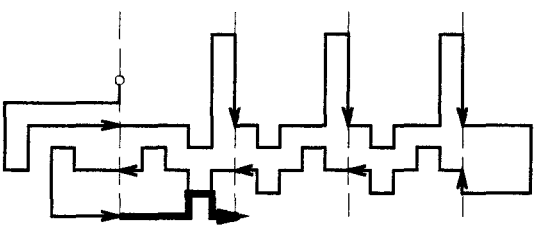


À l'opposé, si la recopie est trop longue un cycle est créé avec $\beta$ (exemple où l'on ajoute un $\zeta_1$ en trop) :

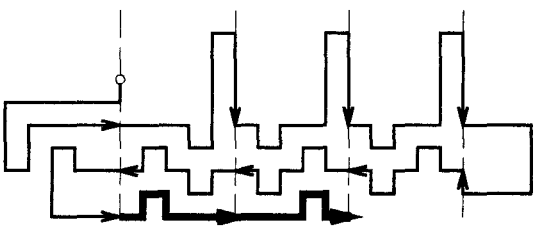


Ceci nous assure bien que $\lambda_1 = \delta.h(\widetilde{u_{i_1}}).\eta$ et qu'ainsi, en dessous d'un motif $\varphi(0)$, on a bien un $\zeta_0$ et qu'en dessous d'un motif $\varphi(1)$, on a bien un $\zeta_1$. Il s'agit maintenant de placer $\psi(v_{i_1})$ et de montrer qu'il n'y a pas de cycles que si $v_{i_1}$ est bien un préfixe de $u_{i_1}$. L'interaction entre les motifs $\zeta_0$, $\zeta_1$ d'une part et les motifs $\psi(0)$ et $\psi(1)$ d'autre part est la même qu'entre les $\varphi$ et les $\psi$, c'est-à-dire qu'il n'y a pas d'intersections si l'on place un $\psi(0)$ (respectivement $\psi(1)$) sous un $\zeta_0$ (resp. $\zeta_1$) et qu'il y en a dans les autres cas. Comme les $\zeta_0$ correspondent à des $\varphi(0)$ de $u_{i_1}$ et de manière similaire les $\zeta_1$ correspondent à des $\varphi(1)$ de $u_{i_1}$, si l'on n'a pas d'intersections, il y a bien correspondance entre les lettres de $v_{i_1}$ et les lettres de $u_{i_1}$.

Par exemple, toujours avec $u_{i_1} = 100$, même si $\lambda_1$ recopie correctement $u_{i_1}$, on a forcément une intersection dans $\mu_0.\varphi(u_{i_1}).\lambda_1.\psi(v_{i_1})$ si $v_{i_1} = 0$ :

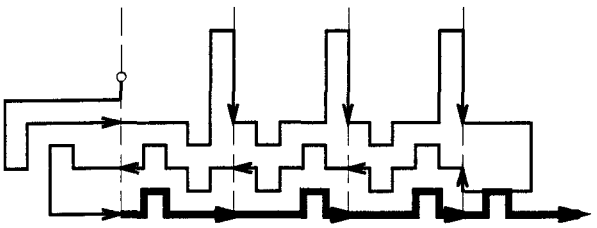


Par contre, si $v_{i_1} = 10$, il n'y a pas de cycles :



Le VPCP nous impose également que « les $v_i$ soient toujours en retard par rapport aux $u_i$ », c'est-à-dire que pour une solution $i_1, \ldots, i_n$, on ait pour tout $1 \leq j \leq n$ $|u_{i_1} \ldots u_{i_j}| \geq$

$|v_{i_1} \dots v_{i_j}|$. C'est le rôle de $\delta$ (présent dans $\lambda_1$) de créer un cycle si les $v_i$ sont trop longs. Par exemple, si $v_{i_1} = 1001$, on a intersection :



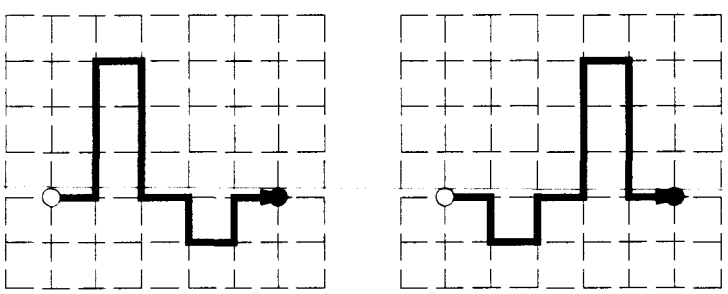
Notons $\nu_i$, pour $1 \leq i < n$ les mots de $\Pi^*$ tels que pour tout $1 \leq j < n$ $u_{i_1} \dots u_{i_j} = v_{i_1} \dots v_{i_j} \nu_j$. On pose $\nu_0 = \varepsilon$.

Dans le mot $\omega$ qui a la forme que l'on s'impose, si l'on n'a pas d'intersections dans le facteur $\mu_0.\varphi(u_{i_1}).\lambda_1.\psi(v_{i_1})$ (avec $\mu_0 = \beta$ et $\lambda_1 \in \delta.(\zeta_0 + \zeta_1)^*.\eta$), cela signifie que $v_{i_1}$ est bien un préfixe de $u_{i_1}$. Le mot $\nu_1$ correspond à la partie de $u_{i_1}$ qui n'a pas encore été comparée au mot formé par les $v_i$. Avant d'ajouter $\varphi(u_{i_2})$, il faut recopier $\nu_1$. Cette opération est faite de la même manière que dans $\lambda_1$ par $\mu_1$ qui est de la forme :

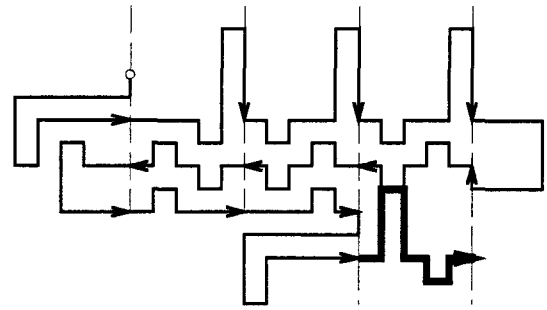
$$\mu_1 \in \beta.(\gamma_0 + \gamma_1)^*$$

Le mot $\beta$ est celui qui se trouve dans $\mu_0$ et on pose :

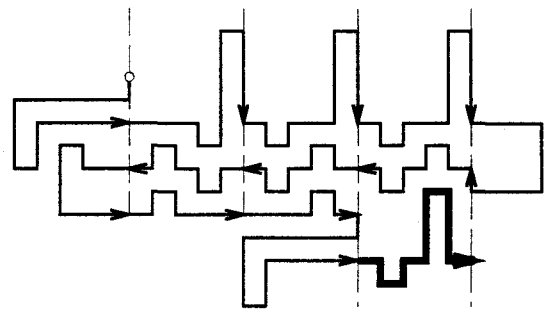
$\gamma_1 = ru^3rd^3rdrur$ $\qquad$ $\gamma_0 = rdrudu^3rd^3r$



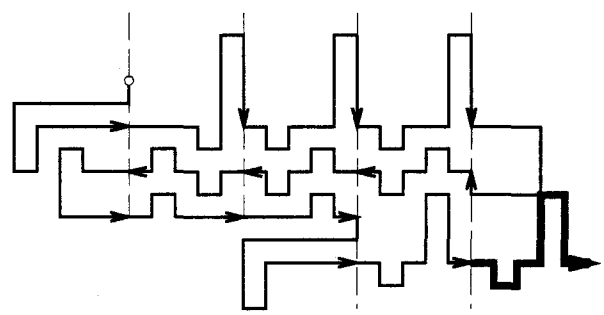
Le facteur $(\gamma_0 + \gamma_1)^*$ de $\mu_1$ correspond à la recopie de $\nu_1$ afin d'être comparé avec le prochain $v_i$. Si l'on considère l'homomorphisme $g$ de $\{0,1\}^*$ dans $\Pi^*$ qui associe à 0 le mot $\gamma_0$ et à 1 le mot $\gamma_1$, on veut montrer que s'il n'y a pas de cycles, alors, $\mu_1 = \beta.g(\nu_1)$. Selon le même principe que celui utilisé pour $\zeta_0$ et $\zeta_1$, si l'on ne veut pas de cycles, on est obligé de mettre un $\gamma_0$ en dessous d'un $\zeta_0$ et un $\gamma_1$ sous un $\zeta_1$. Ainsi dans la suite de l'exemple avec $u_{i_1} = 100$ et $v_{i_1} = 10$, $\mu_0.\varphi(u_{i_1}).\lambda_1.\psi(v_{i_1}).\beta.\gamma_1$ comporte un cycle :



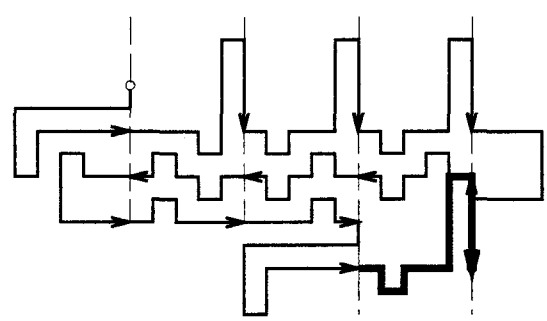
alors que $\mu_0.\varphi(u_{i_1}).\lambda_1.\psi(v_{i_1}).\beta.\gamma_1$ n'en a pas :



Ceci nous assure que $\nu_1$ est recopié correctement. Si la copie est trop longue, le motif de $\gamma_0$ ou $\gamma_1$ entre en collision avec celui de $\delta$ :



Pour vérifier que la recopie n'est pas trop courte, on utilise le fait qu'un $\mu_i$ sera suivi d'un $\varphi(u_i)$. Si l'on place un $\varphi(0)$ ou un $\varphi(1)$ alors que tout $\nu_1$ n'a pas été recopié, il y a intersection :

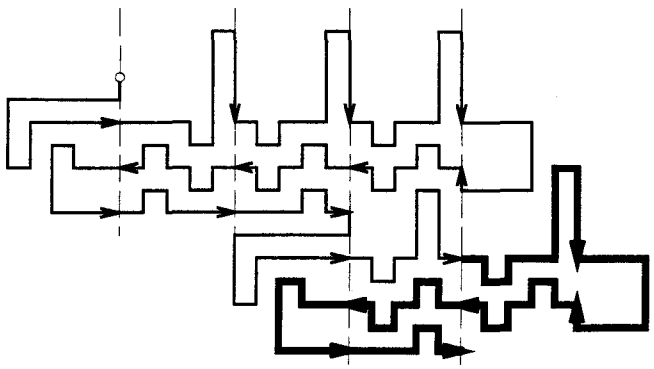


Le mot $\mu_1$ est donc la recopie exacte de $\nu_1$ et on a $\mu_1 = \beta.g(\nu_1)$.

Ensuite, dans $\omega$, on place $\varphi(u_{i_2})$ puis un mot $\lambda_2$, qui comme $\lambda_1$ appartient à $\delta.(\zeta_0+\zeta_1)^*.\eta$. De la même façon qu'avec $\lambda_1$, la seule possibilité pour qu'il n'y ait pas de cycles, est que $\lambda_2 = \delta.h(\nu_1.\widetilde{u_{i_2}}).\eta$. Puis on place $\psi(v_{i_2})$ et on montre comme ci-dessus que s'il n'y a pas de cycles, c'est que $v_{i_2}$ est préfixe de $\nu_1.u_{i_2}$. Si dans l'exemple précédent on prend $u_{i_2} = 0$ et $v_{i_2} = 0$, le mot (sans cycles) :

$$\mu_0.\varphi(u_{i_1}).\lambda_1.\psi(v_{i_1}).\mu_1.\varphi(u_{i_2}).\lambda_2.\psi(v_{i_2})$$

représente la figure :



Et ainsi de suite. En fait, on peut montrer par induction sur $n$ que si on a :

$$\begin{array}{lllll} & \mu_0 & = & \beta \\ \forall 1 \leq j < n & \mu_j & \in & \beta.(\gamma_0+\gamma_1)^* \\ \forall 1 \leq j < n & \lambda_j & \in & \delta.(\zeta_0+\zeta_1)^*.\eta \end{array}$$

alors :

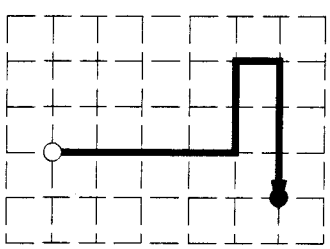
$$\mu_0.\varphi(u_{i_1}).\lambda_1.\psi(v_{i_1}).\mu_1 \ldots \mu_{n-1}.\varphi(u_{i_n}).\lambda_n.\psi(v_{i_n}) \text{ est auto-évitant}$$

$$\Updownarrow$$

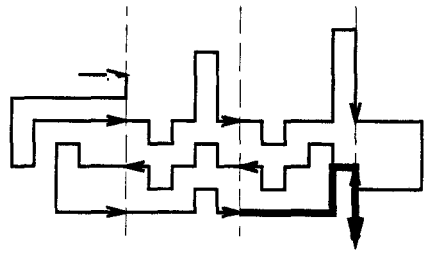
$$\begin{array}{lllll} \forall 1 \leq j < n & \mu_j & = & \beta.g(\nu_j) \\ \forall 1 \leq j < n & \lambda_j & = & \delta.h(\nu_{j-1}.\widetilde{u_{i_j}}).\eta \\ \forall 1 \leq j \leq n & v_{i_j} & \leq & \nu_{j-1}.u_{i_j} \end{array}$$

Reste à montrer que si $\omega$ tout entier est auto-évitant, alors $v_{i_n}$ est exactement $\nu_{n-1}.u_{i_n}$. Pour cela, il suffit de poser $\rho = \theta$ avec :

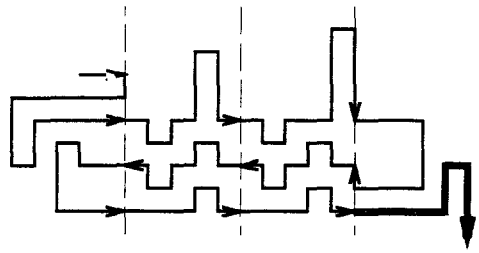
$$\theta = r^4u^2rd^3$$



Si la dernière comparaison finit trop tôt, il se formera un cycle avec $\theta$ :



Par contre, si la comparaison est de bonne longueur, aucun cycles n'est créé :

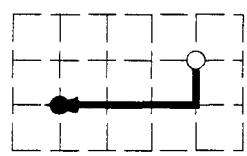


De la manière dont nous avons construit $\omega$, il ne peut être auto-évitant que si la suite d'indices $i_1, \ldots, i_n$ est une solution du VPCP. Pour pouvoir avoir toutes les suites d'indices possibles, il suffit de considérer le langage $K$ :

$$K = \alpha \left\{ \biguplus_{i=1}^{n} \underbrace{\beta(\gamma_0 + \gamma_1)^*}_{\text{« } \mu \text{ »}} \varphi(u_i) \underbrace{\delta(\zeta_0 + \zeta_1)^* \eta}_{\text{« } \lambda \text{ »}} \psi(v_i) \right\}^+ \theta$$

où $\alpha$ est le mot :

$$\alpha = dl^3$$



qui oblige le premier « $\mu$ » à être restreint à $\beta$ (c'est-à-dire au $\mu_0$ vu plus haut).

Ainsi, si $K$ contient un mot auto-évitant alors le VPCP correspondant a une solution. Réciproquement, si le VPCP a une solution, on peut lui associer trivialement un mot auto-évitant dans $K$. □

Notre méthode consiste donc à associer à toute instance du VPCP un langage de mots de figures rationnel qui ne contient un mot qui a la propriété souhaitée — être auto-évitant dans la proposition précédente — que si le VPCP a une solution.

**Définition 4.3** *Soit $(U, V)$ une instance du VPCP avec les ensembles $U = (u_1, \ldots, u_k)$ et $V = (v_1, \ldots, v_k)$, on associe à $(U, V)$ le langage $K_{(U,V)}$ sur l'alphabet $\Sigma$ :*

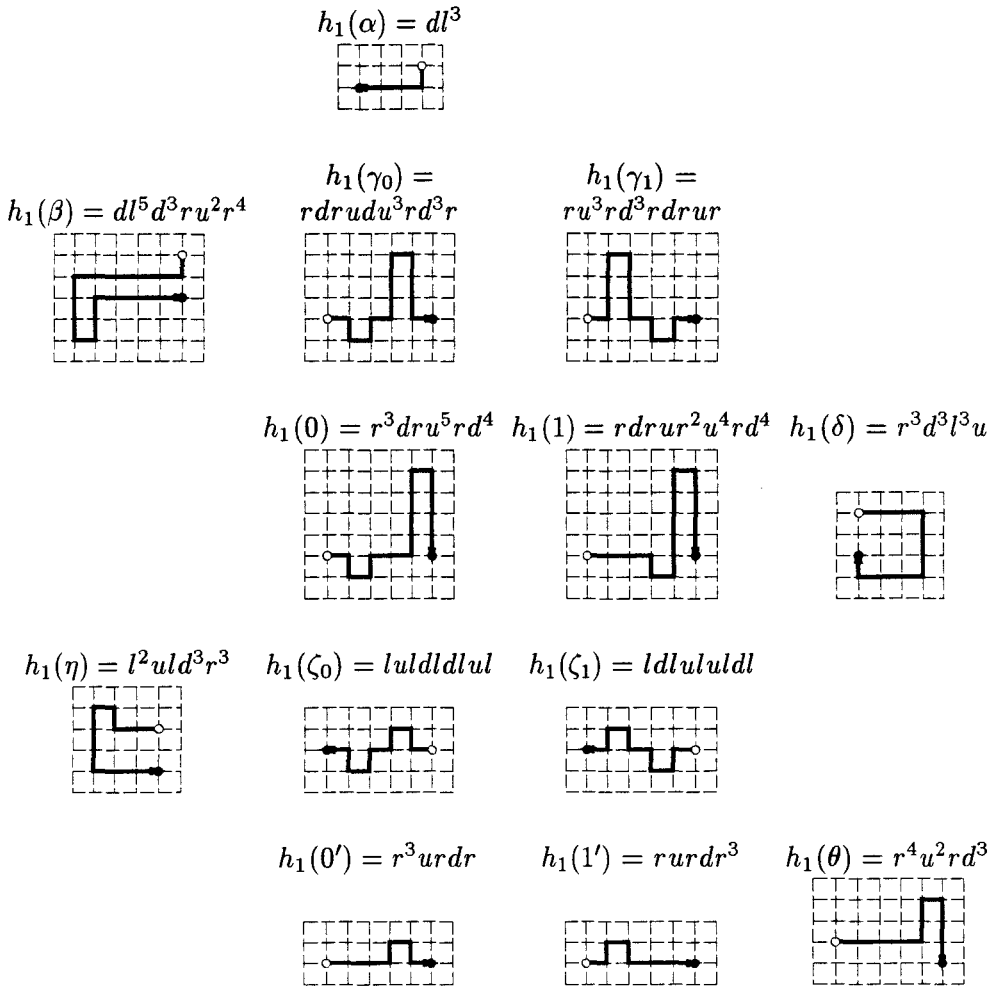



FIG. 4.5 - *Homomorphisme $h_1$ (existence d'un mot auto-évitant)*

$$\Sigma = \{0, 1, 0', 1', \alpha, \beta, \gamma_0, \gamma_1, \delta, \zeta_0, \zeta_1, \eta, \theta\}$$

$$K_{(U,V)} = \alpha \left\{ \biguplus_{i=1}^{n} \beta(\gamma_0 + \gamma_1)^* u_i \delta(\zeta_0 + \zeta_1)^* \eta . f(v_i) \right\}^+ \theta$$

*où $f$ est l'homomorphisme de $\{0,1\}^*$ dans $\Sigma^*$ tel que $f(0) = 0'$ et $f(1) = 1'$.*

Il suffit d'appliquer à $K_{(U,V)}$ un homomorphisme adéquat de $\Sigma^*$ dans $\Pi^*$ pour obtenir le résultat. Par exemple pour la Proposition 4.2, on a appliqué l'homomorphisme $h_1$ illustré Figure 4.5.

## 4.3 Autres résultats

Cette méthode a permis également de démontrer d'autres résultats sur les langages de figures à segments que l'on peut retrouver dans [Rob96]. La méthode permet aussi de montrer des résultats qui ne reposent pas uniquement sur des propriétés géométriques. C'est le cas pour l'existence d'un *mot optimal* ou d'un *mot minimal*. On dit qu'un mot est optimal s'il trace une et une seule fois chaque segment de la figure qu'il décrit. Et on dit qu'un mot est minimal s'il n'existe pas de mot de longueur plus courte qui décrit la même figure. Nous citons ces résultats pour références :

**Proposition 4.4 (Robilliard 96 [Rob96])** *Soit $L$ un langage rationnel sur $\Pi$. Il est indécidable de savoir si $L$ contient :*

1. *un mot de contour de polyomino (Beauquier 91 [Bea91]) ;*
2. *un arbre (Dassow, Hinz 93 [DH93]) ;*
3. *un mot optimal ;*
4. *un mot minimal.*

## 4.4 Langages de figures avec déplacements invisibles

L'utilisation de l'alphabet $\Pi$ pour coder des figures impose que celles-ci soient connexes. Il est intéressant d'étendre cet alphabet avec des lettres permettant d'effectuer des déplacements sans tracer de segment. Considérons l'alphabet $\Pi_i = \{u, r, d, l, u', r', d', l'\}$ où les lettres $u$, $r$, $d$ et $l$ ont la même signification que précédemment et où les lettres primées induisent un déplacement invisible [HW88]. Ceci nous permet donc de décrire des figures connexes ainsi que des figures non-connexes.

Il est donc naturel de se demander si l'on peut décider si toutes les figures d'un langage sont connexes ou toutes non-connexes. Nous répondons par la négative à ces deux questions avec les deux propositions suivantes. La preuve de la première proposition peut être trouvée dans [Rob96] :

**Proposition 4.5 (Robilliard 96 [Rob96])** *Soit $L$ un langage rationnel sur $\Pi_i$. Il est indécidable de savoir si $L$ contient un mot représentant une figure connexe.*

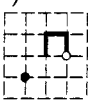
**Proposition 4.6** *Soit $L$ un langage rationnel sur $\Pi_i$. Il est indécidable de savoir si $L$ contient un mot représentant une figure non-connexe.*

*Preuve.* Soit $(U, V)$ une instance du VPCP, on considère le langage $L = h_4(K_{(U,V)})$ où $h_4$ est l'homomorphisme de $\Sigma^*$ dans $\Pi_i^*$ décrit Figure 4.6.

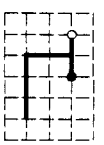
Un exemple de mise en correspondance est donné Figure 4.7. □

La méthode semble donc être suffisamment efficace pour s'étendre à d'autres sémantiques. C'est ce que nous allons encore voir avec une sémantique « à pixels ».

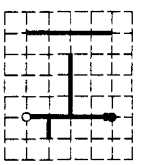
$h_4(\alpha) = uldd'l'$



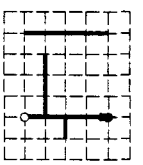
$h_4(\beta) = d^2ur^2d^3r'^2u'^2$



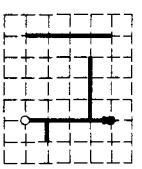
$h_4(\gamma_0) = rduru^3u'l^2r^3d'^4l^2r^2$



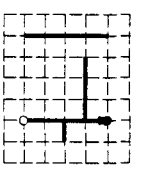
$h_4(\gamma_1) = ru^3u'lr^4d'^4l^3rdur^2$



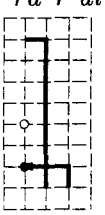
$h_4(0) = rdur^2u^3u'l^3r^4d'^4lr$



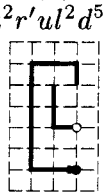
$h_4(1) = r^2duru^3u'l^3r^4d'^4lr$



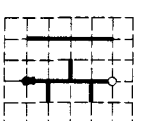
$h_4(\delta) = u'^4rd^7r'ul^2$



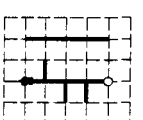
$h_4(\eta) = lu^2r'ul^2d^5r^2$



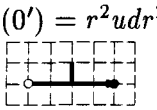
$h_4(\zeta_0) = lduluu'r^2l^4r^2d'^2ldul$



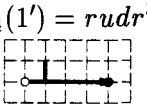
$h_4(\zeta_1) = ldulduluu'r^3l^4rd'^2l$



$h_4(0') = r^2udr^2$



$h_4(1') = rudr^3$



$h_4(\theta) = r^3u$

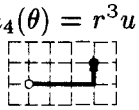


FIG. 4.6 - *Homomorphisme $h_4$ (existence d'un mot représentant une figure non-connexe)*

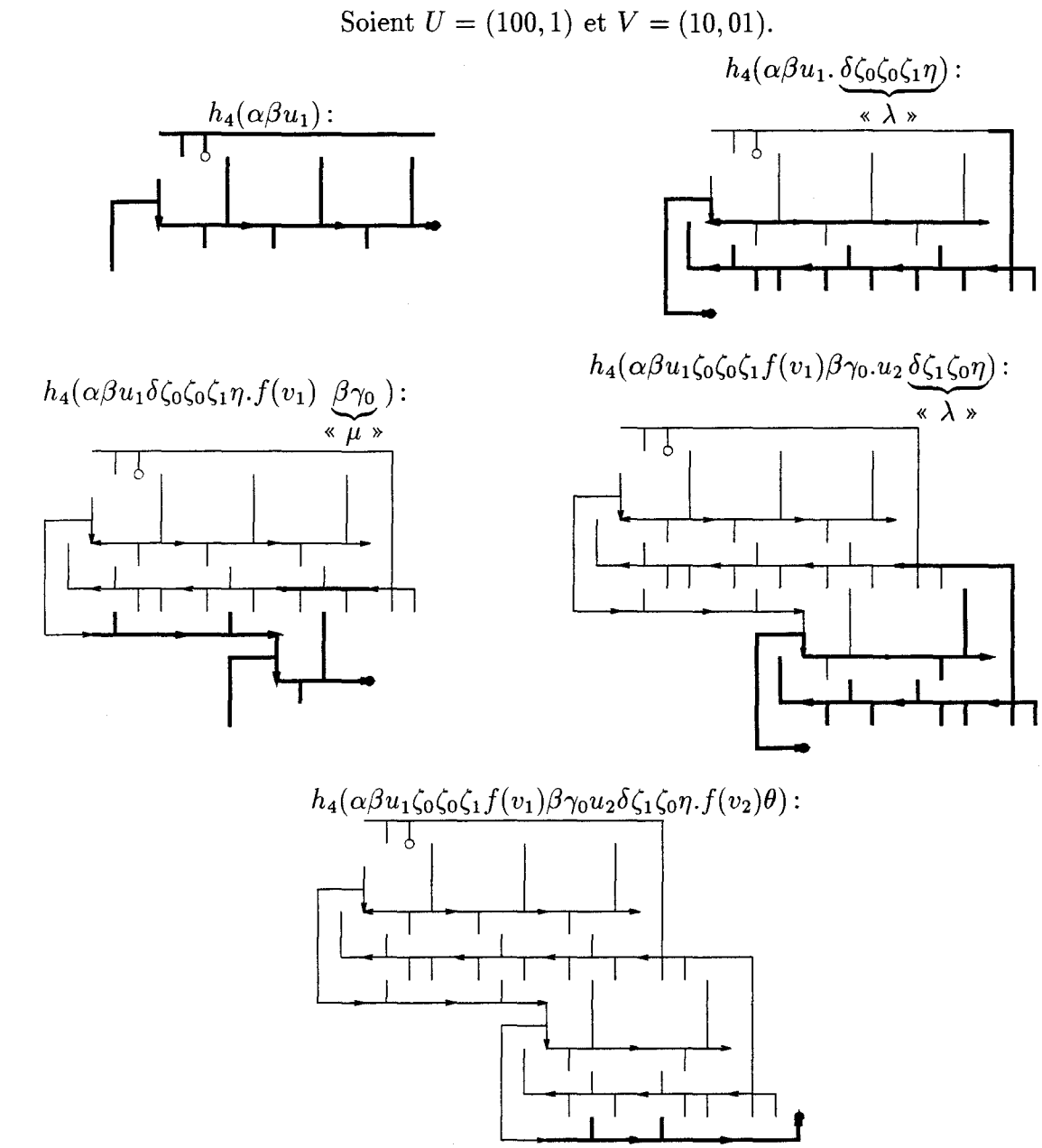



FIG. 4.7 - *Exemple de mise en correspondance pour l'existence d'une figure non-connexe*

## 4.5 Langages de figures de pixels

Nous utilisons donc la sémantique dont les notions de base sont décrites dans le Chapitre 1. En changeant la sémantique de l'alphabet $\Pi$, nous cherchons à décrire des figures composées de pixels (carrés unitaires) et non plus de segments. Pour ceci, les lettres de $\Pi$ induisent toujours des déplacements unitaires mais on ne trace plus le segment sous le déplacement mais le pixel qui se trouve à droite du mouvement. Cette sémantique est à même de décrire des polyominos avec ou sans trous alors que les mots de contour de polyomino rencontrés plus haut ne permettent que de décrire des polyominos sans trous (on parle alors de *polygone*).

**Proposition 4.7** *Soit $L$ un langage rationnel sur $\Pi$. Il est indécidable de savoir si $L$ contient un mot représentant un polyomino.*

*Preuve.* Soit $(U, V)$ une instance du VPCP, on considère le langage $L = h_5(K_{(U,V)})$ où $h_5$ est l'homomorphisme de $\Sigma^*$ dans $\Pi^*$ décrit Figure 4.8.

Un exemple de mise en correspondance est donné Figure 4.9. □

Ceci n'est somme toute pas surprenant au vu de la Proposition 4.4. Plus étonnant est le résultat suivant qui montre également l'indécidabilité de l'existence d'un polyomino convexe dans un rationnel alors que ce problème est décidable pour les mots de contour de polyomino convexe [BLS92].

**Proposition 4.8** *Soit $L$ un langage rationnel sur $\Pi$. Il est indécidable de savoir si $L$ contient un mot représentant un polyomino convexe.*

*Preuve.* Soit $(U, V)$ une instance du VPCP, on considère le langage $L = h_6(K_{(U,V)})$ où $h_6$ est l'homomorphisme de $\Sigma^*$ dans $\Pi^*$ décrit Figure 4.10.

Un exemple de mise en correspondance est donné Figure 4.11. □

Avec des homomorphismes similaires, on peut également montrer l'indécidabilité de l'existence d'un polyomino horizontalement (verticalement) convexe, d'un polygone ou encore, comme dans la Proposition 4.4, de l'existence d'un mot minimal dans un langage de figures de pixels rationnel.

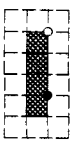
## 4.6 Remarques

La méthode que nous exposons permet donc de montrer l'indécidabilité de nombreux problèmes de propriétés existentielles dans différentes sémantiques et ceci grâce à l'image du même langage rationnel $K_{(U,V)}$ par différents homomorphismes.

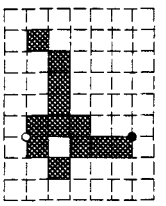
Notons que cette méthode ne peut s'appliquer pour les langages ruban dont nous parlons à la fin de la Section 4.2 car les langages que nous produisons ne peuvent être bornés dans

$h_5(\alpha) = \varepsilon$

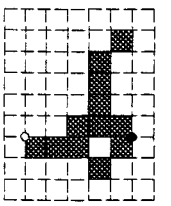
$h_5(\beta) = d^3lr$



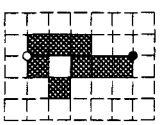
$h_5(\gamma_0) = rud^2ru^2lu^3lurdrd^4r^3$



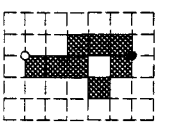
$h_5(\gamma_1) = r^3u^4rurdld^3ld^2ru^2dr$



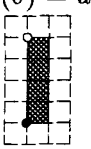
$h_5(0) = rud^2ru^2dr^3$



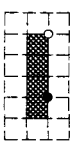
$h_5(1) = r^3ud^2ru^2dr$



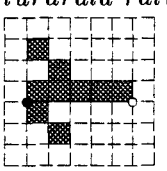
$h_5(\delta) = d^4$



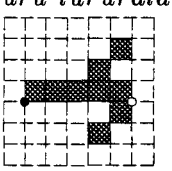
$h_5(\eta) = d^3lr$



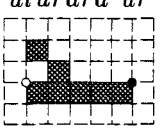
$h_5(\zeta_0) = l^4u^2lurdrdld^2rdlulu$



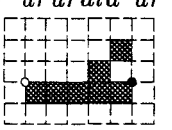
$h_5(\zeta_1) = dldluru^2lururdld^2l^4$



$h_5(0') = rulurdrd^2ur^3$



$h_5(1') = r^3ururdld^2ur$



$h_5(\theta) = \varepsilon$

FIG. 4.8 - *Homomorphisme $h_5$ (existence d'un mot représentant un polyomino)*

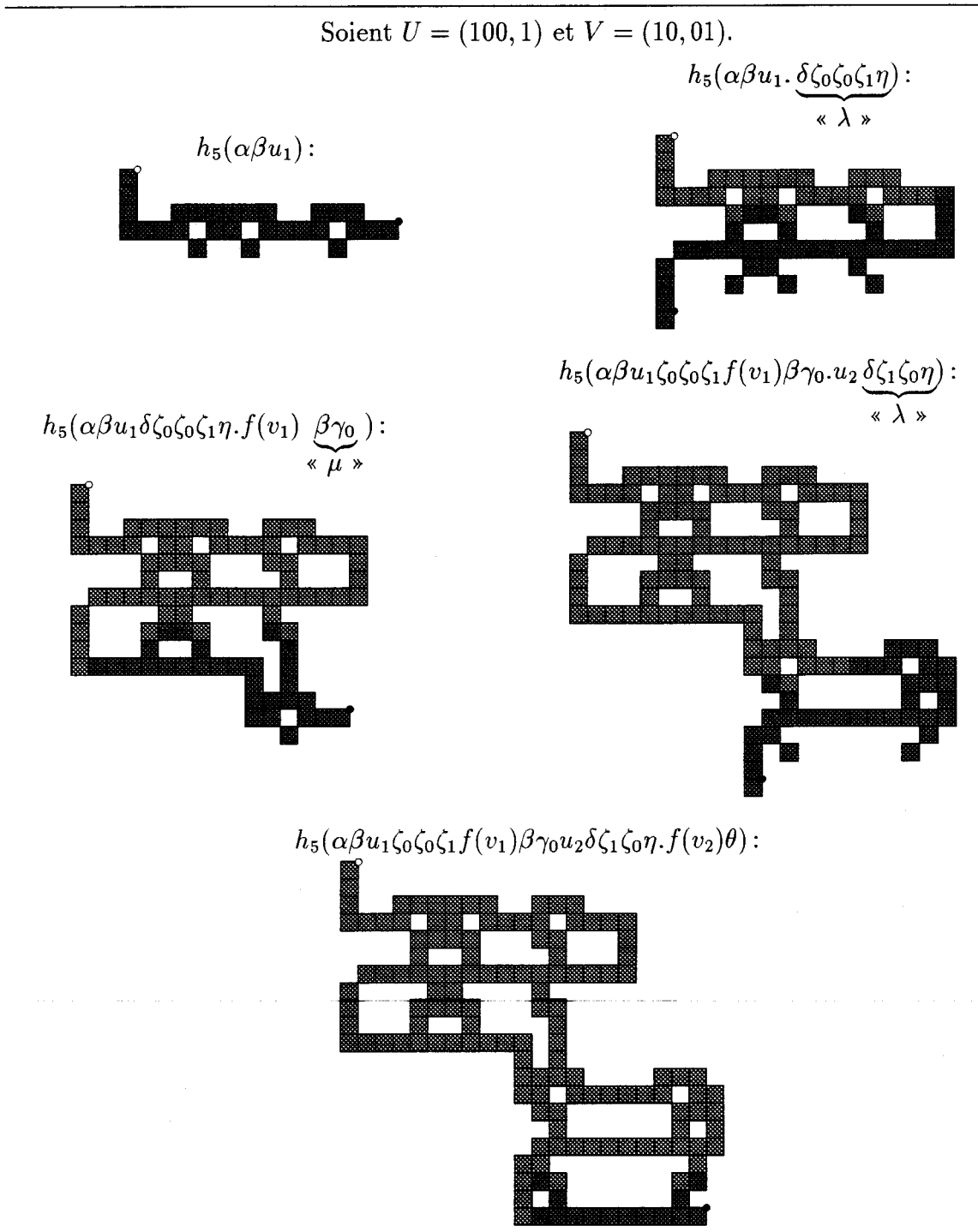



FIG. 4.9 - *Exemple de mise en correspondance pour l'existence d'un polyomino*

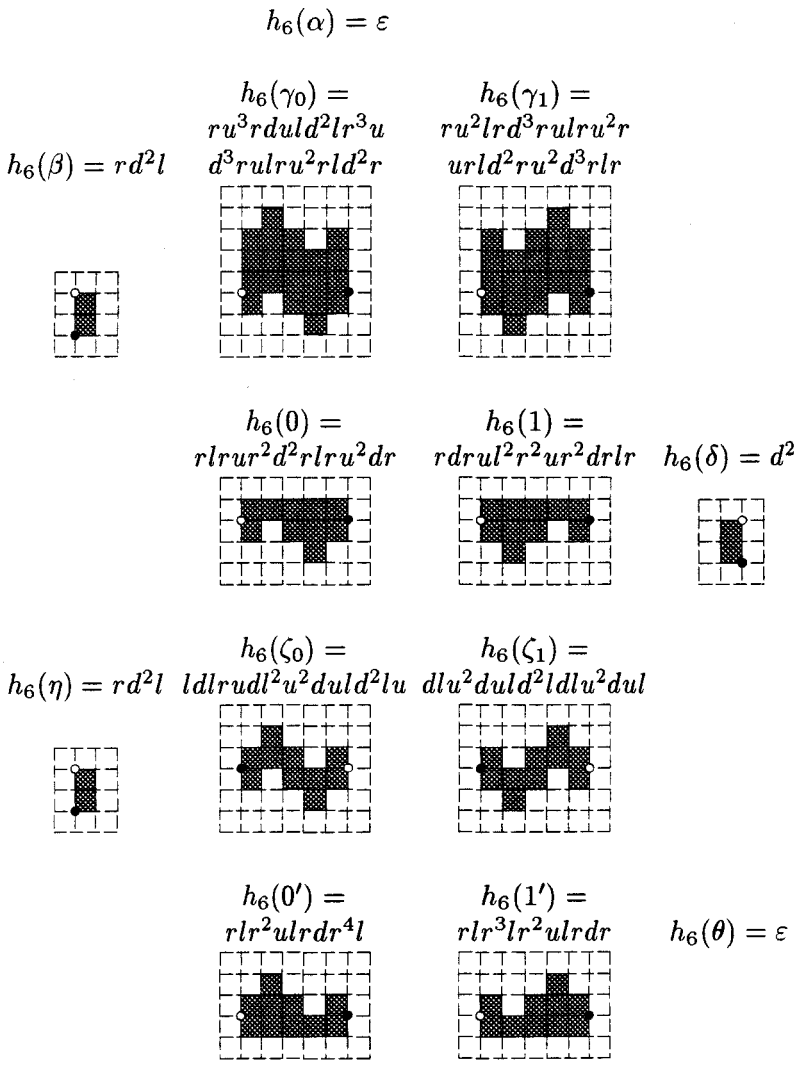



FIG. 4.10 - *Homomorphisme $h_6$ (existence d'un mot représentant un polyomino convexe)*

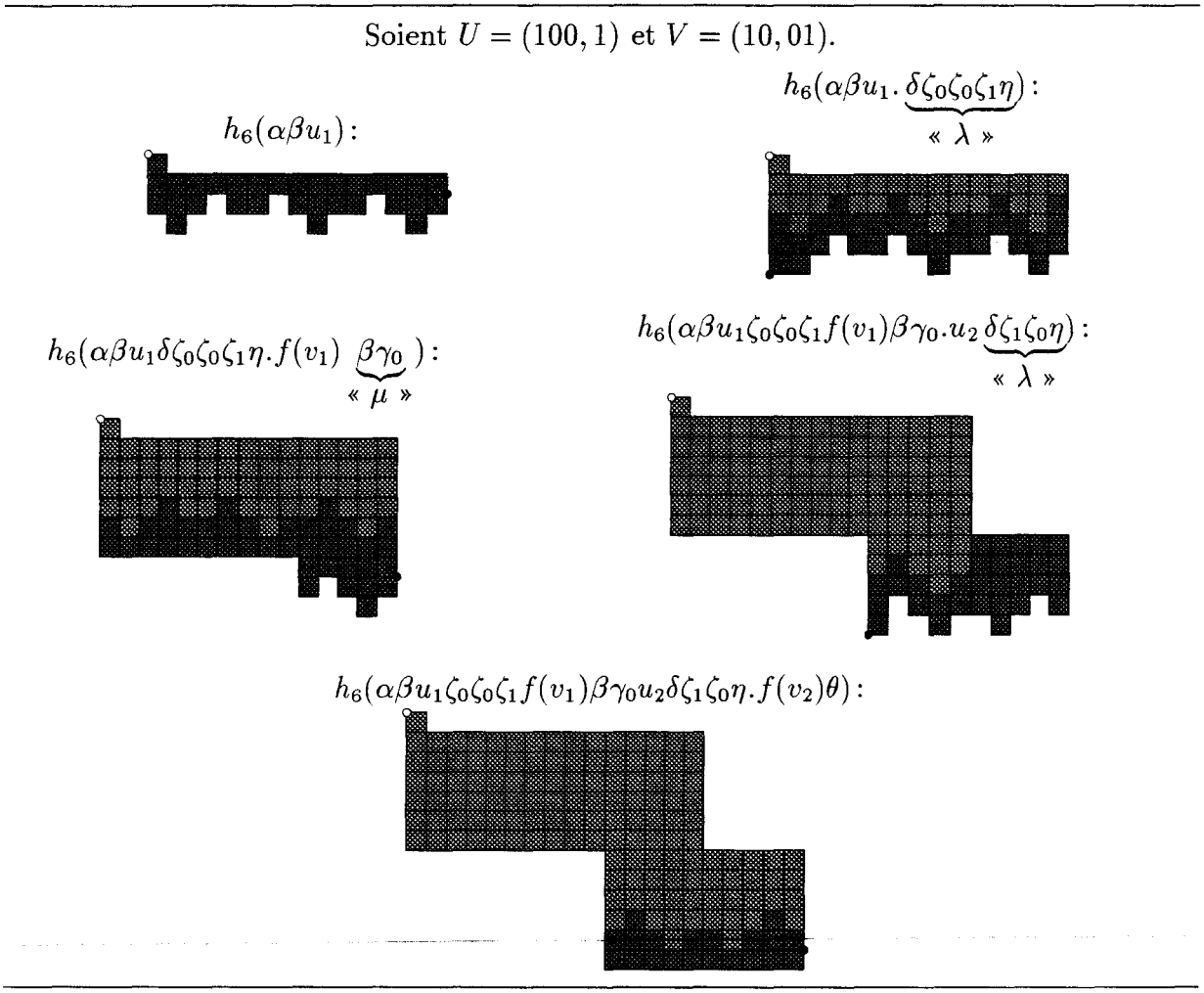



FIG. 4.11 - *Exemple de mise en correspondance pour l'existence d'un polyomino convexe*

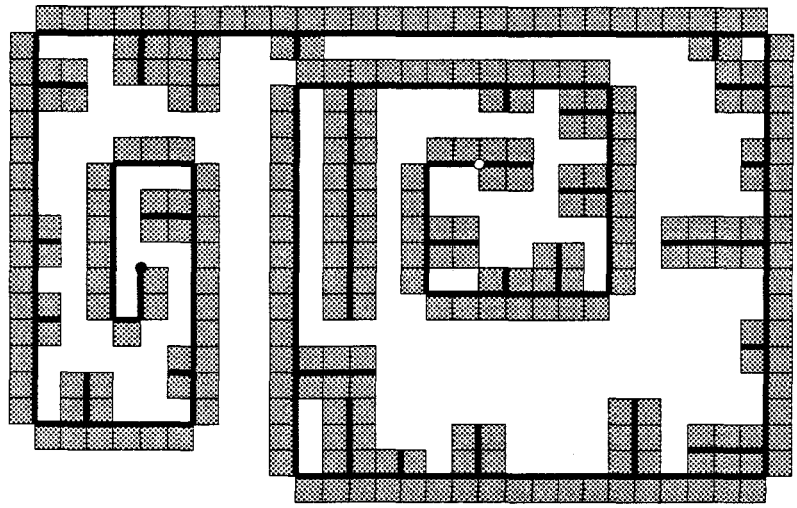


FIG. 4.12 - *Exemple de mot optimal*

aucune direction. Il est d'ailleurs intéressant que dans le cas des langages ruban, certains de ces problèmes deviennent décidables, comme par exemple contenir un mot de contour de polyomino.

Il est également intéressant de noter les différences entre les langages de figures avec segments et les langages à pixels. On a vu que l'existence d'un (mot de contour de) polyomino, ainsi que l'existence d'un mot minimal étaient indécidables dans les deux sémantiques, mais que l'existence d'un (mot de contour de) polyomino convexe était décidable dans les segments mais indécidable dans les pixels.

Enfin, on sait que l'existence d'un mot optimal est indécidable dans les segments et on peut montrer que ce problème est décidable dans les pixels [Sim]. En fait, l'ensemble des mots optimaux est beaucoup plus restreint dans les pixels que dans les segments. Par exemple, il est facile de voir qu'un mot ayant un facteur du type $r.a$ (avec $a \in \Pi$) ne peut être optimal que si $a \in \{r, u, l\}$. En effet, si $a = d$, le $d$ allume la même lettre que le $r$. Enfin, les facteurs d'un mot optimal ayant la forme $r.u^m.d^n.r$ (avec $m, n \in \mathbb{N}$) vérifient obligatoirement $m = n$. On peut en conclure que les mots optimaux ont nécessairement la forme d'une double spirale comme montré Figure 4.12. On peut trouver une étude concernant les mots optimaux décrivant des polygones dans [Rob96]. En considérant un mot optimal appartenant à un langage rationnel, on peut montrer qu'il existe dans ce langage rationnel un mot optimal dont la taille est bornée par une fonction dépendant du nombre d'états de l'automate minimal associé au langage.

# Références

[Bea91] D. Beauquier. An undecidable problem about rational sets and contour words

of polyominoes. *Information Processing Letters*, 37:257–263, 1991.

[BLS92] D. Beauquier, M. Latteux, and K. Slowinski. A decidability result about convex polyominoes. In I. Simon, editor, *Proc. Latin American Theoretical INformatics*, volume 583 of *Lecture Notes in Computer Science*, pages 32–45, Sao Paulo, Brazil, 1992. Springer-Verlag, Berlin.

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

[BN90] D. Beauquier and M. Nivat. Tiling the plane with one tile. In *Annual ACM Symposium on Computational Geometry*, 1990.

[BN91] D. Beauquier and M. Nivat. On translating one polyomino to tile the plane. *Discrete and Computational Geometry*, 6, 1991.

[BPS94] E. Barcucci, R. Pinzani, and R. Sprugnoli. The random generation of directed animals. *Theoretical Computer Science*, 127(2):333–350, 1994.

[Das89] J. Dassow. Graph-theoretical properties and chain code picture languages. *Journal of Information Processing and Cybernetics EIK*, 25:423–433, 1989.

[DH93] J. Dassow and F. Hinz. Decision problems and regular chain code picture languages. *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 45, 1993.

[DV84] M.-P. Delest and X. Viennot. Algebraic languages and polyominoes enumeration. *Theoretical Computer Science*, 34(1/2):169–206, 1984.

[Gol70] S. W. Golomb. Tiling with sets of polyominoes. *Journal of Combinatorial Theory*, 9:60–71, 1970.

[HW88] F. Hinz and E. Welzl. Regular chain code picture languages with invisible lines. Technical Report 252, I.I.G., Techn. Univ. Graz, Austria, 1988.

[KS87] C. Kim and I. H. Sudborough. The membership and equivalence problems for picture languages. *Theoretical Computer Science*, 52(3):177–191, 1987.

[LT90] M. Latteux and P. Turakainen. On characterizations of recursively enumerable languages. *Acta Informatica*, 28(2):179–186, 1990.

[MRW82] H. A. Maurer, G. Rozenberg, and E. Welzl. Using string languages to describe picture languages. *Information and Control*, 54(3):155–185, 1982.

[Pos47] E. L. Post. Recursive unsolvability of a problem of Thue. *Journal of Symbolic Logic*, 12:1–11, 1947.

[Rob96] D. Robilliard. *Langages de Figures.* PhD thesis, Univ. Lille 1, France, January 1996.

[RS96] D. Robilliard and D. Simplot. Undecidability of existential properties in picture languages. Technical Report it-299, L.I.F.L., Univ. Lille 1, France, December 1996. soumis à *Theoretical Computer Science.*

[Sim] D. Simplot. A decidability result about optimal words in pixel picture languages. Technical report, L.I.F.L., Univ. Lille 1, France. en préparation.

[SW85] I. H. Sudborough and E. Welzl. Complexity and decidability for chain code picture languages. *Theoretical Computer Science*, 36(2/3):173–202, 1985.

# Deuxième partie

# Langages à deux dimensions

# Chapitre 1

# Mots à deux dimensions

Les ensembles de mots locaux jouent un rôle considérable dans la théorie des langages de mots reconnaissables. Par exemple, il est bien connu que tout sous-ensemble de $\Sigma^+$ reconnaissable peut être obtenu comme l'image d'un ensemble local par un homomorphisme lettre-à-lettre [Eil74]. Un ensemble local $L$ sur un alphabet $\Delta$ peut être défini par un sous-ensemble $A$ de $(\Delta \cup \{\#\})^2$ qui indique les consécutions autorisées de lettres dans un mot de $L$ : $\omega$ est dans $L$ si et seulement si tout facteur de longueur deux dans $\#\omega\#$ appartient à $A$. Ces facteurs peuvent être vus comme des dominos qui permettent de « scanner » $\#\omega\#$.

Il existe plusieurs extensions dans le cas à deux dimensions de la notion de reconnaissabilité bien connue dans le monoïde libre. La première d'entres elles utilise des machines à deux dimensions comme dans les travaux de M. Blum et C. Hewitt [BH67] ou K. Inoue et I. Takanami [IT90]. A. Rosenfeld [Ros79] et G. Siromoney et al. [SSK73] donnent des extensions qui traitent des grammaires à deux dimensions. Récemment D. Giammarresi et A. Restivo [GR92] donnent une définition plaisante de reconnaissabilité dans les figures. Une figure est un mot à deux dimensions sur un alphabet fini. Par extension des langages de mots locaux, un langage de figures local est défini par un ensemble de pavés autorisés. Les langages de figures reconnaissables sont alors définis comme la projection d'un langage de figures local.

Cette classe de langages de figures est intéressante car elle correspond également à la classe des langages de figures reconnaissables par un cas particulier d'automates cellulaires appelés « on-line tesselation automata » ou « automate mosaïque en ligne » [IN77] et aux langages de figures définissables par des expressions existentielles en logique monadique du second-ordre [GRST96]. Une synthèse des recherches sur ce sujet se trouve dans le *handbook of formal languages* [GR96].

## 1.1 Notions de base

Soit $\Sigma$ un alphabet fini. Une figure sur $\Sigma$ est un tableau à deux dimensions rectangulaire de lettres de $\Sigma$. L'ensemble de toutes les figures sur $\Sigma$ est noté $\Sigma^{**}$.

Si $f$ est une figure, $\mathrm{lig}(f)$ et $\mathrm{col}(f)$ dénotent respectivement le nombre de lignes et le

nombre de colonnes de $f$. Notons que nous ne considérons que des tableaux non-vides : le nombre de colonnes et de lignes sont toujours plus grands que zéro. La taille de $f$, notée taille($f$), est le couple $(\text{lig}(f), \text{col}(f))$. Pour tout $1 \leq i \leq \text{lig}(f)$ et tout $1 \leq j \leq \text{col}(f)$, $f(i,j)$ représente la lettre (de $\Sigma$) qui se trouve sur la $i^e$ ligne et $j^e$ colonne (en partant du coin supérieur gauche). L'ensemble de toutes les figure sur $\Sigma$ de taille $(m,n)$ est noté $\Sigma^{m,n}$.

Si $f$ est une figure sur $\Sigma$ de taille $(m,n)$, nous notons $\tilde{f}$ la figure de taille $(m+2, n+2)$ sur $\Sigma \cup \{\#\}$ où $\#$ est une lettre spéciale qui n'appartient pas à $\Sigma$ et telle que :

1. $\forall 1 \leq i \leq m+2 \quad \tilde{p}(i,1) = \tilde{p}(i,n+2) = \#$
2. $\forall 1 \leq j \leq n+2 \quad \tilde{p}(1,j) = \tilde{p}(m+2,j) = \#$
3. $\forall 2 \leq i \leq m+1, 2 \leq j \leq n+1 \quad \tilde{p}(i,j) = p(i-1,j-1)$

Par exemple, si nous considérons la figure $f$ de taille $(3,4)$ sur $\Sigma = \{0,1,2\}$ :

$$f = \begin{array}{|c|c|c|c|}\hline 1&0&2&0\\\hline 0&1&0&0\\\hline 0&0&2&1\\\hline\end{array} \qquad \tilde{f} = \begin{array}{|c|c|c|c|c|c|}\hline \#&\#&\#&\#&\#&\#\\\hline \#&1&0&2&0&\#\\\hline \#&0&1&0&0&\#\\\hline \#&0&0&2&1&\#\\\hline \#&\#&\#&\#&\#&\#\\\hline\end{array}$$

Si $f$ est une figure sur $\Sigma$ de taille $m,n$, pour $r \leq m$ et $s \leq n$, $T_{r,s}(f)$ est l'ensemble des sous-figures de $f$ de taille $(r,s)$. Nous définissons :

$$T_{r,s}(f) = \left\{ q \in \Sigma^{r,s} \mid \begin{array}{c} \exists 0 \leq x \leq m-r, 0 \leq y \leq n-s \quad \forall 1 \leq i \leq r, 1 \leq j \leq s \\ q(i,j) = f(x+i, y+j) \end{array} \right\}$$

Un langage de figures sur $\Sigma$ est un sous-ensemble de $\Sigma^{**}$. Soit $L$ un langage de figures. Nous définissons $T_{r,s}(L) = \bigcup_{f \in L} T_{r,s}(f)$.

**Définition 1.1** *Soit $L$ un langage de figures sur $\Sigma$, $L$ est local s'il existe un ensemble $\Delta$ de figures de taille $(2,2)$ sur $\Sigma \cup \{\#\}$ tel que $L = \{p \in \Sigma^{**} \mid T_{2,2}(\tilde{p}) \subseteq \Delta\}$.*

*La classe des langages de figures locaux sur un alphabet $\Sigma$ est notée* $\text{Loc}(\Sigma^{**})$.

*Exemple. Nous considérons le langage de figures $L$ sur $\Sigma = \{0,1,2\}$ de toutes les lignes horizontales d'épaisseur deux obtenues par concaténation du carré :*

$$\begin{array}{|c|c|}\hline 1&0\\\hline 2&0\\\hline\end{array}$$

$$L = \left\{ \begin{array}{|c|c|}\hline 1&0\\\hline 2&0\\\hline\end{array}, \begin{array}{|c|c|c|c|}\hline 1&0&1&0\\\hline 2&0&2&0\\\hline\end{array}, \begin{array}{|c|c|c|c|c|c|}\hline 1&0&1&0&1&0\\\hline 2&0&2&0&2&0\\\hline\end{array}, \ldots \right\}$$

*Ce langage de figures est local et on peut lui associer l'ensemble de carrés* $\Delta$ *:*

$$\Delta = \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 0 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & 1 \\ \hline \# & 2 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 2 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 0 & 2 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & \# \\ \hline 0 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & 2 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 2 & 0 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 2 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & \# \\ \hline \# & \# \\ \hline \end{array} \right\}$$

*Exemple. De manière moins triviale, l'ensemble des figures qui représentent une partie du triangle de Pascal modulo n est également un langage de figures local. Prenons par exemple les triangles de Pascal modulo* 2*, on travaille donc sur l'alphabet* $\Sigma = \{0, 1\}$*. Le contrôle est clairement local puisque pour une figure* $f$ *de taille* $(k, l)$ *on doit simplement avoir :*

$$\begin{gathered} \forall 1 \leq i \leq k \quad f(i,1) = 1 \\ \forall 1 < i \leq l \quad f(1,i) = 0 \\ \forall 1 < i \leq k, 1 < j \leq l \quad f(i,j) = (f(i-1,j-1) + f(i-1,j)) \bmod 2 \end{gathered}$$

*Cette ensemble de figures contient des figures du type :*

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | **1** | **1** | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | **1** | 0 | 0 | **1** | **1** | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | **1** | 0 | **1** | 0 | **1** | 0 | 0 | 0 | 0 |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | 0 | 0 | 0 |
| **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 |
| **1** | **1** | 0 | 0 | 0 | 0 | 0 | 0 | **1** | **1** | 0 |
| **1** | 0 | **1** | 0 | 0 | 0 | 0 | 0 | **1** | 0 | **1** |
| **1** | **1** | **1** | **1** | 0 | 0 | 0 | 0 | **1** | **1** | **1** |
| **1** | 0 | 0 | 0 | **1** | 0 | 0 | 0 | **1** | 0 | 0 |

*On peut lui associer l'ensemble de pavés* $(2,2)$ $\Delta$ *défini par* $\Delta = \Delta_1 \cup \Delta_2 \cup \Delta_3 \cup \Delta_4 \cup \Delta_5$ *:*

$$\Delta_1 = \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 1 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 0 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 0 & 0 \\ \hline \end{array} \right\}$$

$$\Delta_2 = \left\{ \begin{array}{|c|c|}\hline \# & 1 \\ \hline \# & a \\ \hline \end{array} ;\ a \in \{1, \#\} \right\}$$

$$\Delta_3 = \left\{ \begin{array}{|c|c|}\hline 0 & 0 \\ \hline a & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline a & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 0 \\ \hline a & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 1 \\ \hline a & 0 \\ \hline \end{array} ;\ a \in \Sigma \right\}$$

$$\Delta_4 = \left\{ \begin{array}{|c|c|}\hline a & \# \\ \hline b & \# \\ \hline \end{array} ;\ a \in \Sigma\ b \in \Sigma \cup \{\#\} \right\}$$

$$\Delta_5 = \left\{ \begin{array}{|c|c|}\hline a & b \\ \hline \# & \# \\ \hline \end{array} ;\ a, b \in \Sigma \right\}$$

Nous savons que tout langage de mots reconnaissable est l'image par un homomorphisme lettre-à-lettre d'un langage local. Nous avons donc besoin de définir une projection dans les figures. Soient $\Sigma$ et $\Sigma'$ deux alphabets finis et $\pi : \Sigma \to \Sigma'$ une projection. La projection par $\pi$ d'une figure $f \in \Sigma^{**}$ est la figure $f' \in \Sigma'^{**}$ telle que taille$(f)$ = taille$(f')$ et que pour tout $1 \leq i \leq \text{lig}(f)$ et tout $1 \leq j \leq \text{col}(f)$ on ait $f'(i,j) = \pi(f(i,j))$. Nous notons $p' = \pi(p)$. Par extension, nous notons $\pi(L)$, la projection par $\pi$ du langage $L$ sur $\Sigma$ et $\pi(L) = \{p' \in \Sigma'^{**} \mid \exists p \in L \quad p' = \pi(p)\}$.

**Définition 1.2** *Soit $L$ un langage de figures sur $\Sigma$, $L$ est reconnaissable s'il existe un langage de figures local $L'$ sur un alphabet $\Sigma'$ et une projection $\pi : \Sigma' \to \Sigma$ tels que $L = \pi(L')$.*

*La classe des langages de figures reconnaissables sur un alphabet $\Sigma$ est noté* Rec$(\Sigma^{**})$.

*Exemple. Si nous regardons le langage de figures $K$ sur $\Gamma = \{a\}$ de toutes les lignes d'épaisseur deux et de longueur paire :*

$$K = \left\{ \begin{array}{|c|c|}\hline a & a \\ \hline a & a \\ \hline \end{array}, \begin{array}{|c|c|c|c|}\hline a & a & a & a \\ \hline a & a & a & a \\ \hline \end{array}, \begin{array}{|c|c|c|c|c|c|}\hline a & a & a & a & a & a \\ \hline a & a & a & a & a & a \\ \hline \end{array}, \ldots \right\}$$

*Grâce à la projection $\pi : \Sigma = \{0,1,2\} \to \Gamma$ telle que $\pi(0) = \pi(1) = \pi(2) = a$, il est facile de voir que $K$ est un langage de figures reconnaissable car $K = \pi(L)$, où $L$ est le langage de figures local donné en exemple précédemment.*

*Exemple. L'ensemble $C$ des carrés sur l'alphabet $\Sigma = \{a\}$ est également un langage de figures reconnaissable. On a :*

$$C = \left\{ \begin{array}{|c|}\hline a \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & a \\ \hline a & a \\ \hline \end{array}, \begin{array}{|c|c|c|}\hline a & a & a \\ \hline a & a & a \\ \hline a & a & a \\ \hline \end{array}, \begin{array}{|c|c|c|c|}\hline a & a & a & a \\ \hline a & a & a & a \\ \hline a & a & a & a \\ \hline a & a & a & a \\ \hline \end{array} \cdots \right\}$$

*En effet, $C$ est la projection d'un langage local $K \subseteq \{0,1\}^{**}$ pour lequel l'ensemble des pavés $2 \times 2$ autorisés est :*

$$\Delta = \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 0 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 0 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 1 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & 1 \\ \hline \# & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & 0 \\ \hline \# & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 0 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & \# \\ \hline 0 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & \# \\ \hline 1 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & 1 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & 0 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 0 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & \# \\ \hline \# & \# \\ \hline \end{array} \right\}$$

$$K = \left\{ \begin{array}{|c|}\hline 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|c|}\hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|c|c|}\hline 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \\ \hline \end{array} \cdots \right\}$$

*En posant $\pi(0) = \pi(1) = a$, on a bien $C = \pi(K)$ qui est donc bien un langage reconnaissable.*

Soit $\Delta$ un ensemble de figures de taille $(l,k)$ sur $\Sigma \cup \{\#\}$. Le langage de figures $L$ défini par :

$$L = \{f \in \Sigma^{**} \mid T_{l,k}(\tilde{f}) \subseteq \Delta\}$$

est $(l,k)$-localement testable et est clairement reconnaissable [GR96].

## 1.2 Propriétés de clotûre des reconnaissables

Avec les figures nous avons deux produits de concaténation. Soit $f$ une figure de taille $(n,m)$ et $f'$ une figure de taille $(n',m')$. La concaténation verticale de $f$ et $f'$, qui est notée $p \ominus p'$, est définie si et seulement si $m = m'$ et est la figure de taille $(n+n',m)$ satisfaisant :

$$\forall 1 \leq i \leq n \; \forall 1 \leq j \leq m \quad (f \ominus f')(i,j) = f(i,j)$$
$$\forall 1 \leq i \leq n' \; \forall 1 \leq j \leq m \quad (f \ominus f')(n+i,j) = f'(i,j)$$

De la même manière, la concaténation horizontale de $f$ et $f'$, qui est notée $p \oplus p'$, est définie si et seulement si $n = n'$ et est la figure de taille $(n, m+m')$ satisfaisant :

$$\forall 1 \leq i \leq n \; \forall 1 \leq j \leq m \quad (f \oplus f')(i,j) = f(i,j)$$
$$\forall 1 \leq i \leq n \; \forall 1 \leq j \leq m' \quad (f \oplus f')(i,m+j) = f'(i,j)$$

*Exemple. Soient $f$ et $g$ les deux figures sur $\Sigma = \{a,b,c\}$ :*

$f =$

| | | | | |
|---|---|---|---|---|
| $b$ | $a$ | $a$ | $b$ | $a$ |
| $c$ | $b$ | $c$ | $b$ | $b$ |
| $a$ | $b$ | $b$ | $a$ | $c$ |

$g =$

| | |
|---|---|
| $c$ | $a$ |
| $b$ | $c$ |
| $c$ | $a$ |

*Comme* $\mathrm{lig}(f) = \mathrm{lig}(g)$, *$f \oplus g$ est définie :*

$f \oplus g =$

| | | | | | | |
|---|---|---|---|---|---|---|
| $b$ | $a$ | $a$ | $b$ | $a$ | $c$ | $a$ |
| $c$ | $b$ | $c$ | $b$ | $b$ | $b$ | $c$ |
| $a$ | $b$ | $b$ | $a$ | $c$ | $c$ | $a$ |

*Par contre $f \ominus g$ n'est pas définie puisque* $\mathrm{col}(f)$ *et* $\mathrm{col}(g)$ *sont différents.*

Ces concaténations sont étendues de manière usuelle aux langages, soient $L$ et $K$ deux langages de figures :

$$L \ominus K = \{f \ominus g \mid f \in L \wedge g \in K\}$$
$$L \oplus K = \{f \oplus g \mid f \in L \wedge g \in K\}$$

D. Giammarresi et A. Restivo ont ainsi montré que la classe des langages de figures reconnaissables était close par concaténation horizontale et verticale.

**Proposition 1.3 (D. Giammarresi et A. Restivo 1992 [GR92])** *Soient $L$ et $K$ des langages de figures reconnaissables sur $\Sigma$, on a :*

$$L \oplus K \in \mathrm{Rec}(\Sigma^{**})$$
$$L \ominus K \in \mathrm{Rec}(\Sigma^{**})$$

Ainsi nous disposons de deux opérations « étoiles », la première utilise la concaténation verticale et est notée $^{\ominus *}$, tandis que la seconde utilise la concaténation horizontale et est notée $^{\oplus *}$. Soit $L$ un langage de figures, on a :

$$\begin{array}{rclrcl} L^{\ominus 1} & = & L & L^{\oplus 1} & = & L \\ L^{\ominus i+1} & = & L \ominus L^{\ominus i} & L^{\oplus i+1} & = & L \oplus L^{\oplus i} \\ L^{\ominus *} & = & \bigcup_{i\geq 1} L^{\ominus i} & L^{\oplus *} & = & \bigcup_{i\geq 1} L^{\ominus i} \end{array}$$

Ainsi, le langage $K$ donné en exemple précédemment peut être défini par :

$$K = \begin{array}{|c|c|}\hline a & a \\ \hline a & a \\ \hline \end{array}^{\oplus *}$$

Les langages de figures reconnaissables sont également clos par étoile horizontale et verticale :

**Proposition 1.4 (D. Giammarresi et A. Restivo 1992 [GR92])** *Soit $L$ un langage de figures reconnaissable sur $\Sigma$, on a :*

$$\begin{array}{rcl} L^{\oplus *} & \in & \mathrm{Rec}(\Sigma^{**}) \\ L^{\ominus *} & \in & \mathrm{Rec}(\Sigma^{**}) \end{array}$$

En utilisant les mêmes arguments que dans les mots, on voit facilement que la classe des langages de figures reconnaissables contient les langages finis et close par union et intersection [GR92]. Néanmoins, si l'on note $\mathrm{Rat}(\Sigma^{**})$ la plus petite classe contenant les langages finis et close par union, concaténation et étoile, on n'obtient pas $\mathrm{Rec}(\Sigma^{**})$.

**Proposition 1.5 (D. Giammarresi et A. Restivo 1996 [GR96])** *La classe des langages de figures rationnels* $\mathrm{Rat}(\Sigma^{**})$ *est incluse proprement dans la classe de langages de figures reconnaissables* $\mathrm{Rec}(\Sigma^{**})$.

Plusieurs tentatives pour obtenir la classe de reconnaissables à partir d'expressions régulières ont été lancées mais n'ont abouties qu'à des inclusions strictes [Mat97]. Néanmoins, si l'on s'autorise à utiliser la projection on réussit à obtenir les reconnaissables. Ainsi, les projections des langages représentés par les expressions régulières qui contiennent les concaténations $\ominus, \oplus$, les étoiles $\ominus *, \oplus *$, l'union et l'intersection, on obtient exactement les langages de figures reconnaissables [GR96].

Par définition des reconnaissables, on obtient immédiatement la clotûre par projection et projection inverse.

**Proposition 1.6** *1. Soient $L \in \mathrm{Rec}(\Sigma^{**})$ et $\pi$ une projection de $\Sigma$ dans $\Sigma'$, on a $\pi(L) \in \mathrm{Rec}(\Sigma'^{**})$.*

*2. Soient $L \in \mathrm{Rec}(\Sigma^{**})$ et $\pi$ une projection de $\Sigma'$ dans $\Sigma$, on a $\pi^{-1}(L) \in \mathrm{Rec}(\Sigma'^{**})$.*

Enfin, on définit une nouvelle opération sur les figures que l'on appelle le produit cartésien que l'on note $\otimes$.

**Définition 1.7** *Soient $f$ une figure de $\Sigma^{**}$ et $g$ une figure de $\Sigma'^{**}$, le produit cartésien de $f$ et $g$, noté $f \otimes g$, n'est défini que si* taille$(f)$ = taille$(g)$ *et correspond alors à la figure sur $\Sigma \times \Sigma'$ satisfaisant :*

$$f \otimes g = h \in (\Sigma \times \Sigma')^{**} \text{ telle que } \begin{cases} \text{taille}(h) = \text{taille}(f) = \text{taille}(g) \\ \forall 1 \leq i \leq \text{lig}(f)\ \forall 1 \leq i \leq \text{lig}(f) \quad h(i,j) = (f(i,j), g(i,j)) \end{cases}$$

Le produit cartésien est étendu de manière canonique aux ensembles de figures et on a la propriété de clotûre suivante :

**Proposition 1.8** *Soient deux langages de figures $L \in \text{Rec}(X^{**})$ et $K \in \text{Rec}(Y^{**})$, on a $L \otimes K \in \text{Rec}((X \times Y)^{**})$.*

*Preuve.* Soient $A \in \text{Loc}(X'^{**})$ et $B \in \text{Loc}(Y'^{**})$ les langages locaux associés à $L$ et $K$ respectivement tels que $L = \pi(A)$ et $K = \sigma(B)$. On considère les ensembles de pavés $\Delta_a$ et $\Delta_b$ autorisés dans $A$ et $B$ respectivement. Il suffit de construire l'ensemble de pavés autorisés :

$$\Theta' = \left\{ \begin{array}{|c|c|}\hline (x_a,x_b) & (y_a,y_b) \\ \hline (z_a,z_b) & (t_a,t_b) \\ \hline \end{array} \mid \begin{array}{|c|c|}\hline x_a & y_a \\ \hline z_a & t_a \\ \hline \end{array} \in \Delta_a \wedge \begin{array}{|c|c|}\hline x_b & y_b \\ \hline z_b & t_b \\ \hline \end{array} \in \Delta_b \right\}$$

Cet ensemble $\Theta'$ contient des couples du type $(\#, a)$, c'est pourquoi on lui applique une intersection et une projection pour obtenir $\Theta$ :

$$\Theta = \alpha(\Theta' \cap (X' \times Y' \cup \{(\#,\#)\})^{2,2})$$

où $\alpha$ est l'identité pour les lettres de $X' \times Y'$ et associe $\#$ à $(\#,\#)$. Enfin, on considère le langage local $M$ associé à $\Theta$ et la projection $\rho$ de $X' \times Y'$ dans $X \times Y$ définie par $\rho((x,y)) = (\pi(x), \sigma(y))$. Il est facile de voir que $L \otimes K = \rho(M)$ qui est donc reconnaissable. □

Néanmoins, les reconnaissables ne sont pas clos par complémentation :

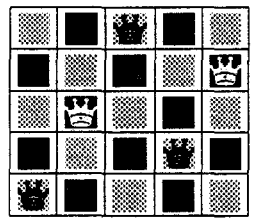
**Proposition 1.9 (D. Giammarresi et A. Restivo 1992 [GR92])** *La famille des langages de figures reconnaissables n'est pas close par complémentation.*

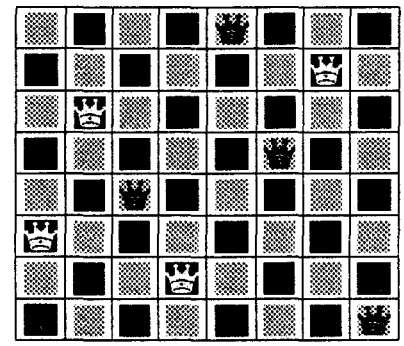
Dans la suite du mémoire, nous identifions $\Sigma^{\oplus *}$ (les figures avec une seule ligne) et le semi-groupe libre $\Sigma^+$.

## 1.3 Langage des $k$-reines

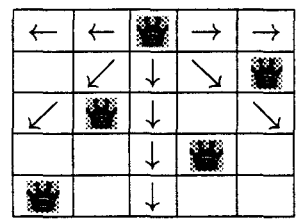
Avant de continuer, et afin de manipuler un peu les outils que nous venons de voir, nous allons traiter le problème des $k$-reines en montrant que l'ensemble des figures qui représentent des échiquiers de taille $k \times k$ où sont positionnées $k$ reines qui ne se mettent pas en échec est un langage de figures reconnaissable. On peut retrouver ce langage reconnaissable dans [LMN97] qui néanmoins ne traite que le cas des 8-reines.

Par exemple, avec l'alphabet $\Sigma = \{$ ■, ▒, ♛, ♔$\}$, voici deux figures qui appartiennent à ce langage que nous notons $L$ :

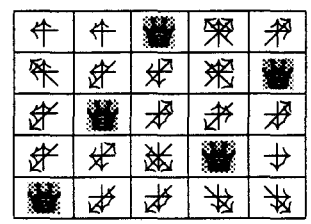




Nous allons un instant oublier les couleurs de l'échiquier pour nous intéresser uniquement à la position des reines. Dans le langage local associé, noté $L'_1$, il suffit de marquer les « cases attaquées » par les reines avec des flèches désignées par $X = \{\uparrow, \nearrow, \rightarrow, \searrow, \downarrow, \swarrow, \leftarrow, \nwarrow\}$. À titre d'exemple, les attaques de la reine en haut de l'échiquier $5 \times 5$ sont notées :



Comme une case peut être attaquée par plusieurs reines, on utilise en fait les parties de $X$ que l'on note $Y = 2^X$. Pour l'échiquier de l'exemple précédent, on a en fait :



Il suffit donc de vérifier que de chaque reine partent bien toutes les flèches, que toute flèche provient d'une flèche dans la même direction ou d'une reine et qu'aucune flèche ne pointe vers une reine. Tout ceci se fait de manière locale. Notons $L_1$ l'image de ce local par la projection qui associe une case blanche aux lettres de $Y$ et une reine à une reine (on

note $\Sigma' = \{$▒, ♛$\}$) ; ce langage contient toutes les figures de $\Sigma'^{**}$ où aucune reine n'est en échec et est reconnaissable.

Néanmoins, reste à être sûr que l'on a un carré et qu'il y a autant de reines que de lignes. Pour ceci, on va construire un langage $L_2$ qui contient tous les carrés qui ont une et une seule reine par colonne et par ligne. On va d'abord construire un langage $L'_2$ sur $\Sigma'$ qui contient toutes les figures où il y a une reine par ligne. Ce langage est la projection du langage local sur $Z = \{$♛$, a, b\}$ défini par :

$$\Delta'_2 = \left\{ \begin{array}{|c|c|}\hline \# & x \\ \hline \# & y \\ \hline \end{array} \; ; \; x,y \in \{\#, a, ♛\} \right\}$$

$$\cup \left\{ \begin{array}{|c|c|}\hline x & \# \\ \hline y & \# \\ \hline \end{array} \; ; \; x,y \in \{\#, b, ♛\} \right\}$$

$$\cup \left\{ \begin{array}{|c|c|}\hline x & y \\ \hline z & t \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline z & t \\ \hline \end{array}, \begin{array}{|c|c|}\hline x & t \\ \hline \# & \# \\ \hline \end{array} \; ; \; \begin{array}{c} x,y,z,t \in \{a,b,♛\} \\ x = a \Rightarrow y \in \{a,♛\} \quad z = a \Rightarrow t \in \{a,♛\} \\ x \in \{♛,b\} \Rightarrow y = b \quad z \in \{♛,b\} \Rightarrow t = b \end{array} \right\}$$

Les figures de ce local ont la forme suivante :

| | | | | | | |
|---|---|---|---|---|---|---|
| $a$ | ♛ | $a$ | $b$ | $b$ | $b$ | $b$ |
| ♛ | $b$ | $b$ | $b$ | $b$ | $b$ | $b$ |
| $a$ | ♛ | $b$ | $b$ | $b$ | $b$ | $b$ |
| $a$ | $a$ | $a$ | $a$ | $a$ | $a$ | ♛ |
| $a$ | $a$ | $a$ | $a$ | ♛ | $b$ | $b$ |

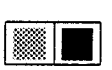
De même le langage $L''_2$ qui contient toutes les figures sur $\Sigma'$ où il y a exactement une reine par colonne est reconnaissable. Le langage $L_2$ est l'intersection de $L'_2$ et $L''_2$ et est donc reconnaissable (on verra dans la Section 2.3 comment on peut définir $L_2$ de manière plus concise). Ainsi, en faisant l'intersection de $L_1$ avec $L_2$, on obtient le langage $L_3$ qui contient tous les carrés de taille $n \in \mathbb{N}$ sur $\Sigma'$ qui ont $n$ reines qui ne se mettent pas en échec[1]. Il reste simplement à colorier l'échiquier.

Pour ceci, on va utiliser le produit cartésien de $L_3$ avec le langage $L_4$ qui contient tous les damiers (de taille supérieure à $(2,2)$) sur $\Sigma'' = \{$■, ▒$\}$. On pose :

$$L'_4 = \left( \begin{array}{|c|c|}\hline ▒ & ■ \\ \hline ■ & ▒ \\ \hline \end{array} \right)^{\oplus^*, \ominus^*}$$

1. notons également qu'en obligeant les lettres de $Y$ présentes dans une figure de $L'_1$ à avoir une flèche horizontale et une flèche verticale, on obtenait directement $L_3$.

Le langage $L'_4$ contient toutes les figures obtenues par concaténation du pavé donné dans l'expression, il contient donc tous les damiers de taille supérieure à $(2,2)$ mais dont les dimensions sont paires. Pour avoir $L_4$, il suffit de prendre :

$L_4 = L'_4 \cup \left( L'_4 \cup L'_4 \oplus$  $^{\ominus *} \right) \ominus \left($  $^{\oplus *} \cup$  $^{\oplus *} \oplus$  $\right)$

On pose $L_5 = L_3 \otimes L_4$ qui est donc reconnaissable puisque le produit cartésien de deux reconnaissables. On considère la projection $\pi$ de $\Sigma' \times \Sigma''$ sur $\Sigma$ :

$\pi($  $) =$ 

$\pi($  $) =$ 

$\pi($  $) =$ 

$\pi($  $) =$ 

On a donc $L = \pi(L_5)$ qui est donc bien un langage de figures reconnaissable.

# Références


[BH67] M. Blum and C. Hewitt. Automata on two-dimensional tape. In *Proc. IEEE 8th Annual Symposium on Switching and Automata Theory*, IEEE Symposium on Switching and Automata Theory, pages 155–160, 1967.

[Eil74] S. Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, New York, 1974.

[GR92] D. Giammarresi and A. Restivo. Recognizable picture languages. *International Journal Pattern Recognition and Artificial Intelligence*, pages 31–46, 1992. Special Issue on *Parallel Image Processing*. M. Nivat, A. Saoudi and P.S.P. Wangs (Eds.).

[GR96] D. Giammarresi and A. Restivo. Two-dimensional languages. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3, pages 215–267. Springer-Verlag, Berlin, 1996.

[GRST96] D. Giammarresi, A. Restivo, S. Seibert, and W. Thomas. Monadic second-order logic over rectangular pictures and recognizability by tiling systems. *Information and Computation*, 125(1):32–45, 1996.

[IN77] K. Inoue and A. Nakamura. Some properties of two-dimensional on-line tessellation acceptors. *Information Sciences*, 13:95–121, 1977.

[IT90] K. Inoue and I. Takanami. A survey of two-dimensional automata theory. In J. Dassow and J. Kelemen, editors, *Proc. Aspects and Prospects of Theoretical Computer Science, 5th International Meeting of Young Computer Scientists*, volume 381 of *Lecture Notes in Computer Science*, pages 72–91. Springer-Verlag, Berlin, 1990.

[LMN97] K. Lindgren, C. Moore, and M. Nordahl. Complexity of two-dimensional patterns. Technical Report 97–03–023, Santa Fe Institute, Unit. States, 1997.

[Mat97] O. Matz. Regular expressions and contextfree grammars for picture languages. In *Proc. STACS'97*, volume 1200 of *Lecture Notes in Computer Science*, pages 283–294, Lübeck, Germany, 1997. Springer-Verlag, Berlin.

[Ros79] A. Rosenfeld. *Picture Languages – Formal Models of Picture Recognition.* Academic Press, New York, 1979.

[SSK73] G. Siromoney, R. Siromoney, and K. Krithivasan. Picture languages with array rewriting rules. *Information and Control*, 22(5):447–470, June 1973.

# Chapitre 2

# Langages de figures reconnaissables et recouvrememement par des dominos

## 2.1 Introduction

Avec les pavés $2 \times 2$ utilisés dans la définition des langages de figures locaux, le contrôle horizontal et le contrôle vertical sont faits en même temps. Il est donc naturel de se demander si ces deux contrôles peuvent être faits séparément. Ainsi, nous définissons les langages de figures hv-locaux en remplaçant dans la définition des langages de figures locaux les pavés $2 \times 2$ par des dominos horizontaux et verticaux et nous prouvons que tout langage de figures reconnaissable peut être obtenu par la projection d'un langage de figure hv-local.

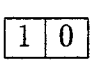
On peut retrouver ces travaux dans [LS97] et ils ont été introduits dans le chapitre du Handbook de D. Giammarresi et A. Restivo [GR96].

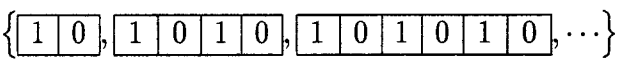
## 2.2 Langages de figures « hv-locaux »

Nous introduisons les langages de figures « hv-locaux » où les contrôles horizontaux et verticaux sont dissociés par l'utilisation de dominos à la place des carrés $2 \times 2$.

**Définition 2.1** *Soit $L$ un langage de figures inclus dans $\Sigma^{**}$. $L$ est hv-local s'il existe un ensemble $\Delta$ de dominos horizontaux et verticaux sur $\Sigma \cup \{\#\}$ tel que $L = \{q \in \Sigma^{**} \mid T_{1,2}(\tilde{q}) \cup T_{2,1}(\tilde{q}) \subseteq \Delta\}$.*

*Exemple. On considère le langage de figures $L$ sur $\Sigma = \{0, 1, 2\}$ de toutes les lignes horizontales d'épaisseur un obtenues par concaténation du domino :*




$L =$ 

*L est un langage de figures hv-local, car on peut lui associer l'ensemble de dominos* $\Delta$ *:*

$$\Delta = \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & \# \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & 0 \\ \hline 0 & \# \\ \hline \end{array} \right\}$$

**Proposition 2.2** *Soit* $L \subseteq \Sigma^{**}$ *un langage de figures. Si L est hv-local, alors L est local.*

*Preuve.* Soit $L \subseteq \Sigma^{**}$ un langage de figures hv-local. Nous construisons un langage de figures local $K$ et nous montrons que $L = K$.

Nous savons qu'il existe un ensemble $\Delta$ de dominos horizontaux et verticaux sur $\Sigma \cup \{\#\}$ ($\Delta \subseteq (\Sigma \cup \{\#\})^{1,2} \cup (\Sigma \cup \{\#\})^{2,1}$) tel que $L = \{f \in \Sigma^{**} \mid T_{1,2}(\tilde{f}) \cup T_{2,1}(\tilde{f}) \subseteq \Delta\}$.

Nous définissons un ensemble de carrés $\Delta'$ :

$$\Delta' = \{q \in (\Sigma \cup \{\#\})^{2,2} \mid T_{1,2}(q) \cup T_{2,1}(q) \subseteq \Delta\}$$

Soit $K = \{f \in \Sigma^{**} \mid T_{2,2}(\tilde{f}) \subseteq \Delta'\}$. Évidemment, $K$ est un langage local et nous montrons que $L = K$.

Considérons une figure $f$ de $K$. Alors $T_{2,2}(\tilde{f}) \subseteq \Delta'$ et $T_{1,2}(\tilde{f}) \subseteq T_{1,2}(T_{2,2}(\tilde{f})) \subseteq T_{1,2}(\Delta') \subseteq \Delta$. De manière similaire, $T_{2,1}(\tilde{f}) \subseteq \Delta$. Ainsi $f \in L$. D'un autre côté, soit $g$ une figure de $L$ et $a \in T_{2,2}(\tilde{g})$. Alors $T_{1,2}(a) \subseteq T_{1,2}(\tilde{g}) \subseteq \Delta$ et $T_{2,1}(a) \subseteq T_{2,1}(\tilde{g}) \subseteq \Delta$. Donc, $a \in \Delta'$ et $g \in K$. □

*Exemple. Le langage de figures donné dans l'exemple précédent est local avec :*

$$\Delta' = \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline \# & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 1 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline 0 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & 1 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 0 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & \# \\ \hline \# & \# \\ \hline \end{array} \right\}$$

Nous pouvons noter l'inclusion propre des langages de figures hv-locaux dans les langages locaux et des langages de figures locaux dans les langages de figures reconnaissables. Par exemple le langage de figures de hauteur deux avec uniquement des $a$ sur la première ligne et des $b$ sur la seconde est clairement local. L'image de ce langage par la projection qui associe $a$ et $b$ avec $c$ n'est pas local mais reconnaissable.

**Proposition 2.3** *Soit* $L \subseteq \Sigma^{**}$ *un langage de figures. Si L est local, il existe un langage de figures hv-local* $L'$ *sur un alphabet* $\Sigma'$ *et une projection* $\pi : \Sigma' \to \Sigma$ *tels que* $L = \pi(L')$.

*Preuve.* Dans un premier temps nous définissons un alphabet étendu de $\Sigma$. Nous notons cet alphabet $ext(\Sigma) = (\Sigma \cup \{\#\})^{3,3}$. Nous définissons maintenant une projection $\theta$ de $\Sigma^{**}$ dans $ext(\Sigma)^{**}$:

$$\begin{aligned} \theta : \ \Sigma^{**} &\longrightarrow ext(\Sigma)^{**} \\ f &\longmapsto f' \in ext(\Sigma)^{**} \text{ avec taille}(f') = \text{taille}(f) \text{ et :} \end{aligned}$$

$$f'(i,j) = \begin{array}{|c|c|c|} \multicolumn{3}{c}{\forall 1 \le i \le \text{col}(f) \quad \forall 1 \le j \le \text{lig}(f)} \\ \hline \tilde{f}(i,j) & \tilde{f}(i,j+1) & \tilde{f}(i,j+2) \\ \hline \tilde{f}(i+1,j) & \tilde{f}(i+1,j+1) & \tilde{f}(i+1,j+2) \\ \hline \tilde{f}(i+2,j) & \tilde{f}(i+2,j+1) & \tilde{f}(i+2,j+2) \\ \hline \end{array} \tag{2.1}$$

Notons que tout pavé $2 \times 2$ apparaissant dans $\tilde{f}$, où $f \in \Sigma^{**}$, apparaît dans les lettres de $\theta(f)$, et à l'inverse, tout pavé apparaissant dans les lettres de $\theta(f)$ est dans $\tilde{f}$, c'est-à-dire :

$$T_{2,2}(\tilde{f}) = \bigcup_{a \in T_{1,1}(\theta(f))} T_{2,2}(a) \tag{2.2}$$

Par exemple, si nous considérons la figure $f \in \{0,1\}^{**}$:

$$f = \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array} \quad \text{alors, nous avons :} \quad \theta(p) = \begin{array}{|c|c|} \hline a & b \\ \hline c & d \\ \hline \end{array}$$

avec

$$a = \begin{array}{|c|c|c|} \hline \# & \# & \# \\ \hline \# & 1 & 0 \\ \hline \# & 0 & 1 \\ \hline \end{array} \quad b = \begin{array}{|c|c|c|} \hline \# & \# & \# \\ \hline 1 & 0 & \# \\ \hline 0 & 1 & \# \\ \hline \end{array}$$

$$c = \begin{array}{|c|c|c|} \hline \# & 1 & 0 \\ \hline \# & 0 & 1 \\ \hline \# & \# & \# \\ \hline \end{array} \quad d = \begin{array}{|c|c|c|} \hline 1 & 0 & \# \\ \hline 0 & 1 & \# \\ \hline \# & \# & \# \\ \hline \end{array}$$

Nous définissons alors la projection $\pi$ de $ext(\Sigma)^{**}$ dans $\Sigma^{**}$ par $\pi(a) = a(2,2)$ pour $a \in ext(\Sigma)$. Par (2.1), il est évident que pour tout $f \in \Sigma^{**}$ nous avons $f = \pi(\theta(f))$.

Considérons le langage de figures $K = \theta(\Sigma^{**})$. Nous allons montrer l'assertion suivante.

**Assertion 2.4** *Le langage de figures $K$ est hv-local.*

Afin de montrer cette assertion, nous considérons l'ensemble des dominos $\Delta'$ défini par :

$$\Delta' = \{T_{2,1}(\tilde{f}) \mid f \in K\} \cup \{T_{1,2}(\tilde{f}) \mid f \in K\}$$

Il est facile de voir que :

$$\Delta' = \left\{ \begin{array}{|c|} \hline \# \\ \hline \# \\ \hline \end{array} \right\} \cup V_1 \cup V_2 \cup V_3 \cup \left\{ \begin{array}{|c|c|} \hline \# & \# \\ \hline \end{array} \right\} \cup H_1 \cup H_2 \cup H_3$$

avec

$$\begin{cases} V_1 = \left\{ \begin{array}{|c|}\hline \# \\ \hline a \\ \hline \end{array} \mid a \in \Sigma' \wedge a(1,1) = a(1,2) = a(1,3) = \# \right\} \\ V_2 = \left\{ \begin{array}{|c|}\hline a \\ \hline \# \\ \hline \end{array} \mid a \in \Sigma' \wedge a(3,1) = a(3,2) = a(3,3) = \# \right\} \\ V_3 = \left\{ \begin{array}{|c|}\hline a \\ \hline b \\ \hline \end{array} \mid a,b \in \Sigma' \wedge \forall 1 \leq i \leq 3 \quad (a(2,i) = b(1,i) \wedge a(3,i) = b(2,i)) \right\} \end{cases}$$

et

$$\begin{cases} H_1 = \left\{ \begin{array}{|c|c|}\hline \# & a \\ \hline \end{array} \mid a \in \Sigma' \wedge a(1,1) = a(2,1) = a(3,1) = \# \right\} \\ H_2 = \left\{ \begin{array}{|c|c|}\hline a & \# \\ \hline \end{array} \mid a \in \Sigma' \wedge a(1,3) = a(2,3) = a(3,3) = \# \right\} \\ H_3 = \left\{ \begin{array}{|c|c|}\hline a & b \\ \hline \end{array} \mid a,b \in \Sigma' \wedge \forall 1 \leq i \leq 3 \quad (a(i,2) = b(i,1) \wedge a(i,3) = b(i,2)) \right\} \end{cases}$$

Maintenant, soit $K'$ le langage de figures hv-local sur $ext(\Sigma)$ défini par $\Delta'$. Par définition de $\Delta'$ et $K'$, le langage $K$ est inclus dans $K'$. Afin de prouver l'inclusion inverse, considérons une figure $f' \in K'$ et la figure $f = \pi(f')$. Nous allons montrer que $f' = \theta(f)$, c'est-à-dire que $f'$ satisfait (2.1). Cette preuve se décompose en trois étapes.

Premièrement, comme $f = \pi(f')$, nous obtenons :

$$\forall i,j \quad f'(i,j)(2,2) = \tilde{f}(i+1,j+1) \qquad \textbf{(P1)}$$

Deuxièmement, considérons $f'(i,j)(1,2)$, nous devons distinguer deux cas :

1. si $i = 1$, le domino suivant appartient à $T_{2,1}(\tilde{f}') \subseteq \Delta'$:

$$\begin{array}{|c|}\hline \# \\ \hline f'(i,j) \\ \hline \end{array} \in V_1$$

   Alors nous avons $f'(i,j)(1,2) = \# = \tilde{f}(i,j+1)$.

2. si $i > 1$, le domino suivant appartient à $T_{2,1}(\tilde{f}') \subseteq \Delta'$:

$$\begin{array}{|c|}\hline f'(i-1,j) \\ \hline f'(i,j) \\ \hline \end{array} \in V_3$$

   Nous avons $f'(i,j)(1,2) = f'(i-1,j)(2,2)$ et par **(P1)**, nous avons $f'(i-1,j)(2,2) = \tilde{f}(i,j+1)$.

De la même manière, pour $f'(i,j)(3,2)$, $f'(i,j)(2,1)$ et $p'(i,j)(2,3)$, nous montrons que :

$$\begin{array}{llcl} \forall i,j & f'(i,j)(1,2) & = & \tilde{f}(i,j+1) \\ \forall i,j & f'(i,j)(3,2) & = & \tilde{f}(i+2,j+1) \\ \forall i,j & f'(i,j)(2,1) & = & \tilde{f}(i+1,j) \\ \forall i,j & f'(i,j)(2,3) & = & \tilde{f}(i+1,j+2) \end{array} \qquad \textbf{(P2)}$$

Enfin, considérons $f'(i,j)(1,1)$. Deux cas différents peuvent arriver :

1. Si $j = 1$, le domino suivant appartient à $T_{1,2}(\tilde{f}') \subseteq \Delta'$:

$$\boxed{\#}\boxed{f'(i,j)} \in H_1$$

et nous avons $f'(i,j)(1,1) = \# = \tilde{p}(i,j)$.

2. Si $j > 1$, le domino suivant appartient à $T_{1,2}(\tilde{f}') \subseteq \Delta'$:

$$\boxed{f'(i,j-1)}\boxed{f'(i,j)} \in H_3$$

et nous avons $f'(i,j)(1,1) = f'(i,j-1)(1,2)$ et par **(P2)**, nous avons $f'(i,j-1)(1,2) = \tilde{p}(i,j)$.

De la même manière, nous prouvons que :

$$\begin{array}{llcl} \forall i,j & f'(i,j)(1,1) & = & \tilde{f}(i,j) \\ \forall i,j & f'(i,j)(1,3) & = & \tilde{f}(i,j+2) \\ \forall i,j & f'(i,j)(3,1) & = & \tilde{f}(i+2,j) \\ \forall i,j & f'(i,j)(3,3) & = & \tilde{f}(i+2,j+2) \end{array} \qquad \textbf{(P3)}$$

Les équations **(P1)**, **(P2)** et **(P3)** signifient que $f' = \theta(f)$, c'est-à-dire que $f'$ appartient à $K$. Nous avons $K = K'$ qui est hv-local. Ceci clôt la preuve de l'assertion.

Maintenant, considérons un langage de figures local $L$ sur $\Sigma$ avec l'ensemble de pavés $\Delta$. Nous allons montrer que $L$ est l'image d'un langage de figures hv-local sur $ext(\Sigma)$ par la projection $\pi$. Selon (2.2), $\theta(L)$ est défini sur l'alphabet $\Sigma_\Delta$ :

$$\Sigma_\Delta = \{a \in ext(\Sigma) \mid T_{2,2}(a) \subseteq \Delta\}$$

et clairement, nous avons :

$$\theta(L) = \{f' \in \Sigma_\Delta^{**} \mid T_{2,1}(f') \cup T_{1,2}(f') \subseteq \Delta'\}$$

Ceci conclut la preuve de la Proposition 2.3 puisque $L = \pi(\theta(L))$. □

**Théorème 2.5** *Soit $L \subseteq \Sigma^{**}$ un langage de figures, $L$ est reconnaissable si et seulement si il existe un langage de figures hv-local $L'$ sur un alphabet $\Sigma'$ et une projection $\pi : \Sigma' \to \Sigma$ tels que $L = \pi(L')$.*

*Preuve.* Soit $L$ un langage de figures reconnaissable sur $\Sigma$. Par définition, nous savons qu'il existe un langage de figures local $M$ sur un alphabet $\Sigma'$ et une projection $\pi : \Sigma' \to \Sigma$ tels que $L = \pi(L')$. Selon la Proposition 2.3, il existe un langage de figures hv-local $L''$ sur un alphabet $\Sigma''$ et $\rho : \Sigma'' \to \Sigma'$ tels que $L' = \rho(L'')$. Nous avons donc $L = \pi(\rho(L''))$ où $L''$ est hv-local.

Maintenant, soit $L'$ un langage de figures hv-local sur $\Sigma'$ et $\pi : \Sigma' \to \Sigma$ une projection. D'après la Proposition 2.2 il en découle que $L'$ est local et donc que le langage de figures $L = \pi(L')$ est reconnaissable. □

Avant de conclure, nous traitons l'exemple d'un langage de figures reconnaissable donné par A. Restivo et al. [GR92, GRST96].

*Exemple. Ce langage est l'ensemble de tous les carrés sur l'alphabet $\Sigma = \{a\}$ déjà donné en exemple dans la Section 1.1 page 80. Nous avons :*

$$L = \left\{ \begin{array}{|c|}\hline a \\ \hline \end{array}, \begin{array}{|c|c|}\hline a & a \\ \hline a & a \\ \hline \end{array}, \begin{array}{|c|c|c|}\hline a & a & a \\ \hline a & a & a \\ \hline a & a & a \\ \hline \end{array}, \begin{array}{|c|c|c|c|}\hline a & a & a & a \\ \hline a & a & a & a \\ \hline a & a & a & a \\ \hline a & a & a & a \\ \hline \end{array} \cdots \right\}$$

*Nous donnons un langage de figures hv-local $K$ associé à $L$ :*

$$K = \left\{ \begin{array}{|c|}\hline 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 2 & 0 \\ \hline \end{array}, \begin{array}{|c|c|c|}\hline 0 & 1 & 1 \\ \hline 2 & 0 & 1 \\ \hline 2 & 2 & 0 \\ \hline \end{array}, \begin{array}{|c|c|c|c|}\hline 0 & 1 & 1 & 1 \\ \hline 2 & 0 & 1 & 1 \\ \hline 2 & 2 & 0 & 1 \\ \hline 2 & 2 & 2 & 0 \\ \hline \end{array} \cdots \right\}$$

*Nous voyons que $L = \pi(K)$ avec $\pi(0) = \pi(1) = \pi(2) = a$. De plus $K$ est hv-local pour l'ensemble de dominos $\Delta$ :*

$$\Delta = \left\{ \begin{array}{|c|}\hline \# \\ \hline \# \\ \hline \end{array}, \begin{array}{|c|}\hline \# \\ \hline 0 \\ \hline \end{array}, \begin{array}{|c|}\hline \# \\ \hline 1 \\ \hline \end{array}, \begin{array}{|c|}\hline 0 \\ \hline 2 \\ \hline \end{array}, \begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline \end{array}, \begin{array}{|c|}\hline 1 \\ \hline 1 \\ \hline \end{array}, \begin{array}{|c|}\hline 2 \\ \hline 2 \\ \hline \end{array}, \begin{array}{|c|}\hline 2 \\ \hline \# \\ \hline \end{array}, \begin{array}{|c|}\hline 0 \\ \hline \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 1 & \# \\ \hline \end{array}, \begin{array}{|c|c|}\hline \# & 2 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 2 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 2 & 2 \\ \hline \end{array}, \begin{array}{|c|c|}\hline 0 & \# \\ \hline \end{array} \right\}$$

## 2.3 Remarques

Dans les langages de figures hv-locaux, les contrôles horizontal et vertical sont séparés. Il est donc naturel de définir les langages de figures h-locaux et les langages de figures

v-locaux qui sont définis par les dominos horizontaux autorisés dans le premier cas et dominos verticaux dans le dernier cas. Il est évident que nous ne pouvons pas obtenir tous les reconnaissables par projection d'un h-local ou d'un v-local. Ceci est dû, entre autres, au fait que nous ne pouvons borner le nombre de lignes (respectivement colonnes) dans un langage de figures h-local (resp. v-local).

Il est facile d'associer à chaque langage de figures h-local (respectivement v-local) un langage de mots local $K$ tel que $f \in L$ si et seulement si chaque ligne (resp. colonne) de $p$ (pris en temps que mot) est dans $K$. De plus, pour chaque langage de figures hv-local $L$ avec un ensemble de dominos autorisés $\Delta$, nous avons $\Delta = \Gamma_h \cup \Gamma_v$ où $\Gamma_h$ est un ensemble de dominos horizontaux et $\Gamma_v$ est un ensemble de dominos verticaux. Si nous considérons le langage de figures h-local associé avec $\Gamma_h$ et le langage de figures $K_v$ associé avec $\Gamma_v$, il est clair que $L = K_h \cap K_v$. Donc, un langage de figures reconnaissable est complètement défini par deux langages de mots locaux (un pour $K_h$ et un pour $K_v$) et une projection.

**Définition 2.6** *Soient $L$ et $K$ deux langages de mots sur $\Sigma$, on note $F = L \oplus K$ le langage de figures de $\Sigma^{**}$ défini par:*

$$F = \{f \in \Sigma^{**} \mid \text{lignes}(f) \subseteq L \wedge \text{colonnes}(f) \subseteq K\}$$

Dans cette définition, lignes($f$) désigne l'ensemble des lignes de $f$ prises comme des mots de $\Sigma^*$ et colonnes($f$) l'ensemble des colonnes prises comme des mots — lues de haut en bas. Cette opération de « croisement » est étendue naturellement aux familles de langages. Pour $\mathcal{A}$ et $\mathcal{B}$ deux familles de langages de mots, $\mathcal{A} \oplus \mathcal{B}$ désigne la famille de langages de figures $\{L \oplus K \mid L \in \mathcal{A}, K \in \mathcal{B}\}$. On a par définition des hv-locaux hv-Loc($\Sigma^{**}$) = Loc($\Sigma^*$) $\oplus$ Loc($\Sigma^*$). On a donc montré dans ce chapitre que:

**Proposition 2.7** *Soit $L \subseteq \Sigma^{**}$, $L$ est un langage de figures reconnaissable si et seulement si il existe un alphabet $\Sigma'$, deux langages de mots locaux $A, B \in \text{Loc}(\Sigma'^*)$ et une projection $\pi : \Sigma' \to \Sigma$ tels que $L = \pi(A \oplus B)$.*

Notons que pour des raisons de concision on peut représenter un langage de figures reconnaissable par deux langages de mots reconnaissables et une projection. En effet, on montre facilement en utilisant le produit cartésien ($\otimes$) que:

**Proposition 2.8** *Soient $A, B \in \text{Rec}(\Sigma^*)$, il existe un alphabet $\Sigma'$, deux langages $A', B' \in \text{Loc}(\Sigma'^*)$ et une projection $\pi : \Sigma' \to \Sigma$ tels que $A \oplus B = \pi(A' \oplus B')$.*

**Corollaire 2.9** *Soit $L \subseteq \Sigma^{**}$, $L$ est un langage de figures reconnaissable si et seulement si il existe un alphabet $\Sigma'$, deux langages de mots reconnaissables $A, B \in \text{Rec}(\Sigma'^*)$ et une projection $\pi : \Sigma' \to \Sigma$ tels que $L = \pi(A \oplus B)$.*

Par exemple l'ensemble de tous les carrés sur l'alphabet $\{a\}$ présenté ci-dessus peut être représenté par $(\pi, R_h, R_v)$ avec $\pi(0) = \pi(1) = a$ et $R_h = R_v = 0^*10^*$ [1]. Il est intéressant

1. Ceci est également une manière facile de définir le langage $L_2$ de la Section 1.3 pour le problème des $k$-reines.

d'étudier les familles de langages de figures qui peuvent être définis en utilisant d'autres classes de la hiérarchie de Chomsky comme ceci est fait dans la section 11 de [GR96]. Nous donnons dans le Chapitre 5 une contribution à cette étude.

# Références

[GR92] D. Giammarresi and A. Restivo. Recognizable picture languages. *International Journal Pattern Recognition and Artificial Intelligence*, pages 31–46, 1992. Special Issue on *Parallel Image Processing*. M. Nivat, A. Saoudi and P.S.P. Wangs (Eds.).

[GR96] D. Giammarresi and A. Restivo. Two-dimensional languages. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3, pages 215–267. Springer-Verlag, Berlin, 1996.

[GRST96] D. Giammarresi, A. Restivo, S. Seibert, and W. Thomas. Monadic second-order logic over rectangular pictures and recognizability by tiling systems. *Information and Computation*, 125(1):32–45, 1996.

[LS97] M. Latteux and D. Simplot. Recognizable picture languages and domino tiling. *Theoretical Computer Science*, 178(1/2):275–283, 1997.

# Chapitre 3

# Pavage et langages de figures reconnaissables

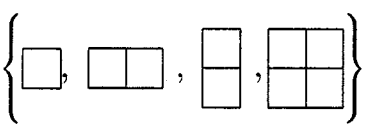
Il est difficile en parlant de figures de ne pas rencontrer les pavages. Dans ce chapitre nous définissons des opérations de pavage qui tendent à s'approcher de la notion intuitive de pavage que l'on peut trouver par exemple dans [GS86]. L'opération de pavage qui correspond à cette notion intuitive est noté $**$ et on montre que la classe des langages de figures reconnaissables est close par cette opération.

En quelque sorte, cette opération est l'extension la plus naturelle de l'opération étoile dans les mots. Comme les langages de mots reconnaissables peuvent être caractérisés par l'image homomorphique de l'intersection de l'étoile de deux langages de mots finis [CFS82, Tur82, KL83, LL83], on cherche une caractérisation du même type utilisant le pavage par deux ensembles finis de figures.

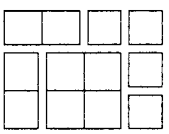
## 3.1 Opérations de pavage

Pour définir un pavage par un ensemble de figures, on peut utiliser les opérations de concaténations et d'étoiles que nous avons déjà vues. Néanmoins, comme l'illustre l'exemple suivant, ces définitions ne correspondent pas à l'idée intuitive que l'on se fait du pavage.

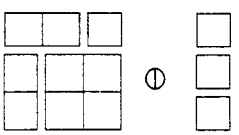
*Exemple. On considère un langage de figures $L$ contenant des figures ayant la forme suivante :*

$L =$ 

*Ainsi, ni $(L^{\ominus *})^{\oplus *}$ ni $(L^{\oplus *})^{\ominus *}$ ne peuvent contenir la figure $f$ ayant la forme :*

$f =$ 

*En effet, pour que $f$ appartienne à $(L^{\ominus *})^{\oplus *}$, il faudrait qu'il existe dans $L^{\ominus *}$ des figures $f_1, f_2, \ldots, f_i$ telles que $f = f_1 \oplus f_2 \oplus \ldots \oplus f_i$. Or ici la seule décomposition verticale de $f$ est :*

$f =$ 

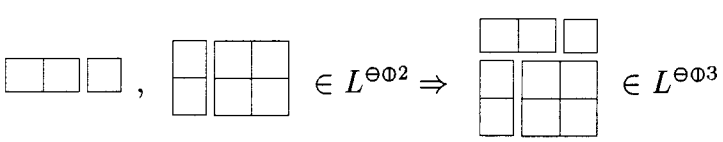
*et la première figure n'appartient pas à $L^{\ominus *}$.*

C'est pourquoi on définit une nouvelle opération étoile notée $^{\ominus \oplus *}$.

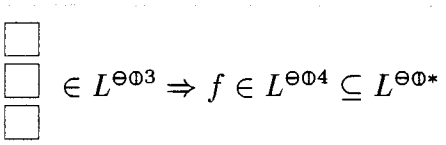
**Définition 3.1** *Soit $L \subseteq \Sigma^{**}$ un langage de figures, on définit :*

1. $L^{\ominus \oplus 1} = L$,
2. $L^{\ominus \oplus i+1} = L^{\ominus \oplus i} \cup L^{\ominus \oplus i} \ominus L^{\ominus \oplus i} \cup L^{\ominus \oplus i} \oplus L^{\ominus \oplus i}$,
3. $L^{\ominus \oplus *} = \bigcup_{i \geq 1} L^{\ominus \oplus i}$.

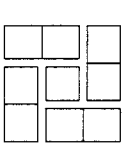
*Exemple. Avec le même ensemble de figures $L$ et la figure $f$ de l'exemple précédent, on a bien $f \in L^{\ominus \oplus *}$ car on a :*



*et*



*Néanmoins, la figure $f'$ suivante n'appartient pas à $L^{\ominus \oplus *}$ puisqu'elle ne peut être obtenue par concaténation de figures de $L$ :*

$f' =$ 

Définir un pavage cohérent qui contient des figures comme celle donnée dans l'exemple consiste à trouver une partition de la figure où chaque composante est une figure de l'ensemble de départ.

**Définition 3.2** *Soit $f$ une figure de $\Sigma^{**}$, un sous-domaine de $f$ est un ensemble du type $\{x,\dots,x'\}\times\{y,\dots,y'\}$ tel que $1\le x\le x'\le \mathrm{lig}(f)$ et $1\le y\le y'\le \mathrm{col}(f)$. On note $D(f)$ l'ensemble des sous-domaines de $f$.*

*Soit $d=\{x,\dots,x'\}\times\{y,\dots,y'\}\in D(f)$, la sous-figure associée à $d$, notée $\mathrm{sfig}(f,d)$, est la figure de taille $(x'-x+1,y'-y+1)$ telle que :*

$$\forall 1\le i\le x'-x+1\ \forall 1\le j\le y'-y+1\quad \mathrm{sfig}(f,d)(i,j)=f(x+i-1,y+j+1)$$

*Exemple. Pour $\Sigma=\{a,b,c\}$, la figure :*

$$f=\begin{array}{|c|c|c|c|c|c|}\hline b&c&a&b&b&a\\\hline a&a&b&c&a&c\\\hline a&b&a&c&a&b\\\hline c&a&c&a&c&a\\\hline\end{array}$$

*et le sous-domaine $d=\{2,3,4\}\times\{3,4\}$, on a :*

$$\mathrm{sfig}(f,d)=\begin{array}{|c|c|}\hline b&c\\\hline a&c\\\hline c&a\\\hline\end{array}$$

**Définition 3.3** *Soit $L$ un langage de figures sur $\Sigma$, l'ensemble des pavages par $L$ ou $L$-pavages, noté $L^{**}$, est le langage qui contient les figures $f\in\Sigma^{**}$ vérifiant :*

$$\exists d_1,\dots,d_k\in D(f)\ \textit{une partition de}\ \{1\dots\mathrm{lig}(f)\}\times\{1\dots\mathrm{col}(f)\}$$
$$\textit{telle que}$$
$$\forall 1\le i\le k\quad \mathrm{sfig}(f,d_i)\in L$$

Cette définition est cohérente avec la notion usuelle de pavage utilisée par exemple pour recouvrir des polyominos ou le plan [Gol66, Gol70, GS86, BN90, BN91]. O. Matz définit également d'autres « pavages » obtenus par concaténation dans [Mat97].

Notons que si les figures d'un langage $L$ sont toutes de même taille, on a $(L^{\Theta*})^{\Phi*}=(L^{\Phi*})^{\Theta*}=L^{\Theta\Phi*}=L^{**}$.

## 3.2 Clotûre des langages de figures reconnaissables par pavage

D. Giammarresi et A. Restivo ont montré dans [GR96] que les figures obtenues par pavage d'un ensemble fini de polyominos constituaient un langage de figures reconnaissable. Le langage obtenu par pavage d'un langage de figures fini est donc reconnaissable. Nous étendons ce résultat aux langages de figures reconnaissables.

**Proposition 3.4** *Soit $L \in \mathrm{Rec}(\Sigma^{**})$ un langage de figures reconnaissable, $L^{**}$ est reconnaissable.*

*Preuve.* Le langage de figures reconnaissable $L$ est la projection d'un langage hv-local $K \in \mathrm{Loc}(\Sigma'^{**})$ par $\pi : \Sigma' \to \Sigma$. Notons $\Delta$ l'ensemble des dominos autorisés dans $K$.

Nous construisons d'abord un langage reconnaissable $P$ qui contient toutes les partitions des figures. On utilise l'alphabet $X$ défini par :

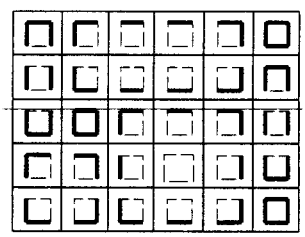
$$\begin{aligned} X' &= \{\square,\square,\square,\square\} \\ X &= 2^{X'} \end{aligned}$$

Cet alphabet $X$ sera dénoté sans ambiguïté par :

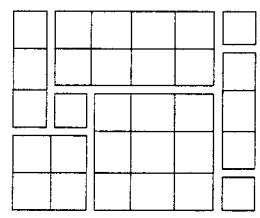
$$X = \left\{ \begin{array}{c} \square,\square,\square,\square,\square, \\ \square,\square,\square,\square,\square,\square, \\ \square,\square,\square,\square,\square \end{array} \right\}$$

Par exemple, le symbole $\square$ représente l'ensemble $\{\square,\square\}$.

Ce que nous appelons « partition » est une figure sur $X$ du type :



qui représente la partition d'une figure de taille $(5,6)$ :



Ce langage $P$ est un langage hv-local car on peut lui associer l'ensemble de dominos autorisés :

$$\Theta = \left\{ \begin{array}{c} \boxed{a}\boxed{b} \in (X \cup \{\#\})^{1,2} \\ \text{tels que} \\ (a = \# \vee \square \in a) \iff (b = \# \vee \square \in b) \\ (\square \notin a \wedge \square \in a) \iff (\square \notin b \wedge \square \in b) \\ (\square \notin a \wedge \square \in a) \iff (\square \notin b \wedge \square \in b) \end{array} \right\}$$

$$\cup \left\{ \begin{array}{c} \begin{array}{|c|}\hline a \\ \hline b \\ \hline \end{array} \in (X \cup \{\#\})^{2,1} \\ \text{tels que} \\ (a = \# \vee \square \in a) \iff (b = \# \vee \square \in b) \\ (\square \notin a \wedge \square \in a) \iff (\square \notin b \wedge \square \in b) \\ (\square \notin a \wedge \square \in a) \iff (\square \notin b \wedge \square \in b) \end{array} \right\}$$

Notons $P'$ le langage de figures hv-local obtenu par produit cartésien de $P$ et $\Sigma'^{**}$ : $P' = P \otimes \Sigma'^{**}$. L'ensemble des dominos autorisés dans $P'$ est noté $\Theta'$ et est dérivé de $\Theta$. Soit $\varphi : (X \times \Sigma') \cup \{\#\} \to \Sigma' \cup \{\#\}$ qui associe à $(x, a) \in X \times \Sigma'$ la lettre $a$ et $\#$ à $\#$. On a $\Theta' = \varphi^{-1}(\Theta)$.

Il faut maintenant vérifier que les figures de chaque composante de la partition appartiennent à $K$. Ceci se fait en deux étapes.

La première consiste à vérifier que les lettres se trouvant sur le bord d'une composante sont autorisées à être sur le bord d'une figure de $K$. On utilise un sous-alphabet de $X \times \Sigma'$ que l'on note $Y$ :

$$Y = \left\{ (x, a) \in X \times \Sigma' \mid \begin{array}{lcl} \square \in x & \Rightarrow & \begin{array}{|c|}\hline \# \\ \hline a \\ \hline \end{array} \in \Delta \\ \square \in x & \Rightarrow & \boxed{a}\boxed{\#} \in \Delta \\ \square \in x & \Rightarrow & \begin{array}{|c|}\hline a \\ \hline \# \\ \hline \end{array} \in \Delta \\ \square \in x & \Rightarrow & \boxed{\#}\boxed{a} \in \Delta \end{array} \right\}$$

Soit $P''$ le langage hv-local défini comme l'intersection de $P'$ et $Y^{**}$. L'ensemble des dominos autorisés dans $P''$ est noté $\Theta'' = \Theta' \cap (\{\#\} \cup Y)^{**}$.

La deuxième étape consiste à vérifier que les figures à l'intérieur des composantes de la partition ne contiennent que des figures de $K$. On construit un langage de figures hv-local $P'''$ sur $Y$ qui vérifie que les dominos supposés être dans une même composante sont bien dans $\Delta$. On pose $\Theta'''$ :

$$\Theta''' = \left\{ \boxed{\#}\boxed{(x,a)}, \boxed{(x,a)}\boxed{\#} \in \Theta'' \right\}$$
$$\cup \left\{ \begin{array}{|c|}\hline \# \\ \hline (x,a) \\ \hline \end{array}, \begin{array}{|c|}\hline (x,a) \\ \hline \# \\ \hline \end{array} \in \Theta'' \right\}$$
$$\cup \left\{ \boxed{(x,a)}\boxed{(y,b)} \in \Theta'' \mid \square \notin x \Rightarrow \boxed{a}\boxed{b} \in \Delta \right\}$$
$$\cup \left\{ \begin{array}{|c|}\hline (x,a) \\ \hline (y,b) \\ \hline \end{array} \in \Theta'' \mid \square \notin x \Rightarrow \begin{array}{|c|}\hline a \\ \hline b \\ \hline \end{array} \in \Delta \right\}$$
$$\cup \left\{ \boxed{\#}\boxed{\#}, \begin{array}{|c|}\hline \# \\ \hline \# \\ \hline \end{array} \right\}$$

Ce langage hv-local contient toutes les partitions (par construction de $P'$) où les lettres sur les bords des composantes peuvent se situer sur un bord de même type d'une figure de $K$ (par construction de $P''$) et où les dominos à l'intérieur des composantes sont dans $\Delta$ (par construction de $P'''$).

On définit $\sigma : Y \times \Sigma' \to \Sigma$ la projection telle que $\sigma((x,a)) = \pi(a)$ et on a $L^{**} = \sigma(P''')$. Le langage $L^{**}$ est donc bien reconnaissable. □

## 3.3 Caractérisation des reconnaissables par pavage

Il est bien connu [CFS82, Tur82, KL83, LL83] que pour tout langage de mots reconnaissable $R \in \mathrm{Rec}(\Sigma^*)$ il existe deux ensembles de mots finis $A$ et $B$ sur $\Sigma'$ et un homomorphisme $\varphi : \Sigma'^* \to \Sigma^*$ tels que :

$$R^* = \varphi(A^* \cap B^*)$$

On en déduit que pour tout $R \in \mathrm{Rec}(\Sigma^*)$, il existe un homomorphisme $\varphi : \Sigma'^* \to \Sigma^*$ et deux ensembles finis $A, B \subset (\Sigma' \cup \{\#\})^*$ tels que :

$$R = \{\varphi(\omega) \mid \#\omega\# \in A^* \cap B^*\}$$

Nous essayons ici d'obtenir un résultat similaire dans les figures en utilisant les pavages.

**Proposition 3.5** *Soit $L \subseteq \Sigma^{**}$, $L$ est un langage de figures reconnaissable si et seulement si il existe quatre ensembles (finis) de figures de $(\Sigma' \cup \{\#\})^{2,2} \cup \{\boxed{\#}\}$ notés $A_1$, $A_2$, $A_3$ et $A_4$ et une projection $\pi : \Sigma' \to \Sigma$ tels que :*

$$L = \{\pi(f) \mid \tilde{f} \in A_1^{**} \cap A_2^{**} \cap A_3^{**} \cap A_4^{**}\}$$

*Preuve.* La classe des reconnaissables contenant les langages finis et étant close par pavage, intersection et pavage, il est clair que la proposition est vraie dans le sens $\Leftarrow$. Maintenant, pour tout reconnaissable $L$, on construit les ensembles $A_1$, $A_2$, $A_3$ et $A_4$ ainsi que la projection $\pi$ qui vérifient la propriété.

Le langage $L$ est l'image par une projection $\varphi$ d'un langage de figures local $K$ sur $X$ auquel est associé l'ensemble de pavés $(2,2)$ autorisés $\Delta$.

L'idée de la preuve est identique à celle dans les mots. On considère le langage local associé au reconnaissable et on construit les ensembles $A_1$, $A_2$, $A_3$ et $A_4$ de manière à ce que, pour une figure entourée de #, le pavage par $A_1$ commence sur la cellule $(1,1)$, le pavage par $A_2$ sur $(1,2)$, le pavage par $A_3$ sur $(2,1)$ et le pavage par $A_4$ sur la cellule $(2,2)$. Pour ceci, on va « colorier » les cases avec quatre couleurs distinctes.

On pose :

$$\Sigma' = \{a_1, a_2, a_3, a_4 \mid a \in X\}$$

Les indices correspondent aux « couleurs » que l'on impose aux cases. On désire obtenir un coloriage du type :

| | | | | | | |
|---|---|---|---|---|---|---|
| # | # | # | # | # | # | # |
| # | $-_1$ | $-_2$ | $-_1$ | $-_2$ | $-_1$ | # |
| # | $-_3$ | $-_4$ | $-_3$ | $-_4$ | $-_3$ | # |
| # | $-_1$ | $-_2$ | $-_1$ | $-_2$ | $-_1$ | # |
| # | $-_3$ | $-_4$ | $-_3$ | $-_4$ | $-_3$ | # |
| # | # | # | # | # | # | # |

On note $\psi$ la projection de $(\Sigma' \cup \{\#\}) \to X$ qui associe # à # et $a$ à $a_i \in \Sigma'$. On désigne par $X_i$ l'ensemble des lettres de $\Sigma'$ qui sont indicées par $i$ ($i \in \{1,2,3,4\}$).

Seuls les pavages de $A_1$ doivent commencer en $(1,1)$ :

$$A_1 = \psi^{-1}(\Delta) \cap \left\{ \begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline \end{array} \;\middle|\; \begin{array}{ll} a \in \{\#\} \cup X_4 & b \in \{\#\} \cup X_3 \\ c \in \{\#\} \cup X_2 & d \in \{\#\} \cup X_1 \end{array} \right\} \cup \left\{ \boxed{\#} \right\}$$

Le pavage par $A_2$ doit commencer en $(1,2)$ :

$$A_2 = \left( \psi^{-1}(\Delta) \cap \left\{ \begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline \end{array} \;\middle|\; \begin{array}{ll} a \in \{\#\} \cup X_3 & b \in \{\#\} \cup X_4 \\ c \in \{\#\} \cup X_1 & d \in \{\#\} \cup X_2 \end{array} \right\} \cup \left\{ \boxed{\#} \right\} \right) \setminus \left\{ \begin{array}{|c|c|}\hline \# & a \\ \hline \# & b \\ \hline \end{array} \;\middle|\; \begin{array}{l} a \in \{\#\} \cup X_4 \\ b \in \{\#\} \cup X_2 \end{array} \right\}$$

Celui de $A_3$ doit commencer en $(2,1)$ :

$$A_3 = \left( \psi^{-1}(\Delta) \cap \left\{ \begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline \end{array} \;\middle|\; \begin{array}{ll} a \in \{\#\} \cup X_2 & b \in \{\#\} \cup X_1 \\ c \in \{\#\} \cup X_4 & d \in \{\#\} \cup X_3 \end{array} \right\} \cup \left\{ \boxed{\#} \right\} \right)$$
$$\setminus \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline a & b \\ \hline \end{array} \;\middle|\; a \in \{\#\} \cup X_4 \quad b \in \{\#\} \cup X_3 \right\}$$

Enfin, le pavage de $A_4$ doit commencer en $(2,2)$ :

$$A_4 = \left( \psi^{-1}(\Delta) \cap \left\{ \begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline \end{array} \;\middle|\; \begin{array}{ll} a \in \{\#\} \cup X_1 & b \in \{\#\} \cup X_2 \\ c \in \{\#\} \cup X_3 & d \in \{\#\} \cup X_4 \end{array} \right\} \cup \left\{ \boxed{\#} \right\} \right)$$
$$\setminus \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline a & b \\ \hline \end{array} \;\middle|\; a \in \{\#\} \cup X_3 \quad b \in \{\#\} \cup X_4 \right\}$$
$$\setminus \left\{ \begin{array}{|c|c|}\hline \# & a \\ \hline \# & b \\ \hline \end{array} \;\middle|\; \begin{array}{l} a \in \{\#\} \cup X_2 \\ b \in \{\#\} \cup X_4 \end{array} \right\}$$

Pour une figure $f'$ de $\Sigma'^{m,n}$ dont $\tilde{f}'$ est dans $A_1^{**}$, on a obligatoirement :

$$\forall f' \in \Sigma'^{m,n}$$

$$\tilde{f}' \in A_1^{**}$$
$$\Downarrow$$
$$\forall 2 \leq i \leq m+1 \; \forall 2 \leq j \leq n+1$$

$$\left\{ \begin{array}{lll} \tilde{f}'(i,j) \in X_1 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i-1,j-1) & \tilde{f}'(i-1,j) \\ \hline \tilde{f}'(i,j-1) & \tilde{f}'(i,j) \\ \hline \end{array} \in A_1 \\ \tilde{f}'(i,j) \in X_2 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i-1,j) & \tilde{f}'(i-1,j+1) \\ \hline \tilde{f}'(i,j) & \tilde{f}'(i,j+1) \\ \hline \end{array} \in A_1 \\ \tilde{f}'(i,j) \in X_3 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i,j-1) & \tilde{f}'(i,j) \\ \hline \tilde{f}'(i+1,j-1) & \tilde{f}'(i+1,j) \\ \hline \end{array} \in A_1 \\ \tilde{f}'(i,j) \in X_4 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i,j) & \tilde{f}'(i,j+1) \\ \hline \tilde{f}'(i+1,j) & \tilde{f}'(i+1,j+1) \\ \hline \end{array} \in A_1 \end{array} \right. \qquad \textbf{(P1)}$$

On a des propriétés identiques pour $A_2$, $A_3$ et $A_4$ :

$$\forall f' \in \Sigma'^{m,n}$$

$$\tilde{f}' \in A_2^{**}$$
$$\Downarrow$$
$$\forall 2 \leq i \leq m+1 \; \forall 2 \leq j \leq n+1$$

$$\begin{cases} \tilde{f}'(i,j) \in X_2 \Rightarrow \begin{array}{|c|c|}\hline \tilde{f}'(i-1,j-1) & \tilde{f}'(i-1,j) \\ \hline \tilde{f}'(i,j-1) & \tilde{f}'(i,j) \\ \hline \end{array} \in A_2 \\ \tilde{f}'(i,j) \in X_1 \Rightarrow \begin{array}{|c|c|}\hline \tilde{f}'(i-1,j) & \tilde{f}'(i-1,j+1) \\ \hline \tilde{f}'(i,j) & \tilde{f}'(i,j+1) \\ \hline \end{array} \in A_2 \\ \tilde{f}'(i,j) \in X_4 \Rightarrow \begin{array}{|c|c|}\hline \tilde{f}'(i,j-1) & \tilde{f}'(i,j) \\ \hline \tilde{f}'(i+1,j-1) & \tilde{f}'(i+1,j) \\ \hline \end{array} \in A_2 \\ \tilde{f}'(i,j) \in X_3 \Rightarrow \begin{array}{|c|c|}\hline \tilde{f}'(i,j) & \tilde{f}'(i,j+1) \\ \hline \tilde{f}'(i+1,j) & \tilde{f}'(i+1,j+1) \\ \hline \end{array} \in A_2 \end{cases} \quad \textbf{(P2)}$$

$$\forall f' \in \Sigma'^{m,n}$$

$$\tilde{f}' \in A_3^{**}$$
$$\Downarrow$$
$$\forall 2 \leq i \leq m+1 \; \forall 2 \leq j \leq n+1$$

$$\begin{cases} \tilde{f}'(i,j) \in X_3 \Rightarrow \begin{array}{|c|c|}\hline \tilde{f}'(i-1,j-1) & \tilde{f}'(i-1,j) \\ \hline \tilde{f}'(i,j-1) & \tilde{f}'(i,j) \\ \hline \end{array} \in A_3 \\ \tilde{f}'(i,j) \in X_4 \Rightarrow \begin{array}{|c|c|}\hline \tilde{f}'(i-1,j) & \tilde{f}'(i-1,j+1) \\ \hline \tilde{f}'(i,j) & \tilde{f}'(i,j+1) \\ \hline \end{array} \in A_3 \\ \tilde{f}'(i,j) \in X_1 \Rightarrow \begin{array}{|c|c|}\hline \tilde{f}'(i,j-1) & \tilde{f}'(i,j) \\ \hline \tilde{f}'(i+1,j-1) & \tilde{f}'(i+1,j) \\ \hline \end{array} \in A_3 \\ \tilde{f}'(i,j) \in X_2 \Rightarrow \begin{array}{|c|c|}\hline \tilde{f}'(i,j) & \tilde{f}'(i,j+1) \\ \hline \tilde{f}'(i+1,j) & \tilde{f}'(i+1,j+1) \\ \hline \end{array} \in A_3 \end{cases} \quad \textbf{(P3)}$$

$$\forall f' \in \Sigma'^{m,n}$$

$$\tilde{f}' \in A_4^{**}$$

$$\Downarrow$$

$$\forall 2 \leq i \leq m+1 \; \forall 2 \leq j \leq n+1$$

$$\left\{ \begin{array}{lll} \tilde{f}'(i,j) \in X_4 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i-1,j-1) & \tilde{f}'(i-1,j) \\ \hline \tilde{f}'(i,j-1) & \tilde{f}'(i,j) \\ \hline \end{array} \in A_4 \\ \tilde{f}'(i,j) \in X_3 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i-1,j) & \tilde{f}'(i-1,j+1) \\ \hline \tilde{f}'(i,j) & \tilde{f}'(i,j+1) \\ \hline \end{array} \in A_4 \\ \tilde{f}'(i,j) \in X_2 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i,j-1) & \tilde{f}'(i,j) \\ \hline \tilde{f}'(i+1,j-1) & \tilde{f}'(i+1,j) \\ \hline \end{array} \in A_4 \\ \tilde{f}'(i,j) \in X_1 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i,j) & \tilde{f}'(i,j+1) \\ \hline \tilde{f}'(i+1,j) & \tilde{f}'(i+1,j+1) \\ \hline \end{array} \in A_4 \end{array} \right. \quad \textbf{(P4)}$$

Soit $a \in \Sigma'$ la première lettre supérieure gauche de $f'$ : $a = \tilde{f}'(2,2)$. Cette lettre peut appartenir à $X_1$, $X_2$, $X_3$ ou $X_4$. Néammoins en utilisant les définitions des ensembles $A_i$ et les propriétés (**Pi**) ($i \in \{1,2,3,4\}$), on montre que le seul cas possible est $a \in X_1$.

En effet, si $a \in X_i$ avec $i \in \{1,2,3,4\}$, on a d'après (**Pi**) :

$$\begin{array}{|c|c|}\hline \# & \# \\ \hline \# & a \\ \hline \end{array} \in A_i$$

Ceci est impossible, d'après les définitions des $A_i$, si $i = \{2,3,4\}$.

Avec le même raisonnement, on montre que si $\tilde{f}'$ appartient aux $A_i^{**}$ avec $1 \leq i \leq 4$, on a :

$$\begin{array}{rr} \forall 1 \leq 2i+1 \leq m \; \forall 1 \leq 2j+1 \leq n & f'(2i+1,2j+1) \in X_1 \\ \forall 1 \leq 2i+1 \leq m \; \forall 1 \leq 2j \leq n & f'(2i+1,2j) \in X_2 \\ \forall 1 \leq 2i \leq m \; \forall 1 \leq 2j+1 \leq n & f'(2i,2j+1) \in X_3 \\ \forall 1 \leq 2i \leq m \; \forall 1 \leq 2j \leq n & f'(2i,2j) \in X_4 \end{array}$$

En utilisant les propriétés (**Pi**) énoncées précédemment, on montre que :

$$
\begin{array}{rl}
\forall 1 \leq 2i+1 \leq m\ \forall 1 \leq 2j+1 \leq n & \begin{array}{|c|c|}\hline \tilde{f'}(2i+1,2j+1) & \tilde{f'}(2i+1,2j+2) \\ \hline \tilde{f'}(2i+2,2j+1) & \tilde{f'}(2i+2,2j+2) \\ \hline \end{array} \in A_1 \\
\forall 1 \leq 2i \leq m\ \forall 1 \leq 2j+1 \leq n & \begin{array}{|c|c|}\hline \tilde{f'}(2i+1,2j) & \tilde{f'}(2i+2,2j) \\ \hline \tilde{f'}(2i+2,2j) & \tilde{f'}(2i+2,2j+1) \\ \hline \end{array} \in A_2 \\
\forall 1 \leq 2i \leq m\ \forall 1 \leq 2j+1 \leq n & \begin{array}{|c|c|}\hline \tilde{f'}(2i,2j+1) & \tilde{f'}(2i,2j+2) \\ \hline \tilde{f'}(2i+1,2j+1) & \tilde{f'}(2i+1,2j+2) \\ \hline \end{array} \in A_3 \\
\forall 1 \leq 2i \leq m\ \forall 1 \leq 2j \leq n & \begin{array}{|c|c|}\hline \tilde{f'}(2i,2j) & \tilde{f'}(2i,2j+1) \\ \hline \tilde{f'}(2i+1,2j) & \tilde{f'}(2i+1,2j+1) \\ \hline \end{array} \in A_4
\end{array}
\qquad (\mathbf{P})
$$

*Exemple. Pour la figure $f'$ dont la figure entourée appartient aux pavages respectifs de $A_1$, $A_2$, $A_3$ et $A_4$ :*

$$
f' = \begin{array}{|c|c|c|c|c|}\hline a_1 & b_2 & b_1 & a_2 & c_1 \\ \hline a_3 & c_4 & a_3 & b_4 & c_3 \\ \hline b_1 & b_2 & c_1 & c_2 & c_1 \\ \hline c_3 & a_4 & b_3 & c_4 & b_3 \\ \hline \end{array}
$$

*Ces différents pavages sont du type :*

$$
\begin{array}{|c|c|}\hline \# & \# \\ \hline \# & a_1 \\ \hline \end{array}\ \begin{array}{|c|c|}\hline \# & \# \\ \hline b_2 & b_1 \\ \hline \end{array}\ \begin{array}{|c|c|}\hline \# & \# \\ \hline a_2 & c_1 \\ \hline \end{array}\ \begin{array}{|c|}\hline \# \\ \hline \# \\ \hline \end{array}
$$

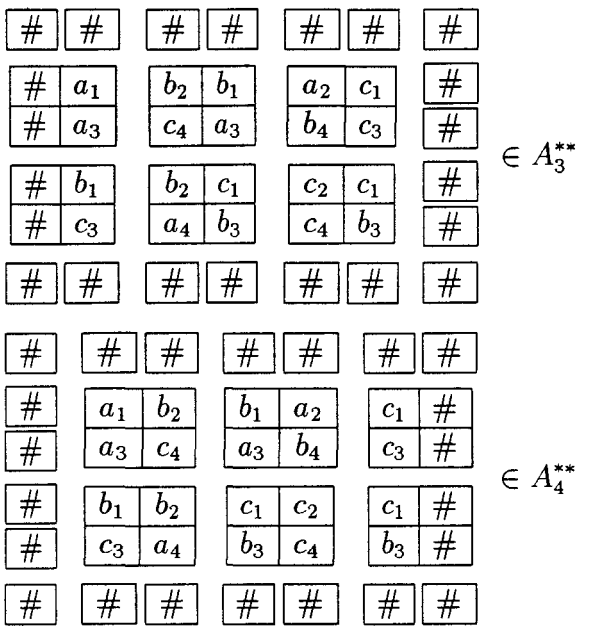
$$
\begin{array}{|c|c|}\hline \# & a_3 \\ \hline \# & b_1 \\ \hline \end{array}\ \begin{array}{|c|c|}\hline c_4 & a_3 \\ \hline b_2 & c_1 \\ \hline \end{array}\ \begin{array}{|c|c|}\hline b_4 & c_3 \\ \hline c_2 & c_1 \\ \hline \end{array}\ \begin{array}{|c|}\hline \# \\ \hline \# \\ \hline \end{array} \in A_1^{**}
$$

$$
\begin{array}{|c|c|}\hline \# & c_3 \\ \hline \# & \# \\ \hline \end{array}\ \begin{array}{|c|c|}\hline a_4 & b_3 \\ \hline \# & \# \\ \hline \end{array}\ \begin{array}{|c|c|}\hline c_4 & b_3 \\ \hline \# & \# \\ \hline \end{array}\ \begin{array}{|c|}\hline \# \\ \hline \# \\ \hline \end{array}
$$

$$
\begin{array}{|c|}\hline \# \\ \hline \# \\ \hline \end{array}\ \begin{array}{|c|c|}\hline \# & \# \\ \hline a_1 & b_2 \\ \hline \end{array}\ \begin{array}{|c|c|}\hline \# & \# \\ \hline b_1 & a_2 \\ \hline \end{array}\ \begin{array}{|c|c|}\hline \# & \# \\ \hline c_1 & \# \\ \hline \end{array}
$$

$$
\begin{array}{|c|}\hline \# \\ \hline \# \\ \hline \end{array}\ \begin{array}{|c|c|}\hline a_3 & c_4 \\ \hline b_1 & b_2 \\ \hline \end{array}\ \begin{array}{|c|c|}\hline a_3 & b_4 \\ \hline c_1 & c_2 \\ \hline \end{array}\ \begin{array}{|c|c|}\hline c_3 & \# \\ \hline c_1 & \# \\ \hline \end{array} \in A_2^{**}
$$

$$
\begin{array}{|c|}\hline \# \\ \hline \# \\ \hline \end{array}\ \begin{array}{|c|c|}\hline c_3 & a_4 \\ \hline \# & \# \\ \hline \end{array}\ \begin{array}{|c|c|}\hline b_3 & c_4 \\ \hline \# & \# \\ \hline \end{array}\ \begin{array}{|c|c|}\hline b_3 & \# \\ \hline \# & \# \\ \hline \end{array}
$$




En d'autres termes, la propriété (**P**) signifie que toute sous-figure dans $T_{2,2}(\tilde{f'})$ est dans un $A_i$. La figure $f = \psi(f') \in \Sigma^{**}$ est donc une figure de $K$ puisque toute sous-figure dans $T_{2,2}(\tilde{f})$ est une figure de $\psi(A_i) \subseteq \Delta$.

Dans l'autre sens, pour toute figure $f \in K$, il est facile de construire une figure $f' \in \psi^{-1}$ telle que $\tilde{f'}$ appartienne à $A_1^{**}$, $A_2^{**}$, $A_3^{**}$ et $A_4^{**}$. Pour ceci, il suffit de colorier $f$ de manière adéquate.

En posant $\pi = \varphi \circ \psi$, on obtient bien le résultat escompté. □

En utilisant les langages locaux, on peut donc trouver quatre ensembles finis de carrés $(2,2)$ pour caractériser les reconnaissables. On peut raffiner ce résultats, passer à deux ensembles de carrés $(2,2)$, en utilisant les hv-locaux.

**Proposition 3.6** *Soit $L \subseteq \Sigma^{**}$, $L$ est un langage de figures reconnaissable si et seulement si il existe deux ensembles (finis) de figures de $(\Sigma' \cup \{\#\})^{2,2} \cup \{\boxed{\#}\}$ $A_1$ et $A_2$ et une projection $\pi : \Sigma' \to \Sigma$ tels que :*

$$L = \{\pi(f) \mid \tilde{f} \in A_1^{**} \cap A_2^{**}\}$$

*Preuve.* De la même manière que précédemment, on a juste besoin de construire ces ensembles $A_1$ et $A_2$ pour tout reconnaissable $L$. Le langage $L$ est l'image par une projection $\varphi$ d'un langage de figures hv-local $K$ sur $X$ auquel est associé l'ensemble de dominos autorisés $\Delta$. On note $\Delta'$ l'ensemble des pavés $(2,2)$ autorisés dans $K$ (tout hv-local est local — Proposition 2.2) :

$$\Delta' = \{p \in (X \cup \{\#\})^{2,2} \mid T_{2,1}(p) \subseteq \Delta \wedge T_{1,2}(p) \subseteq \Delta\}$$

Nous allons procéder de la même manière que dans la preuve précédente, nous colorions les cases des figures sur $X$ en utilisant l'alphabet $\Sigma' = \{a_1, a_2, a_3, a_4 \mid a \in X\}$ afin d'obtenir un coloriage du type :

| | | | | | |
|---|---|---|---|---|---|
| # | # | # | # | # | # |
| # | $-_1$ | $-_2$ | $-_1$ | $-_2$ | # |
| # | $-_3$ | $-_4$ | $-_3$ | $-_4$ | # |
| # | $-_1$ | $-_2$ | $-_1$ | $-_2$ | # |
| # | $-_3$ | $-_4$ | $-_3$ | $-_4$ | # |
| # | # | # | # | # | # |

On note $\psi$ la projection de $(\Sigma' \cup \{\#\}) \to X$ qui associe # à # et $a$ à $a_i \in \Sigma'$. On désigne par $X_i$ l'ensemble des lettres de $\Sigma'$ qui sont indicées par $i$ ($i \in \{1,2,3,4\}$).

Le pavage par $A_1$ commencera en $(1,1)$ :

$$A_1 = \psi^{-1}(\Delta') \cap \left\{ \begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline \end{array} \mid \begin{array}{ll} a \in \{\#\} \cup X_4 & b \in \{\#\} \cup X_3 \\ c \in \{\#\} \cup X_2 & d \in \{\#\} \cup X_1 \end{array} \right\} \cup \left\{ \boxed{\#} \right\}$$

Le pavage par $A_2$ commencera en $(2,2)$ :

$$\begin{aligned} A_2 = & \left( \psi^{-1}(\Delta') \cap \left\{ \begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline \end{array} \mid \begin{array}{ll} a \in \{\#\} \cup X_1 & b \in \{\#\} \cup X_2 \\ c \in \{\#\} \cup X_3 & d \in \{\#\} \cup X_4 \end{array} \right\} \cup \left\{ \boxed{\#} \right\} \right) \\ & \setminus \left\{ \begin{array}{|c|c|}\hline \# & a \\ \hline \# & b \\ \hline \end{array} \mid \begin{array}{l} a \in \{\#\} \cup X_4 \\ b \in \{\#\} \cup X_2 \end{array} \right\} \\ & \setminus \left\{ \begin{array}{|c|c|}\hline \# & \# \\ \hline a & b \\ \hline \end{array} \mid a \in \{\#\} \cup X_4 \quad b \in \{\#\} \cup X_3 \right\} \end{aligned}$$

Pour une figure $f'$ de $\Sigma'^{m,n}$ dont $\tilde{f}'$ est dans $A_1^{**}$, on a obligatoirement :

$$\forall f' \in \Sigma'^{m,n}$$

$$\tilde{f}' \in A_1^{**}$$

$$\Downarrow$$

$$\forall 2 \leq i \leq m+1 \; \forall 2 \leq j \leq n+1$$

$$\left\{ \begin{array}{lcl} \tilde{f}'(i,j) \in X_1 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i-1,j-1) & \tilde{f}'(i-1,j) \\ \hline \tilde{f}'(i,j-1) & \tilde{f}'(i,j) \\ \hline \end{array} \in A_1 \\ \tilde{f}'(i,j) \in X_2 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i-1,j) & \tilde{f}'(i-1,j+1) \\ \hline \tilde{f}'(i,j) & \tilde{f}'(i,j+1) \\ \hline \end{array} \in A_1 \\ \tilde{f}'(i,j) \in X_3 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i,j-1) & \tilde{f}'(i,j) \\ \hline \tilde{f}'(i+1,j-1) & \tilde{f}'(i+1,j) \\ \hline \end{array} \in A_1 \\ \tilde{f}'(i,j) \in X_4 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i,j) & \tilde{f}'(i,j+1) \\ \hline \tilde{f}'(i+1,j) & \tilde{f}'(i+1,j+1) \\ \hline \end{array} \in A_1 \end{array} \right. \quad \textbf{(P1)}$$

De même; si $f'$ appartient à $A_2^{**}$ :

$$\forall f' \in \Sigma'^{m,n}$$

$$\tilde{f}' \in A_2^{**}$$

$$\Downarrow$$

$$\forall 2 \leq i \leq m+1 \; \forall 2 \leq j \leq n+1$$

$$\left\{ \begin{array}{lcl} \tilde{f}'(i,j) \in X_4 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i-1,j-1) & \tilde{f}'(i-1,j) \\ \hline \tilde{f}'(i,j-1) & \tilde{f}'(i,j) \\ \hline \end{array} \in A_2 \\ \tilde{f}'(i,j) \in X_3 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i-1,j) & \tilde{f}'(i-1,j+1) \\ \hline \tilde{f}'(i,j) & \tilde{f}'(i,j+1) \\ \hline \end{array} \in A_2 \\ \tilde{f}'(i,j) \in X_2 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i,j-1) & \tilde{f}'(i,j) \\ \hline \tilde{f}'(i+1,j-1) & \tilde{f}'(i+1,j) \\ \hline \end{array} \in A_2 \\ \tilde{f}'(i,j) \in X_1 & \Rightarrow & \begin{array}{|c|c|}\hline \tilde{f}'(i,j) & \tilde{f}'(i,j+1) \\ \hline \tilde{f}'(i+1,j) & \tilde{f}'(i+1,j+1) \\ \hline \end{array} \in A_2 \end{array} \right. \quad \textbf{(P2)}$$

En utilisant ces deux propriétés, on montre facilement que si $f'$ est dans le pavage de $A_1$ et dans le pavage de $A_2$ le coloriage des lettres est bien celui que l'on souhaitait. De plus, on a :

$$\forall 1 \leq 2i+1 \leq m \; \forall 1 \leq 2j+1 \leq n \quad \begin{array}{|c|c|}\hline \tilde{f}'(2i+1,2j+1) & \tilde{f}'(2i+1,2j+2) \\ \hline \tilde{f}'(2i+2,2j+1) & \tilde{f}'(2i+2,2j+2) \\ \hline \end{array} \in A_1$$

$$\forall 1 \leq 2i \leq m \; \forall 1 \leq 2j \leq n \quad \begin{array}{|c|c|}\hline \tilde{f}'(2i,2j) & \tilde{f}'(2i,2j+1) \\ \hline \tilde{f}'(2i+1,2j) & \tilde{f}'(2i+1,2j+1) \\ \hline \end{array} \in A_2$$

Ainsi tous les dominos apparaissant dans $\tilde{f}'$ sont des dominos de $\psi^{-1}(\Delta)$ — c'est-à-dire que $f'$ est une de $K$.

Enfin, en appliquant le coloriage idoine à une figure $f$ de $K$, il est clair que cette figure colorié sera à la fois dans $A_1^{**}$ et dans $A_2^{**}$.

Il suffit donc de considérer la projection $\pi = \varphi \circ \psi$ pour avoir la propriété souhaitée. □

En utilisant la définition des langages de figures reconnaissables par le croisement de langages de mots reconnaissables horizontalement et verticalement, on obtient directement la proposition suivante.

**Proposition 3.7** *Soit $L \subseteq \Sigma^{**}$, $L$ est un langage de figures reconnaissable si et seulement si il existe quatre ensembles (finis) de figures de $(\Sigma' \cup \{\#\})^{1,2} \cup (\Sigma' \cup \{\#\})^{2,1} \cup \{\boxed{\#}\}$ notés $A_1$, $A_2$, $A_3$ et $A_4$ et une projection $\pi : \Sigma' \to \Sigma$ tels que :*

$$L = \{\pi(f) \mid \tilde{f} \in A_1^{**} \cap A_2^{**} \cap A_3^{**} \cap A_4^{**}\}$$

On peut améliorer ce résultat en montrant que l'on peut se limiter à trois ensembles de dominos ou encore de deux ensembles finis de figures de dimension un — en l'occurence un ensemble comportant des figures de taille $(3,1)$ et l'autre des dominos verticaux. Pour ceci, on utilise des techniques similaires à celles appliquées dans la preuve de la Proposition 2.3 lorsque l'on code des renseignements supplémentaires dans les lettres afin de faire passer l'information sur les contrôles qui ne sont pas effectués directement. Ces résultats feront l'objet d'une publication en préparation [Sim97].

# Références

[BN90] D. Beauquier and M. Nivat. Tiling the plane with one tile. In *Annual ACM Symposium on Computational Geometry*, 1990.

[BN91] D. Beauquier and M. Nivat. Tiling pictures of the plane with two bars, a horizontal and a vertical one. Technical Report 91.67, L.I.T.P., Univ. Paris 7, France, 1991.

[CFS82] K. Culik II, F. E. Fich, and A. Salomaa. A homomorphic characterization of regular languages. *Discrete Appl. Math.*, 4:149–152, 1982.

[Gol66] S. W. Golomb. Tiling with polyominoes. *Journal of Combinatorial Theory*, 1:280–296, 1966.

[Gol70] S. W. Golomb. Tiling with sets of polyominoes. *Journal of Combinatorial Theory*, 9:60–71, 1970.

[GR96] D. Giammarresi and A. Restivo. Two-dimensional finite state recognizability. *Fundamenta Informaticae*, 25(3/4):399–422, 1996. Special issue: Formal Language Theory.

[GS86] B. Grünbaum and G. C. Shepard. *Tilings and Patterns*. W. H. Freeman and Company, New York, 1986.

[KL83] J. Karhumäki and M. Linna. A note on morphic characterization of languages. *Discrete Appl. Math.*, 5:243–246, 1983.

[LL83] M. Latteux and J. Leguy. On composition of morphisms and inverse morphisms. *Lecture Notes in Computer Science*, 154:420–432, 1983.

[Mat97] O. Matz. Regular expressions and contextfree grammars for picture languages. In *Proc. STACS'97*, volume 1200 of *Lecture Notes in Computer Science*, pages 283–294, Lübeck, Germany, 1997. Springer-Verlag, Berlin.

[Sim97] D. Simplot. A characterization of recognizable picture languages by tilings of finite sets. Technical report, L.I.F.L., Univ. Lille 1, France, 1997. en préparation.

[Tur82] P. Turakainen. A homomorphic characterization of principal semi-AFLs without using intersection with regular sets. Technical report, University of Oulu, Finland, 1982.

# Chapitre 4

# Langages de mots contextuels et langages de figures reconnaissables

## 4.1 Introduction

Le lien entre les langages de mots algébriques et le feuillage des langages d'arbres reconnaissables est un résultat de base dans la théorie des langages formels [MW67]. Dans ce chapitre nous montrons la connexion entre les langages de mots contextuels et les frontières, définies en tant que premières lignes, des langages de figures reconnaissables. Plus précisément nous prouvons que la famille des frontières des langages de figures reconnaissables est exactement la classe des langages contextuels. Ce résultat renforce l'intérêt de la famille des langages de figures reconnaissables.

Les résultats de ce chapitre ont été soumis à publication et seront publiés dans [LS]. Dans une synthèse des recherches sur les ordres partiels W. Thomas a également repris ces résultats en établissant notamment des comparaisons entre les cas particuliers d'ordres partiels que sont les arbres d'une part et les figures d'autre part [Tho97].

## 4.2 Frontières des langages de figures reconnaissables

Dans cette section, nous montrons notre résultat principal qui établit la correspondance exacte entre langages contextuels et frontières des langages de figures reconnaissables.

**Définition 4.1** *Soit $f$ une figure de $\Sigma^{n,m}$. La frontière de $f$ est la ligne supérieure de $f$ qui est le mot $f(1,1)\dots f(1,m)$ et est notée* $\mathrm{fr}(f)$.

La notion de frontière est étendue aux langages de figures et aux classes de langages de figures.

**Définition 4.2** *Soit $L$ un langage de figures (respectivement une classe de langages de figures). Nous notons* $\mathrm{fr}(L)$ *le langage (resp. la classe de langages) de $\Sigma^*$ définie par:*

$$\mathrm{fr}(L) = \{\mathrm{fr}(f) \mid f \in L\}$$

Dans un premier temps, nous montrons que la frontière d'un langage de figures reconnaissable est contextuelle.

**Proposition 4.3** *Pour tout langage de figures reconnaissable $A$, le langage* $\mathrm{fr}(A)$ *est un langage contextuel.*

*Preuve.* Nous montrons que pour un langage hv-local $K$ sur un alphabet $\Sigma$, le langage $\mathrm{fr}(K)$ est contextuel. En effet, nous contruisons une grammaire contextuelle pour $\#\mathrm{fr}(K)\#\#$. Clairement ceci suffit à montrer que $\mathrm{fr}(K)$ est contextuel. L'idée est de construire une grammaire contextuelle qui vérifie ligne par ligne, en partant du bas, une figure de $K$. Soit $\Delta$ l'ensemble de dominos associé à $K$. Nous pouvons supposer que les dominos$\# \ominus \#$ et $\# \oplus \#$ appartiennent à $\Delta$ sinon $K$ est vide.

Nous définissons la grammaire $G =< X, V, P, S >$ par :

- l'ensemble des symboles terminaux $X = \left\{ a \in \Sigma \mid \begin{array}{|c|}\hline \# \\ \hline a \\ \hline \end{array} \in \Delta \right\} \cup \{\#\}$.

- l'ensemble des variables $V = (\Sigma \setminus X) \cup \{S, A, L, R\}$ où $S$ est l'axiome.

- l'ensemble des règles de production $P$ qui est divisé en quatre parties :

$$P = P_1 \cup P_2 \cup P_3 \cup P_4$$

1. $P_1 =$« générer la ligne inférieure » :

$$S \to \#aA \quad \text{pour} \quad \begin{array}{|c|c|}\hline \# & a \\ \hline \end{array}\,, \begin{array}{|c|}\hline a \\ \hline \# \\ \hline \end{array} \in \Delta \quad a \in \Sigma$$

$$aA \to abA \quad \text{pour} \quad \begin{array}{|c|c|}\hline a & b \\ \hline \end{array}\,, \begin{array}{|c|}\hline b \\ \hline \# \\ \hline \end{array} \in \Delta \quad a, b \in \Sigma$$

$$aA \to aL\# \quad \text{pour} \quad \begin{array}{|c|c|}\hline a & \# \\ \hline \end{array} \in \Delta \quad a \in \Sigma$$

2. $P_2 =$« retour à la première colonne » :

$$aL \to La \quad \text{pour } a \in \Sigma$$
$$\#L \to \#R$$

3. $P_3$ =« vérifier le pavage horizontal et vertical »:

$$bRa \rightarrow ba'R \quad \text{pour} \quad \boxed{b}\boxed{a'}, \begin{array}{|c|}\hline a' \\ \hline a \\ \hline \end{array} \in \Delta \qquad \begin{array}{l} a, a' \in \Sigma \\ b \in \Sigma \cup \{\#\} \end{array}$$

$$aR\# \rightarrow aL\# \quad \text{pour} \quad \boxed{a}\boxed{\#} \in \Sigma \quad a \in \Sigma$$

4. $P_4$ =« fin de la dérivation »:

$$L\# \rightarrow \#\#$$

Soit $K_1$, $K_2$ et $K_3$ les langages de figures hv-locaux sur $\Sigma$ définis par les ensembles de dominos autorisés $\Delta_1$, $\Delta_2$ et $\Delta_3$ respectivement:

$$\Delta_1 = \Delta \cup \left\{ \begin{array}{|c|}\hline \# \\ \hline a \\ \hline \end{array} \mid a \in \Sigma \right\}$$

$$\Delta_2 = \Delta_1 \cup \left\{ \begin{array}{|c|}\hline a \\ \hline \# \\ \hline \end{array} \mid a \in \Sigma \right\}$$

$$\Delta_3 = \Delta_2 \cup \left\{ \boxed{a}\boxed{\#} \mid a \in \Sigma \right\}$$

On montre facilement que l'on a les propriétés suivantes:

$$\forall p \in K_1 \quad (p \in K \iff \mathrm{fr}(p) \in X^*) \tag{4.1}$$

$$\forall p \in K_1, u \in \Sigma^{\oplus *} \quad (u \ominus \mathrm{fr}(p) \in K_2 \Rightarrow u \ominus p \in K_1) \tag{4.2}$$

$$\forall u, v \in \Sigma^{\oplus *} \quad (\Sigma^{**} \ominus u \ominus v \ominus \Sigma^{**} \cap K_1 \neq \emptyset \Rightarrow u \ominus v \in K_2) \tag{4.3}$$

Afin de montrer la proposition, nous prouvons l'assertion suivante.

**Assertion 4.4** $S \xrightarrow[G]{*} \#uL\# \iff u \in \mathrm{fr}(K_1)$.

Nous pouvons déduire cette assertion des propriétés suivantes:

$$\forall u \in \Sigma^* \quad (S \xrightarrow[P1]{*} \#uL\# \implies u \in K_1) \tag{4.4}$$

$$\forall u \in \mathrm{fr}(K_1) \quad (\#Ru\# \xrightarrow[P3]{*} \#vL\# \iff v \ominus u \in K_2) \tag{4.5}$$

En raison des relations entre les règles de $P_1$ et les ensembles de dominos $\Delta$ et $\Delta_1$, la propriété (4.4) est clairement vraie. La propriété (4.5) peut être dérivée de la suivante:

$$\forall u \in \mathrm{fr}(K_1) \quad \left( \#Ru\# \overset{+}{\underset{P3}{\rightarrow}} \#v'Rv''\# \iff \exists u' \in \Sigma^+ \begin{array}{c} u = u'v'' \\ v' \ominus u' \in K_3 \end{array} \right)$$

Cette dernière propriété est facilement prouvée par induction sur la longueur de $u'$ en utilisant la définition des règles de $P_3$.

Une dérivation de $G$ qui génère un mot $\#uL\#$ est une séquence ayant la forme suivante :

$$S \overset{+}{\underset{P1}{\rightarrow}} \#u_1L\# \overset{+}{\underset{P2}{\rightarrow}} \#Ru_1\# \overset{+}{\underset{P3}{\rightarrow}} \#u_2L\# \overset{+}{\underset{P2}{\rightarrow}} \#Ru_2\# \overset{+}{\underset{P3}{\rightarrow}} \ldots \overset{+}{\underset{P3}{\rightarrow}} \#u_nL\#$$

Pour montrer l'implication de gauche-à-droite de l'assertion, nous raisonnons par induction sur la longueur de la dérivation en utilisant la propriété (4.4) pour commencer la récurrence et les propriétés (4.2) et (4.5) pour les étapes suivantes. Pour l'implication inverse, nous prenons une figure $f$ de $K_1$ et nous montrons que $\#\mathrm{fr}(p)L\#$ peut être dérivé de l'axiome. De la même manière, nous raisonnons par induction sur la hauteur de $f$. Nous utilisons la propriété (4.4) pour commencer la récurrence et nous utilisons les propriétés (4.3) et (4.5) pour les étapes suivantes. Ceci achève la preuve de l'Assertion 4.4.

Il est maintenant facile de conclure. La seule manière d'effacer la variable $L$ (la variable $R$ peut seulement être remplacée par $L$) est d'appliquer la règle de $P_4$ :

$$S \overset{*}{\rightarrow} \#uL\# \underset{P4}{\rightarrow} \#u\#\#$$

Selon l'assertion, nous savons que $u \in K_1$. Si $u$ contient seulement des symboles terminaux, par (4.1) nous savons que $u$ appartient à $\mathrm{fr}(K)$. De l'autre côté, si $u \in \mathrm{fr}(K)$, puisque $K$ est inclus dans $K_1$, nous pouvons dériver $\#uL\#$ de $S$. Ainsi la grammaire $G$ génère le langage $\#\mathrm{fr}(K)\#\#$. □

La classe des langages-frontières des langages de figures reconnaissables $\mathrm{fr}(\mathrm{Rec}(\Sigma^{**}))$ est donc incluse dans $\mathrm{CS}(\Sigma^*)$. Nous montrons maintenant l'inclusion inverse.

Il est bien connu que tout langage contextuel est reconnu par un automate linéaire borné. Un automate linéaire borné $M$ est un t-uple $M =< Q, \Sigma, V, \delta, q_0, F >$ où $Q$ est l'ensemble des états, $\Sigma$ l'alphabet d'entrée, $V$ l'aphabet de calcul (qui contient $\Sigma$), $\delta$ la fonction de transition $\delta : Q \times V \rightarrow 2^{Q \times V \times \{-1,0,1\}}$, $q_0 \in Q$ l'état initial et $F$ l'ensemble des états finaux.

Un pas de calcul est donné par la fonction de transition :

$$\begin{array}{rcll} u.b\,q\,a.v & \vdash & u\,q'\,b.a'.v & \text{pour } u, v \in V^*\ a, b \in V\ (q', a', -1) \in \delta(q, a) \\ u\,q\,a.v & \vdash & u\,q'\,a'.v & \text{pour } u, v \in V^*\ a \in V\ (q', a', 0) \in \delta(q, a) \\ u\,q\,a.v & \vdash & u.a'\,q'\,v & \text{pour } u, v \in V^*\ a \in V\ (q', a', 1) \in \delta(q, a) \end{array}$$

Un mot $w \in \Sigma^*$ est accepté par $M$, s'il existe un calcul

$$q_0\,w \vdash \ldots \vdash w'\,q$$

pour un $w' \in V^*$ et $q \in F$.

**Proposition 4.5** *Soit $L$ un langage contextuel sur $\Sigma$. Il existe un langage de figures reconnaissable $K \in \mathrm{Rec}(\Sigma^{**})$ tel que $L = \mathrm{fr}(K)$.*

*Preuve.* Soit $A =< Q, \Sigma, V, \delta, q_0, F >$ un automate linéaire borné qui reconnaît $L$. Nous construisons un langage de figures (2,3)-localement testable $K$ qui simule le comportement de l'automate. Ce langage est sur l'alphabet $\Sigma'$ qui est défini par :

$$\Sigma' = V \cup (Q \times V)$$

Le langage $K$ est défini par l'ensemble $\Delta$ de figures $2 \times 3$ autorisées :

$$K = \{p \in \Sigma'^{**} \mid T_{2,3}(\tilde{p}) \subseteq \Delta\}$$

Nous définissons maintenant l'ensemble $\Delta = \Delta_1 \cup \Delta_2 \cup \Delta_3 \cup \Delta_4$ :

1. positionner l'état initial sur la première cellule de la première ligne :

$$\Delta_1 = \left\{ \begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline \# & (q_0,a) & c \\ \hline \end{array}, \begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline (q_0,a) & b & c \\ \hline \end{array}, \begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline a & b & c \\ \hline \end{array} \; ; \; a,b \in \Sigma \wedge c \in \Sigma \cup \{\#\} \right\}$$

2. simuler les transitions de l'automate :

$$\Delta_2 = \left\{ \begin{array}{|c|c|c|}\hline a & (q,b) & c \\ \hline a & (q',b') & c \\ \hline \end{array} \; ; \; (q',b',0) \in \delta(q,b) \wedge a,c \in V \cup \{\#\} \right\}$$

$$\cup \left\{ \begin{array}{|c|c|c|}\hline a & (q,b) & c \\ \hline (q',a) & b' & c \\ \hline \end{array} \; ; \; \begin{array}{l}(q',b',-1) \in \delta(q,b) \\ \wedge c \in V \cup \{\#\} \wedge a \in V\end{array} \right\}$$

$$\cup \left\{ \begin{array}{|c|c|c|}\hline a & (q,b) & c \\ \hline a & b' & (q',c) \\ \hline \end{array} \; ; \; \begin{array}{l}(q',b',1) \in \delta(q,b) \\ \wedge a \in V \cup \{\#\} \wedge c \in V\end{array} \right\}$$

$$\Delta_3 = \left\{ \begin{array}{|c|c|c|}\hline a_1 & a_2 & a_3 \\ \hline a'_1 & a'_2 & a'_3 \\ \hline \end{array} \; ; \; \begin{array}{l} a_1,a'_1,a'_2,a_3,a'_3 \in \Sigma' \cup \{\#\} \wedge a_2 \in V \\ \wedge (a'_2 \in Q \times V \Rightarrow (a_1,a_3) \notin (V \cup \{\#\})^2) \\ \wedge (a_i \in V \Rightarrow a'_i \in \{a_i\} \cup Q \times \{a_i\}) \end{array} \right\}$$

Notons que dans un pavé de $\Delta_3$ aucun état ne peut apparaître en position $1,2$ et que si un état apparaît en position 2,2 un état doit apparaître en $1,1$ ou en $1,3$. L'autre rôle de $\Delta_3$ est de s'assurer que les lettres sans état ne changent pas d'une ligne à l'autre.

3. vérifier que la dernière ligne mène l'automate dans un état final :

$$\Delta_4 = \left\{ \begin{array}{|c|c|c|}\hline a & b & c \\ \hline \# & \# & \# \\ \hline \end{array} \; ; \quad \begin{array}{c} a \in V \cup \{\#\} \wedge b \in V \\ \wedge c \in \Sigma' \end{array} \right\}$$

$$\cup \left\{ \begin{array}{|c|c|c|}\hline a & (q,b) & \# \\ \hline \# & \# & \# \\ \hline \end{array} \; ; \quad \begin{array}{c} a \in V \cup \{\#\} \wedge b \in V \\ \wedge (q', c', 1) \in \delta(q, b) \\ \wedge q' \in F \wedge c' \in \Sigma \end{array} \right\}$$

Soit $f$ une figure de taille $(n, m)$ appartenant à $K$. Par la définition de $\Delta_1$, il est facile de voir que $\mathrm{fr}(f)$ appartient à $(\{q_0\} \times \Sigma.)\Sigma^*$. Par induction sur la hauteur de $f$, nous montrons les propriétés suivantes :

$$\forall 1 \leq i \leq n \quad f(i,1) \ldots f(i,m) \in V^*.(Q \times V).V^* \tag{4.6}$$

$$\forall 1 \leq i < n \quad \left. \begin{array}{c} f(i,1) \ldots f(i,m) = u.(q,a).v \\ f(i+1,1) \ldots f(i+1,m) = u'.(q',a').v' \end{array} \right\} \Rightarrow u\,q\,a.v \underset{A}{\vdash} u'\,q'\,a'.v' \tag{4.7}$$

Nous utilisons le fait que la seule manière pour une lettre de $Q \times V$ d'être en position $(1,2)$ dans un pavé $2 \times 3$ est que ce pavé appartienne à $\Delta_2$ ou $\Delta_4$. Donc, si sur une ligne $i$ nous avons un état, sur la ligne suivante nous avons au moins un état. Si un état est en position $(2,2)$ dans un pavé $2 \times 3$, ce pavé est dans $\Delta_2$ ou $\Delta_3$ et donc nous avons un état en position $(1,1)$, $(1,2)$ ou $(1,3)$. Puisque cet état est recouvert en position $(1,2)$ d'un pavé de $\Delta_2$, nous avons clairement la propriété (4.6).

Afin d'obtenir la propriété (4.7), nous utilisons le fait que les pavés de $\Delta_2$ vérifient que la transition est correcte.

Si nous notons $f(1,1) \ldots f(1,m) = (q_0, a).u$ et $f(n,1) \ldots f(n,m) = u'.(q,a').v'$, où $q \in Q$, $a \in \Sigma$, $u \in \Sigma^*$, $a' \in V$ et $u', v' \in V^*$, nous avons :

$$q_0\,a.u \overset{n-1}{\underset{A}{\vdash}} u'\,q\,a'.v'$$

Selon les pavés autorisés pour la dernière ligne apparaissant dans $\Delta_4$, nous déduisons que $v' = \varepsilon$ et que $q\,a' \underset{A}{\vdash} a''\,q''$ où $q'' \in F$. Nous avons alors :

$$q_0\,a.u \overset{n}{\underset{A}{\vdash}} u.a''\,q'' \quad q'' \in F$$

Nous considérons une projection $\pi$ de $\Sigma'$ dans $\Sigma$ définie par :

$$\forall a \in \Sigma' \quad \pi(a) = \left\{ \begin{array}{ll} a & \text{si } a \in \Sigma \\ b & \text{avec } b \in \Sigma \text{ si } a \in V \setminus \Sigma \\ c & \text{si } a = (q, c) \end{array} \right.$$

Nous notons $K' = \pi(K)$, le mot $\pi(\mathrm{fr}(f)) = \mathrm{fr}(\pi(f)) \in \mathcal{L}(A)$. Ceci montre que $\mathrm{fr}(K') \subseteq \mathcal{L}(A)$.

Pour l'inclusion inverse, il est facile de voir que pour un calcul

$$q_0\, u \overset{*}{\underset{A}{\vdash}} u'\, q'\, a \underset{A}{\vdash} u''\, q''$$

avec $q'' \in F$, nous pouvons construire une figure de $K$ qui simule ce calcul.

Nous avons donc bien $\mathrm{fr}(K') = \mathcal{L}(A)$. □

Des Propositions 4.3 et 4.5, nous déduisons le résultat escompté. Il établit qu'il y a équivalence entre CS dans les monoïdes libres fr(Rec) et fr(hv-Loc) dans les figures. Plus précisément, nous avons :

**Théorème 4.6** *Étant donné un langage $L \subseteq \Sigma^*$, les propriétés suivantes sont équivalentes :*

1. $L \in \mathrm{CS}(\Sigma^*)$ ;
2. $L = \mathrm{fr}(K)$ *pour un* $K \in \mathrm{Rec}(\Sigma^{**})$ ;
3. $L = \mathrm{fr}(K)$ *pour un* $K \in \text{hv-Loc}(\Sigma'^{**})$ *avec* $\Sigma \subseteq \Sigma'$.

Le dernier point est facile à déduire du précédent. Si nous avons un langage de figures reconnaissable $K$ tel que $\mathrm{fr}(K) = L$, nous savons que $K$ est l'image d'un langage de figures hv-local $K' \subseteq \Sigma'^{**}$ par une projection $\pi$. Nous pouvons supposer que $\Sigma'$ et $\Sigma$ sont disjoints. Soit $\Delta$ l'ensemble de dominos associé à $K'$. Nous définissons un ensemble de dominos $\Delta'$ :

$$\begin{aligned}\Delta' \;=\; \Delta \setminus \left\{ \begin{array}{|c|}\hline \# \\ \hline a \\ \hline \end{array} \mid a \in \Sigma' \right\} \cup \left\{ \begin{array}{|c|}\hline \# \\ \hline a \\ \hline \end{array} \mid a \in \Sigma \right\} \\ \cup \left\{ \begin{array}{|c|}\hline a \\ \hline b \\ \hline \end{array} \mid a \in \Sigma, b \in \Sigma', \pi(b) = a \right\} \\ \cup \left\{ \begin{array}{|c|c|}\hline \# & a \\ \hline \end{array} \;,\; \begin{array}{|c|c|}\hline a & b \\ \hline \end{array} \;\; \begin{array}{|c|c|}\hline a & \# \\ \hline \end{array} \mid a, b \in \Sigma \right\}\end{aligned}$$

Une figure du langage de figures hv-local de $(\Sigma \cup \Sigma')^{**}$ défini par $\Delta'$ est la concaténation verticale d'un mot $u$ de $\Sigma^+$ et ne figure $f$ de $K'$ tels que $u = \pi(\mathrm{fr}(f))$. Ainsi le dernier point est clairement vrai.

## 4.3 Remarques

Il est intéressant de noter qu'un langage de mots contextuel est complètement défini par deux ensembles de dominos (ou deux langages de mots locaux) qui correspondent au langage hv-local. De plus, on peut déduire facilement le résultat suivant :

**Proposition 4.7** *Soit $L$ un langage de mots sur $\Sigma$. Le langage $L$ est contextuel si et seulement s'il existe $\Sigma' \supset \Sigma$, $A \in \mathrm{Rec}(\Sigma'^*)$ et une transduction rationnelle $\tau$ de $\Sigma'^*$ dans $\Sigma'^*$ préservant les longueurs tels que $L = \tau^*(A) \cap \Sigma^*$.*

En fait, on montre ceci à partir de deux lemmes sur les langages de figures.

**Lemme 4.8** *Soit $L \in \text{hv-Loc}(\Sigma^{**})$ un langage hv-local. Il existe $A \in \mathrm{Rec}(\Sigma^*)$, un sous-alphabet $\Sigma' \subseteq \Sigma$ et une transduction rationnelle $\tau$ de $\Sigma^*$ dans $\Sigma^*$ préservant les longueurs tels que :*

$$\forall f \in \Sigma^{n,m}$$
$$f \in L$$
$$\Updownarrow$$
$$\begin{cases} & f(n,1)\dots f(n,m) \in A \\ \forall 1 \leq i < n & f(i,1)\dots f(i,m) = \tau(f(i+1,1)\dots f(i+1,m)) \\ & f(1,1)\dots f(1,m) \in \Sigma'^* \end{cases}$$

*Preuve.* Le langage $L$ étant hv-local, il existe deux langages de mots locaux $C, D \in \mathrm{Loc}(\Sigma^*)$ tels que $L = C \oplus D$. On considère l'ensemble $\Delta \subseteq \{\Sigma \cup \{\#\}\}^2$ des facteurs de deux lettres autorisés dans $D$. Les mots pouvant apparaître sur la dernière ligne d'une figure de $L$ sont des mots de $C$ dont les lettres peuvent se situer en dernière position d'un mot de $D$. On pose :

$$\begin{aligned} \Sigma' &= \{a \in \Sigma \mid a\# \in \Delta\} \\ A &= C \cap \Sigma'^* \end{aligned}$$

Soient une figure $f \in L$ de taille $(n,m)$ et $1 < i \leq n$, au dessus de la $i^{\text{ème}}$ ligne $f(i,1)\dots f(i,m)$, le mot $f(i-1,1)\dots f(i-1,m)$ est un mot de $C$ tel que :

$$\forall 1 \leq j \leq m \quad f(i+1,j)f(i,j) \in \Delta$$

On définit $\tau$ en utilisant le théorème de Nivat [Niv68] (voir Théorème 4.1 page 66 de [Ber79]) :

$$\forall \omega \in \Sigma^* \quad \tau(\omega) = \psi(\varphi^{-1}(\omega) \cap K)$$

où $\varphi$ et $\psi$ sont deux homomorphismes de $\Sigma''^*$ dans $\Sigma^*$ et $K$ un langage régulier de $\Sigma''^*$. Cet alphabet $\Sigma''$ est défini par les facteurs autorisés dans $D$ :

$$\Sigma'' = \{(a,b) \in \Sigma \times \Sigma \mid ab \in \Delta\}$$

Les homomorphismes $\varphi$ et $\psi$ et le langage $K$ sont alors facilement définis :

$$\begin{array}{c} \forall (a,b) \in \Sigma'' \quad \varphi((a,b)) = b \\ \forall (a,b) \in \Sigma'' \quad \psi((a,b)) = a \\ K = \psi^{-1}(C) \end{array}$$

En appliquant $\varphi^{-1}$ à un mot $x_1 \dots x_k$, on génère les mots $(y_1, x_1) \dots (y_k, x_k)$, où les mots $y_i x_i$ sont autorisés dans $D$. En faisant l'intersection avec $K$, on ne retient que les mots où $y_1 \dots y_k$ est dans $C$. Enfin en appliquant $\psi$ on ne retient que la première composante, c'est-à-dire le mot $y_1 \dots y_k$.

La transduction $\tau$ permet donc de passer de la ligne $f(i,1) \dots f(i,m)$ à la ligne $f(i-1,1) \dots f(i-1,m)$. Reste à vérifier que l'on peut placer des $\#$ au-dessus de la ligne $f(1,1) \dots f(1,m)$. Pour ceci, il suffit de n'autoriser dans $\Sigma'$ que des lettres qui peuvent apparaître en première position d'un mot de $D$ :

$$\Sigma' = \{a \in \Sigma \mid \#a \in \Delta\}^*$$

□

La réciproque de ce résultat n'est pas exacte, mais on peut l'adapter aux langages de figures reconnaissables.

**Lemme 4.9** *Soient $A \in \mathrm{Rec}(\Sigma^*)$, $\Sigma'$ un sous-alphabet de $\Sigma$ et $\tau$ une transduction rationnelle de $\Sigma^*$ dans $\Sigma^*$ préservant les longueurs, il existe un langage de figures reconnaissable $L \in \mathrm{Rec}(\Sigma^{**})$ tel que :*

$$\begin{array}{c} \forall f \in \Sigma^{n,m} \\ f \in L \\ \Updownarrow \\ \left\{ \begin{array}{ll} & f(n,1) \dots f(n,m) \in A \\ \forall 1 \leq i < n & f(i,1) \dots f(i,m) = \tau(f(i+1,1) \dots f(i+1,m)) \\ & f(1,1) \dots f(1,m) \in \Sigma'^* \end{array} \right. \end{array}$$

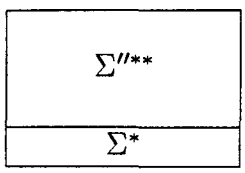
*Preuve.* Dans un premier temps on construit un langage de figures reconnaissable $L' \in \mathrm{Rec}(\Sigma^{**})$ qui vérifie :

$$\begin{array}{c} \forall f \in \Sigma^{n,m} \\ f \in L' \\ \Updownarrow \\ \left\{ \begin{array}{ll} & f(n,1) \dots f(n,m) \in A \\ \forall 1 \leq i < n & f(i,1) \dots f(i,m) = \tau(f(i+1,1) \dots f(i+1,m)) \end{array} \right. \end{array}$$

Comme $\tau$ préserve les longueurs, on sait (Théorème 4.2 page 34 de [AB88] — voir aussi [Leg80]) qu'il existe deux homomorphismes strictement alphabétiques $\varphi$ et $\psi$ de $\Sigma''^*$ dans $\Sigma'' *$ et un langage rationnel $K$ tels que :

$$\forall \omega \in \Sigma^* \quad \tau(\omega) = \psi(\varphi^{-1}(\omega) \cap K)$$

On peut supposer que $\Sigma$ et $\Sigma''$ sont disjoints et on définit $K'$ un langage de figures local par $K' = C \oplus D$ avec $C$ et $D$ deux langages de mots locaux. Le langage $D$ sera inclus dans $\Sigma''^*.\Sigma$, les figures de $K'$ seront donc de la forme :




Horizontalement, on autorise les mots de $A$ (pour la dernière ligne uniquement) et de $K$ (pour les autres lignes) : $C = A \cup K$. Verticalement, il faut tester qu'il s'agit bien de l'application des homomophismes. Les facteurs autorisés dans $D$ seront donc :

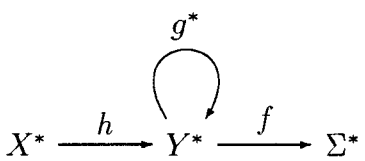
$$\begin{aligned} \Delta \;=\; & \{\#a \mid a \in \Sigma \cup \Sigma''\} \\ & \cup \{ab \in \Sigma''^2 \mid a \in \varphi^{-1}(\psi(b))\} \\ & \cup \{ab \in \Sigma''.\Sigma \mid a \in \varphi^{-1}(b)\} \\ & \cup \{a\# \mid a \in \Sigma\} \end{aligned}$$

On définit la projection $\pi : \Sigma \cup \Sigma'' \to \Sigma$ telle que $\pi(a) = a$ si $a \in \Sigma$ et $\pi(a) = \psi(a)$ si $a \in \Sigma''$. Le langage $L'$ est reconnaissable puisque $L' = \pi(K')$.

Pour achever la preuve il suffit de faire l'intersection de $L'$ avec le langage reconnaissable $L'' = (\Sigma'^{\Phi *} \ominus \Sigma^{**} \cup \Sigma'^{\Phi *})$. Le langage de figures $L$ est donc reconnaissable car on a $L = L' \cap L''$. □

Dans le même ordre d'idées, on peut caractériser les langages contextuels uniquement à l'aide de transductions rationnelles conservant les longueurs. Cette caractérisation se dérive directement de la Proposition 4.7.

**Proposition 4.10** *Soit $L \subseteq \Sigma^*$ un langage de mots. $L$ est contextuel si et seulement si il existe deux alphabets $X, Y$ et trois transductions rationnelles préservant les longueurs $f : Y^* \to \Sigma^*$, $g : Y^* \to Y^*$ et $h : X^* \to Y^*$ tels que $L = f \circ g^* \circ h(X^*)$ :*



Il est aussi intéressant de noter que la classe des transductions rationnelles de type $fg^*h$ où $f$, $g$ et $h$ sont des transductions rationnelles conservant les longueurs est close par composition et contient les transductions rationnelles conservant les longueurs ainsi que les itérations de transductions rationnelles conservant les longueurs.

Ce type de « mappings » itérés a déjà été étudié, notamment par les physiciens pour connaître l'ensemble des états atteignables par un système dynamique tel qu'un automate cellulaire [LMN97, Sei77] , nous reviendrons sur ce point dans le Chapitre 5.

On sait que les contextuels sont clos par complémentaire [Imm88, Sze88], mais les langages de figures reconnaissables ne le sont pas (voir Proposition 1.9). Néanmoins en utilisant une version adaptée du Théorème d'Immerman-Szelepzényi on peut construire, pour un langage de figures reconnaissable $L$, un langage de figures reconnaissable $K$ tel que $\mathrm{fr}(K) = \overline{\mathrm{fr}(L)}$.

En utilisant l'indécidabilité de la vacuité dans les langages contextuels, nous obtenons le corollaire suivant qui est aussi montré dans [GR92, GR96].

**Corollaire 4.11** *Le problème de la vacuité pour les langages de figures reconnaissables et pour les langages de figures hv-locaux est indécidable.*

**Note:** Les auteurs remercient W. Thomas pour les avoir informés que le Théorème 4.6 apparaît déjà (dans une terminologie quelque peu différente) dans le mémoire non-publié [Spe85].

# Références

[AB88] J.-M. Autebert and L. Boasson. *Transductions Rationnelles - Application aux Langages Algébriques.* Masson, Paris, 1988.

[Ber79] J. Berstel. *Tranductions and Context-Free Languages.* Teubner Studienbücher, Stuttgard, 1979.

[GR92] D. Giammarresi and A. Restivo. Recognizable picture languages. *International Journal Pattern Recognition and Artificial Intelligence*, pages 31–46, 1992. Special Issue on *Parallel Image Processing.* M. Nivat, A. Saoudi and P.S.P. Wangs (Eds.).

[GR96] D. Giammarresi and A. Restivo. Two-dimensional languages. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3, pages 215–267. Springer-Verlag, Berlin, 1996.

[Imm88] Neil Immerman. Nondeterministic space is closed under complementation. *SIAM J. Comput.*, 17(5):935–938, October 1988.

[Leg80] J. Leguy. *Transductions Rationnelles Décroissantes et Substitution.* PhD thesis, Univ. Lille 1, France, June 1980.

[LMN97] K. Lindgren, C. Moore, and M. Nordahl. Complexity of two-dimensional patterns. Technical Report 97–03–023, Santa Fe Institute, Unit. States, 1997.

[LS] M. Latteux and D. Simplot. Context-sensitive string languages and recognizable picture languages. *Information and Computation.* à paraître.

[MW67] J. Mezei and J. B. Wright. Algebraic automata and context-free sets. *Information and Control*, 11(2/3):3–29, 1967.

[Niv68] M. Nivat. Transductions des langages de Chomsky. *Ann. de l'Inst. Fourier*, 18:339–456, 1968.

[Sei77] J. I. Seiferas. Linear-time computation by nondeterministic multidimensional iterative arrays. *SIAM J. Comput.*, 6(3):487–504, 1977.

[Spe85] H. Sperber. Idealautomaten. Dissertation, Univ. Erlangen-Nürnberg, in german, 2985.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26(3):279–284, 1988.

[Tho97] W. Thomas. Automata theory on trees and partial orders. In *Proc. TAPSOFT'97*, volume 1214 of *Lecture Notes in Computer Science*, pages 20–34, Lille, France, 1997. Springer-Verlag, Berlin.

# Chapitre 5

# Frontière des langages de figures

Nous avons vu dans le chapitre précédent que la classe des frontières des langages de figures reconnaissables correspondait exactement à la classe des langages contextuels. Mieux, nous avons vu que la famille des frontières des langages de figures hv-locaux était la famille des contextuels. Un langage de figures hv-local $K$ est complètement défini par la donnée de deux langages de mots locaux $L_h$ et $L_v$ tels que $K = L_h \oplus L_v$ (voir section 2.3). Il est donc naturel de se demander à quelle classe de langages appartiennent les frontières des langages de figures de type $A \oplus B$ avec $A$ et $B$ dans $\{\text{Loc}, \text{Rec}, \text{Lin}, \text{Alg}, \text{CS}, \text{r.é.}\}$. Cette hiérarchie de Chomsky « croisée » a déjà mentionnée dans [GR96].

Comme les figures de taille nulle sont exclues dans les langages de figures, on ne peut obtenir le mot vide comme frontière d'une figure. C'est pourquoi, nous ne considérons que les langages de mots récursivement énumérables qui ne contiennent pas le mot vide.

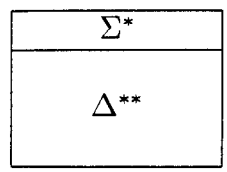
Enfin, pour un langage de figures reconnaissable $L$, la hauteur d'une figure peut être vue comme le temps de calcul nécessaire pour savoir si le mot sur la frontière supérieure est dans le langage $\text{fr}(L)$. Si nous ne limitons pas le temps de calcul, le Théorème 4.6 nous dit que $\text{fr}(K)$ est contextuel. Néanmoins, on peut s'interroger sur la nature du langage-frontière d'un langage de figures qui ne contient que des carrés, c'est-à-dire où le calcul se fait en temps-réel. Ceci nous amènera à faire des analogies avec les langages reconnus par automates cellulaires.

## 5.1 Langages contextuels

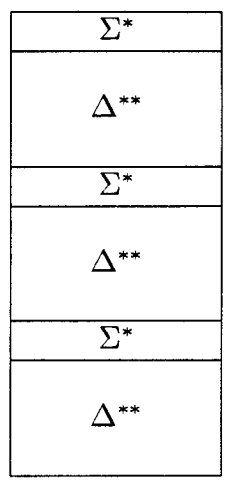
Il est clair que pour tout langage $L$ de type $A \oplus B$, si l'on a $A \notin \text{CS}(\Sigma^*)$ et $B$ quelconque, on n'a pas nécessairement $\text{fr}(L) \in \text{CS}(\Sigma^*)$ (par exemple, si l'on prend $B = \Sigma$, on a $\text{fr}(L) = A$). Néanmoins, si $A$ est contextuel, comme tout langage contextuel est la frontière d'un langage de figures hv-local, on peut construire effectivement un langage de figures $L'$ correspondant à $A' \oplus B'$ avec $A'$ local et $B'$ du même type que $B$.

**Lemme 5.1** *Soit* $L \in \text{CS}(\Sigma^*) \oplus \psi(\Sigma^*)$, *avec* $\psi \in \{\text{Loc}, \text{Rec}, \text{Lin}, \text{Alg}, \text{CS}, \text{r.é.}\}$. *Il existe* $\Sigma' \supseteq \Sigma$, $L' \in \text{Loc}(\Sigma'^*) \oplus \psi(\Sigma'^*)$ *tels que* $\text{fr}(L) = \text{fr}(L')$.

*Preuve.* Soit $L = A \oplus B$ avec $A \in \mathrm{CS}(\Sigma^*)$ et $B \in \psi(\Sigma^*)$. Comme $A \in \mathrm{CS}(\Sigma^*)$, il existe $\Sigma' \supseteq \Sigma$, $K \in \mathrm{Loc}(\Sigma'^*) \oplus \mathrm{Loc}(\Sigma'^*) = \text{hv-Loc}(\Sigma'^{**})$ tels que $\mathrm{fr}(K) = A$. De plus on peut supposer que $\Sigma' = \Sigma \cup \Delta$ (avec $\Sigma \cap \Delta = \emptyset$) et que $K = H \oplus V$ avec $H \subseteq \Sigma^* \cup \Delta^*$ et $V \subseteq \Sigma.\Delta^*$, c'est-à-dire que les figures de $K$ sont toutes de la forme :




On pose $A' = H$ et $B' = V^* \cap (B \sqcup \Delta^*)$. Les figures de $L' = A' \oplus B'$ ont la forme suivante :




Chaque ligne d'une figure appartient soit à $\Sigma^*$ soit à $\Delta^*$. Celles qui sont sur $\Sigma$ sont des mots de $A$ car elles sont la frontière d'une figure de $K$. Chaque colonne $c$ d'une telle figure est dans $(\Sigma.\Delta^*)^*$ et on a $\Pi_\Sigma(c) \in B$. Ainsi, on a $\mathrm{fr}(L') = \mathrm{fr}(L)$. □

Ceci nous permet d'étendre le Théorème 4.6 :

**Proposition 5.2** *Pour tout* $\varphi \in \{\mathrm{Loc}, \mathrm{Rec}, \mathrm{Lin}, \mathrm{Alg}, \mathrm{CS}\}$ *et* $\psi \in \{\mathrm{Loc}, \mathrm{Rec}\}$, *on a :*

$$\mathrm{fr}(\varphi \oplus \psi) \in \mathrm{CS}$$

Si l'on n'autorise que des reconnaissables verticalement, nous restons donc dans les contextuels, toutefois, placer un langage linéaire verticalement nous fait passer des langages contextuels aux langages récursivement énumérables, c'est ce que nous montrons dans la section suivante.

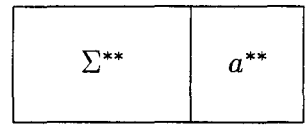
## 5.2 Langages récursivement énumérables

Nous allons d'abord nous assurer qu'en prenant des langages horizontaux et verticaux dans les récursivement énumérables nous ne sortons pas de cette classe.

**Lemme 5.3** *Soit* $L \in \text{r.é.}(\Sigma^*) \oplus \psi(\Sigma^*)$, *avec* $\psi \in \{\text{Loc}, \text{Rec}, \text{Lin}, \text{Alg}, \text{CS}, \text{r.é.}\}$ *il existe* $\Sigma' \supseteq \Sigma$, $L' \in \text{Loc}(\Sigma'^*) \oplus \psi(\Sigma'^*)$ *et un homomorphisme* $h : \Sigma'^* \to \Sigma^*$ *tels que* $\text{fr}(L) = h(\text{fr}(L'))$.

*Preuve.* Soit $L = A \oplus B$ avec $A \in \text{r.é.}(\Sigma^*)$ et $B \in \psi(\Sigma^*)$. Comme $A$ est récursivement énumérable, il existe un langage contextuel $A' \subseteq \Sigma^*.a^*$ (avec $a \notin \Sigma$, on note $\Sigma' = \Sigma \cup \{a\}$) tel que $A = \Pi_\Sigma(A')$.

On considère le langage de figures $K = A'.a^* \oplus (B \cup a^*)$. Les figures de $K$ ont la forme suivante :



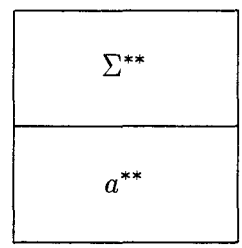

La sous-figure composée uniquement des lettres de $\Sigma$ est une figure de $L$. De même, pour toute figure de $L$, il existe une figure de $K$ qui la contient de cette manière. On a donc $\text{fr}(L) = \Pi_\Sigma(\text{fr}(K))$.

Maintenant $K \in \text{CS}(\Sigma'^*) \oplus \psi(\Sigma'^*)$, on peut donc appliquer le Lemme 5.1, il existe un langage de figures $L' \in \text{Loc}(\Sigma'^*) \oplus \psi(\Sigma'^*)$ tel que $\text{fr}(L') = \text{fr}(K)$. On a donc bien $\text{fr}(L) = \Pi_\Sigma(\text{fr}(L')$. □

**Lemme 5.4** *Soit* $L \in \varphi(\Sigma^*) \oplus \text{r.é.}(\Sigma^*)$, *avec* $\varphi \in \{\text{Loc}, \text{Rec}, \text{Lin}, \text{Alg}, \text{CS}, \text{r.é.}\}$ *il existe* $\Sigma' \supseteq \Sigma$, $L' \in \varphi(\Sigma'^*) \oplus \text{Loc}(\Sigma'^*)$ *et un homomorphisme* $h : \Sigma'^* \to \Sigma^*$ *tels que* $\text{fr}(L) = h(\text{fr}(L'))$.

*Preuve.* Soit $L = A \oplus B$ avec $A \in \varphi(\Sigma^*)$ et $B \in \text{r.é.}(\Sigma^*)$. Comme $B$ est récursivement énumérable, il est la projection sur $\Sigma$ d'un langage contextuel $C$ inclus dans $\Sigma^*.a^*$, avec $a \notin \Sigma$.

On pose $\Sigma'' = \Sigma \cup \{a\}$ et on construit un langage de figures $L''$ à partir du langage horizontal $A'' = A \cup a^*$ et du langage vertical $B'' = C.a^*$. Ainsi, les figures de $L'' = A'' \oplus B''$ ont la forme suivante :



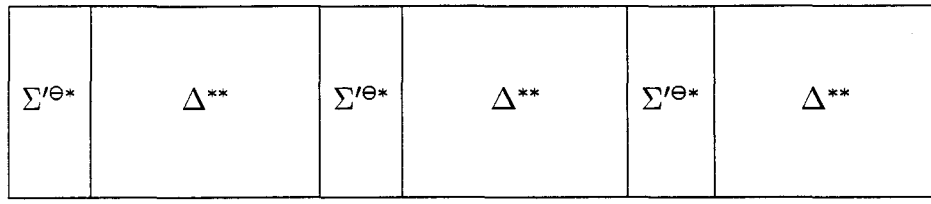

Il est facile de voir que les parties supérieures qui contiennent les lettres de $\Sigma$ des figures de $L''$ sont des figures de $L$ et que, réciproquement, pour toute figure $f$ de $L$, il existe une figure dans $L''$ qui appartient à $f \ominus a^{**}$. Ainsi, on a bien $\text{fr}(L'') = \text{fr}(L)$.

Le langage $B''$ étant contextuel, d'après le Théorème 4.6 on sait que $B''$ est la frontière d'un langage de figures $D \oplus E$ où $D$ et $E$ sont des langages de mots locaux sur un alphabet $\Sigma'$. De plus on peut s'assurer que $E \subseteq \Sigma'.\Delta^*$ avec $\Delta \cap \Sigma' = \emptyset$. En utilisant la même argumentation que dans la preuve du Lemme 5.1, et en posant :

$$\begin{aligned} A' &= E^* \cap (A'' \sqcup \Delta^*) \\ B' &= D \end{aligned}$$

on obtient bien $\mathrm{fr}(L) = \Pi_\Sigma(\mathrm{fr}(L'))$ avec $L' = A' \oplus B'$.

En effet les figures de $L'$ sont de la forme :




Le langage $B'$ est bien local, et si $\varphi \in \{\mathrm{Rec}, \mathrm{Lin}, \mathrm{Alg}, \mathrm{CS}, \text{r.é.}\}$ le langage $A'$ appartient bien à $\varphi(\Sigma'^*)$. Néanmoins, si $A$ est local, $A'$ ne l'est pas forcément aussi. Toutefois $A'$ est reconnaissable, donc le langage $L'$ est un langage de figurs reconnaissable. Il suffit de considérer le langage hv-local dont $L'$ est l'image. □

On a donc :

**Proposition 5.5** *Soit $L \subseteq \Sigma^*$ un langage de mots, $L$ est récursivement énumérable. si et seulement si il existe un alphabet $\Sigma' \supseteq \Sigma$, deux langages $A, B \in$ r.é.$(\Sigma'^*)$ tels que $L = \mathrm{fr}(A \oplus B)$.*

*Preuve.* Il est clair que pour $L \in$ r.é.$(\Sigma^*)$, on a $L = \mathrm{fr}(L \oplus \Sigma^*)$. Maintenant pour $A, B \in$ r.é.$(\Sigma'^*)$, en utilisant les Lemmes 5.3 et 5.4 et le Théorème 4.6, on a bien $\mathrm{fr}(A \oplus B) \in$ r.é.$(\Sigma'^*)$. □

La Proposition 5.2 nous dit que si l'on se limite aux contextuels horizontalement et aux reconnaissables verticalement, la frontière du langage de figures reste dans les contextuels. Il est donc intéressant de s'interroger sur les parties de la hiérarchie « croisée » de Chomsky pour lesquelles on obtient les récursivement énumérables.

**Proposition 5.6** $\mathrm{fr}(\text{r.é.}(\Sigma^*) \oplus \mathrm{Rec}(\Sigma^*)) = \text{r.é.}(\Sigma^*)$.

*Preuve.* Soit $L =$ r.é.$(\Sigma^*)$ un langage récursivement énumérable, on a $L = \mathrm{fr}(L \oplus \Sigma)$. Dans l'autre sens, on utilise la Proposition 5.5 pour montrer que, pour $A \in$ r.é.$(\Sigma^*)$ et $B = \mathrm{Rec}(\Sigma^*)$, $\mathrm{fr}(A \oplus B) \in$ r.é.$(\Sigma^*)$. □

Afin de continuer nos investigations nous nous dotons de lemmes techniques.

**Lemme 5.7** *Soient* $L \in \varphi(\Sigma^*) \oplus \psi(\Sigma^*)$, *avec* $\varphi, \psi \in \{\text{Loc}, \text{Rec}, \text{Lin}, \text{Alg}, \text{CS}, \text{r.é.}\}$ *et* $K \in \text{Rec}(\Sigma^{**})$, *alors* $\exists \Sigma' \supseteq \Sigma, L' \in \varphi(\Sigma'^*) \oplus \psi(\Sigma'^*)$ *tels que* $\text{fr}(L') = \text{fr}(L \cap K)$.

*Preuve.* Soit $L = A \oplus B$ avec $A \in \varphi(\Sigma^*)$ et $B \in \psi(\Sigma^*)$. Soit $K = h(C \oplus D) \in \text{Rec}(\Sigma^{**})$ avec $C, D \in \text{Loc}(X^*)$ (on suppose $X \cap \Sigma = \emptyset$) et $h : X^* \to \Sigma^*$ un homomorphisme lettre-à-lettre. On note $K' = C \oplus D$.

On pose $L' = h^{-1}(A) \oplus h^{-1}(B)$, il est clair que $h(L') = L$ et que $h(L' \cap K') = L \cap K$. On peut écrire :

$$L' \cap K' = (h^{-1}(A) \cap C) \oplus (h^{-1}(B) \cap D)$$

Si l'on pose :

$$\begin{aligned} A' &= (h^{-1}(A) \cap C) \cup \Sigma^* \\ B' &= \Sigma.(h^{-1}(B) \cap D) \cap \{a.b.\Sigma'^* \mid a \in \Sigma, b \in \Sigma' \quad h(b) = a\} \end{aligned}$$

On a bien $\text{fr}(A' \oplus B') = \text{fr}(h(L' \cap K')) = fr(L \cap K)$ et de plus $A' \in \varphi(\Sigma'^*)$ et $B' \in \psi(\Sigma'^*)$ avec $\Sigma' = \Sigma \cup X$. □

Ainsi en mettant verticalement des langages linéaires, on peut approcher les langages récursivement énumérables.

**Proposition 5.8** *Pour tout langage* $L \subseteq X^2.X^*$ *récursivement énumérable qui ne contient que des mots de longueur strictement supérieure à deux, il existe un alphabet* $Y \supset X$ *et deux langages* $K \in Loc(Y^*), K' \in Lin(Y^*)$ *tels que* $L = fr(K \oplus K')$.

*Preuve.* Soit $L \subseteq X^2.X^*$ un langage récursivement énumérable, il existe $L_1, L_2 \in Lin((X \cup \{\#\})^*)$ tels que :

$$\begin{aligned} L &= L_1.L_2^{-1} \\ \text{avec} & \\ X &= \Sigma \cup \Delta \quad (\Sigma \cap \Delta = \emptyset) \\ L_1 &\subseteq \Sigma^*.\#.\Delta^+ \\ L_2 &\subseteq \#.\Delta^+ \end{aligned}$$

On pose $L_2' = h(L_2)$ avec $h(\#) = \$$ ($\$ \notin X$) et $h(a) = a$ pour $a \in X$.

On pose $X' = X \cup \{\#, \$, \&\}$, $A = \Sigma^* \cup \{a^n \mid a \in \Delta, n \in \mathbb{N}\} \cup \#.\$.\&^*$ et $A' = L_1 \cup \Sigma^*.L_2' \cup \Sigma^*.\&.\Delta^*$ (avec $\& \notin X$). Notons que l'on a $A \in Loc(X'^*)$ et $A' \in Lin(X'^*)$.

Une figure de $A \oplus A'$ a donc la forme :

| | | | | | | |
|---|---|---|---|---|---|---|
| $x_1$ | | | | | | |
| $x_2$ | | | | | | |
| $\vdots$ | $\Sigma^{**}$ | | | | | |
| $x_{i-1}$ | | | | | | |
| $x_i$ | | | | | | |
| # | \$ | & | & | & | & | & |
| $y_1$ | $y_1$ | $y_1$ | $y_1$ | $y_1$ | $y_1$ | $y_1$ |
| $y_2$ | $y_2$ | $y_2$ | $y_2$ | $y_2$ | $y_2$ | $y_2$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $y_j$ | $y_j$ | $y_j$ | $y_j$ | $y_j$ | $y_j$ | $y_j$ |

avec $x_1 \dots x_i.\#.y_1 \dots y_j \in L_1$ et $\$.y_1 \dots y_j \in L'_2$.

On considère l'ensemble $C$ des carrés dont les lettres sur les transversales sud-ouest nord-est ne contiennent qu'une seule lettre de $\Sigma$ :

| | | | | | |
|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ |
| $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
| $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ |
| $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ |
| $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ |
| $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ | $x_{11}$ |

On considère $C'$ le langage :

$$C' = C \ominus (\# \oplus \$ \cup \# \oplus \$ \oplus \&^{\oplus *}) \ominus \Delta^{**}$$

Il est clair que $C'$ est reconnaissable, ses figures sont de la forme :

| | | | | | |
|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ |
| $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
| $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ |
| $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ |
| $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ |
| $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ | $x_{11}$ |
| # | \$ | & | & | & | & |
| $\Delta^{**}$ | | | | | |

L'intersection de $A \oplus A'$ avec $C'$ donne des figures de la forme :

| | | | | | |
|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ |
| $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
| $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ |
| $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ |
| $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ |
| $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ | $x_{11}$ |
| # | \$ | & | & | & | & |
| $y_1$ | $y_1$ | $y_1$ | $y_1$ | $y_1$ | $y_1$ |
| $y_2$ | $y_2$ | $y_2$ | $y_2$ | $y_2$ | $y_2$ |
| $y_3$ | $y_3$ | $y_3$ | $y_3$ | $y_3$ | $y_3$ |

La frontière de ces figures est bien $L_1.L_2^{-1}$. En appliquant le Lemme 5.7, on sait qu'il existe $Y \supseteq X$, $K \in Loc(Y^*)$, $K' \in Lin(Y^*)$ tels que $fr(K \oplus K') = fr((A \oplus A') \cap C') = L$. □

Néanmoins on a bien une inclusion stricte de Lin $\oplus$ Rec dans les récursivement énumérables. Il est en effet bien connu que même l'appartenance des mots de longueur un est indécidable pour les récursivement énumérables. Or pour $A \in \text{Rec}(\Sigma^*)$ et $B \in \text{Lin}(\Sigma^*)$, une lettre $a$ de $\Sigma$ est dans $\text{fr}(A \oplus B)$ si et seulement si $a \in A \wedge B \cap a.(A \cap \Sigma)^* \neq \emptyset$ — ce qui est décidable. On a le même problème avec un langage vertical algébrique et un langage horizontal contextuel. Or, si horizontalement on utilise un récursivement énumérable $A$ et un langage vertical linéaire $B$, on ne peut pas construire $a.(A \cap \Sigma)^*$ et donc décider si $B \cap a.(A \cap \Sigma)^*$ est vide — c'est pourquoi la Proposition 5.5 est valable dans ce cas.

**Corollaire 5.9** *Pour tout langage récursivement énumérable $L \in$ r.é.$(X^*)$ il existe un alphabet $Y \supseteq X$ et deux langages $K \in \text{Loc}(Y^*)$, $K' \in \text{Lin}(Y^*)$ tels que $\#.L.\$ = fr(K \oplus K')$ avec $\#, \$ \notin X$.*

Nous avons déjà utilisé le fait que tout récursivement énumérable était l'image par un homomorphisme d'un langage contextuel. En utilisant la même idée que dans la Proposition 5.8 on montre qu'en mettant les contextuels verticalement on obtient les récursivement énumérables.

**Proposition 5.10** *Soit $L \subseteq \Sigma^*$ un langage de mots, $L$ est récursivement énumérable si et seulement si il existe un alphabet $\Sigma' \supseteq \Sigma$, deux langages $A \in \text{Loc}(\Sigma'^*)$ et $B \in \text{CS}(\Sigma'^*)$ tels que $L = \text{fr}(A \oplus B)$.*

*Preuve.* La Proposition 5.5 montrant déjà que la frontière du croisement d'un local avec un contextuel est un langage récursivement énumérable, il nous reste à montrer que pour tout récursivement énumérable $L$, il existe $A \in \text{Loc}(\Sigma'^*)$ et $B \in \text{CS}(\Sigma'^*)$ tels que $L = \text{fr}(A \oplus B)$.

Le langage $L$ est la projection sur $\Sigma$ d'un langage contextuel $L' \subseteq \Sigma^*.a^*$ (avec $a \notin \Sigma$). On pose $\Sigma'' = \Sigma \cup \{a, b\}$ et on définit deux langages de mots $A'$ et $B'$ :

$$A' = \Sigma^* \cup a.b^*$$
$$B' = L' \cup \Sigma^*.b^*$$

Le langage $A'$ est un langage local et $B'$ est un langage contextuel. Ainsi en posant $K = A' \oplus B'$, on obtient toutes les figures de la forme (avec les $x_i$ des lettres de $\Sigma$) :

| | |
|---|---|
| $x_1$ | $\Sigma^{**}$ |
| $x_2$ | |
| $x_3$ | |
| $x_4$ | |
| $x_5$ | |
| $a$ | $b^{**}$ |
| $a$ | |
| $a$ | |

où la première colonne est un mot de $L'$.

En utilisant le langage de figures reconnaissable $C$ défini dans la preuve de la Proposition 5.8 qui contient tous les carrés sur $\Sigma$ où les transversales nord-est sud-est ne contiennent qu'une lettre :

| | | | | | |
|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ |
| $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
| $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ |
| $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ |
| $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ |
| $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ | $x_{11}$ |

On pose $C' = C \cup C \ominus \{a, b\}^{**}$ et $K' = K \cap C'$, on obtient les figures de la forme :

| | | | | |
|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
| $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ |
| $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
| $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ |
| $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ |
| $a$ | $b$ | $b$ | $b$ | $b$ |
| $a$ | $b$ | $b$ | $b$ | $b$ |
| $a$ | $b$ | $b$ | $b$ | $b$ |

La frontière de $K'$ est bien exactement $L$. Ce langage de figures est l'intersection d'un langage de figures reconnaissable avec un langage de $\mathrm{Loc}(\Sigma''^*) \oplus \mathrm{CS}(\Sigma''^*)$, donc d'après le Lemme 5.7, il existe bien un alphabet $\Sigma' \supset \Sigma'' \supset \Sigma$ et deux langages $A \in \mathrm{Loc}(\Sigma'^*)$ et $B \in \mathrm{CS}(\Sigma'^*)$ tels que $L = \mathrm{fr}(A \oplus B)$. □

Un tableau récapitulatif des ces résultats est donné Table 5.1.

| | | verticalement | | | | | |
|---|---|---|---|---|---|---|---|
| | $\oplus$ | Loc | Rec | Lin | Alg | CS | r.é. |
| horizontalement | Loc | CS | CS | r.é.† | r.é.† | r.é. | r.é. |
| | Rec | CS | CS | r.é.† | r.é.† | r.é. | r.é. |
| | Lin | CS | CS | r.é.† | r.é.† | r.é. | r.é. |
| | Alg | CS | CS | r.é.† | r.é.† | r.é. | r.é. |
| | CS | CS | CS | r.é.† | r.é.† | r.é. | r.é. |
| | r.é. | r.é. | r.é. | r.é. | r.é. | r.é. | r.é. |

† Dans ce cas un langage récursivement énumérable $L$ appartient à cette classe si et seulement si l'on peut décider l'appartenance des mots de longueur 1 à $L$.

TAB. 5.1 - *Tableau récapitulatif pour la frontière de la hiérarchie de Chomsky « croisée »*

## 5.3 Remarques sur les automates cellulaires

Les langages de figures semblent être un bon modèle de calcul pour les langages de mots. Ce n'est pas sans rappeler les automates cellulaires. Ceci n'est pas surprenant puisque les langages de figures reconnaissables sont exactement les langages de figures reconnus par automates mosaïque en ligne [IN77, IT91, GR92].

**Définition 5.11** *Un automate mosaïque en ligne est défini par un t-uple*

$$A = (\Sigma, Q, q_0, F, \delta)$$

*avec :*

- $\Sigma$, *l'alphabet d'entrée ;*
- $Q$, *un ensemble fini d'états ;*
- $q_0 \in Q$, *l'état initial ;*
- $F \subseteq Q$, *l'ensemble d'états finaux ;*
- $\delta : Q \times Q \times (\Sigma \cup \{\#\}) \to 2^Q$, *la fonction de transition.*

Un calcul de $A$ sur une figure $f$ de taille $(m, n)$ consiste à associer un état à chaque position $(i, j)$ $(i, j \geq 2)$ de $\tilde{f}$. Cet état $q$ est donné par la fonction de transition des états $q_u$ et $q_l$ respectivement déjà associés aux positions $(i-1, j)$ et $(i, j-1)$ et de la lettre $a$ à la position $\tilde{p}(i, j)$ : $q \in \delta(q_u, q_l, a)$. Initialement seules les positions $(i, j)$ avec $i = 1$ ou $j = 1$ sont étiquetées par un état qui est l'état initial. Une figure est acceptée par $A$ s'il existe un calcul qui termine avec un état final sur la position $(m+2, n+2)$.

*Exemple. L'automate mosaïque en ligne $A$ qui reconnaît l'ensemble des carrés sur $\Sigma = \{a\}$ peut être défini par $A = (\Sigma, Q, q_0, F, \delta)$ avec :*

- $Q = \{q_0, q_1, q_2, q_3\}$ ;
- $F = \{q_1\}$ ;
- *la fonction de transition $\delta$ est définie de la manière suivante pour $x \in \{a, \#\}$ :*

$$\begin{aligned}
\delta(q_0, q_0, x) &= \{q_1\} \\
\delta(q_0, q_1, x) &= \{q_2\} \\
\delta(q_0, q_2, x) &= \{q_2\} \\
\delta(q_0, q_3, x) &= \emptyset \\
\delta(q_1, q_0, x) &= \{q_3\} \\
\delta(q_1, q_1, x) &= \emptyset \\
\delta(q_1, q_2, x) &= \emptyset \\
\delta(q_1, q_3, x) &= \{q_3\} \\
\delta(q_2, q_0, x) &= \emptyset \\
\delta(q_2, q_1, x) &= \{q_2\} \\
\delta(q_2, q_2, x) &= \{q_2\} \\
\delta(q_2, q_3, x) &= \{q_1\} \\
\delta(q_3, q_0, x) &= \{q_3\} \\
\delta(q_3, q_1, x) &= \emptyset \\
\delta(q_3, q_2, x) &= \emptyset \\
\delta(q_3, q_3, x) &= \{q_3\}
\end{aligned}$$

*Il s'agit ici d'un automate déterministe puisque $\delta$ ne renvoie jamais plus d'un état. Considérons la figure de taille $(4,4)$ sur $\Sigma$. Le calcul commence sur la configuration suivante :*

| | | | | | |
|---|---|---|---|---|---|
| $\#, q_0$ | $\#, q_0$ | $\#, q_0$ | $\#, q_0$ | $\#, q_0$ | $\#, q_0$ |
| $\#, q_0$ | $a$ | $a$ | $a$ | $a$ | $\#$ |
| $\#, q_0$ | $a$ | $a$ | $a$ | $a$ | $\#$ |
| $\#, q_0$ | $a$ | $a$ | $a$ | $a$ | $\#$ |
| $\#, q_0$ | $a$ | $a$ | $a$ | $a$ | $\#$ |
| $\#, q_0$ | $\#$ | $\#$ | $\#$ | $\#$ | $\#$ |

*Au premier pas de calcul la seule position où l'on peut déterminer l'état est la position $(2,2)$. Comme $\delta(q_0, q_0, a) = \{q_1\}$, on a :*

| | | | | | |
|---|---|---|---|---|---|
| $\#, q_0$ | $\#, q_0$ | $\#, q_0$ | $\#, q_0$ | $\#, q_0$ | $\#, q_0$ |
| $\#, q_0$ | $a, q_1$ | $a$ | $a$ | $a$ | $\#$ |
| $\#, q_0$ | $a$ | $a$ | $a$ | $a$ | $\#$ |
| $\#, q_0$ | $a$ | $a$ | $a$ | $a$ | $\#$ |
| $\#, q_0$ | $a$ | $a$ | $a$ | $a$ | $\#$ |
| $\#, q_0$ | $\#$ | $\#$ | $\#$ | $\#$ | $\#$ |

*Pour les quatres pas de calculs suivants, on peut déterminer les états pour les positions* $(2,3)$, *puis* $(2,4)$, *puis* $(2,5)$ *et* $(2,6)$ *:*

| | | | | | |
|---|---|---|---|---|---|
| $\#, q_0$ | $\#, q_0$ | $\#, q_0$ | $\#, q_0$ | $\#, q_0$ | $\#, q_0$ |
| $\#, q_0$ | $a, q_1$ | $a, q_2$ | $a, q_2$ | $a, q_2$ | $\#, q_2$ |
| $\#, q_0$ | $a$ | $a$ | $a$ | $a$ | $\#$ |
| $\#, q_0$ | $a$ | $a$ | $a$ | $a$ | $\#$ |
| $\#, q_0$ | $a$ | $a$ | $a$ | $a$ | $\#$ |
| $\#, q_0$ | $\#$ | $\#$ | $\#$ | $\#$ | $\#$ |

*On peut procéder ainsi ligne par ligne pour obtenir finalement :*

| | | | | | |
|---|---|---|---|---|---|
| $\#, q_0$ | $\#, q_0$ | $\#, q_0$ | $\#, q_0$ | $\#, q_0$ | $\#, q_0$ |
| $\#, q_0$ | $a, q_1$ | $a, q_2$ | $a, q_2$ | $a, q_2$ | $\#, q_2$ |
| $\#, q_0$ | $a, q_3$ | $a, q_1$ | $a, q_2$ | $a, q_2$ | $\#, q_2$ |
| $\#, q_0$ | $a, q_3$ | $a, q_3$ | $a, q_1$ | $a, q_2$ | $\#, q_2$ |
| $\#, q_0$ | $a, q_3$ | $a, q_3$ | $a, q_3$ | $a, q_1$ | $\#, q_2$ |
| $\#, q_0$ | $\#, q_3$ | $\#, q_3$ | $\#, q_3$ | $\#, q_3$ | $\#, q_1$ |

*Cette figure est acceptée tandis que pour la figure de taille* $(2,3)$, *la configuration finale est :*

| | | | | |
|---|---|---|---|---|
| $\#, q_0$ | $\#, q_0$ | $\#, q_0$ | $\#, q_0$ | $\#, q_0$ |
| $\#, q_0$ | $a, q_1$ | $a, q_2$ | $a, q_2$ | $\#, q_2$ |
| $\#, q_0$ | $a, q_3$ | $a, q_1$ | $a, q_2$ | $\#, q_2$ |
| $\#, q_0$ | $\#, q_3$ | $\#, q_3$ | $\#, q_1$ | $\#, q_2$ |

*qui est donc rejetée puisque la cellule inférieure droite ne contient pas un état final.*

Ceci laisse présager des liens étroits entre les automates cellulaires et les frontières des langages de figures reconnaissables. En effet, on peut associer à tout langage de figures reconnaissables un automate cellulaire non-déterministe qui reconnaît le langage $\mathrm{fr}(L)$.

**Définition 5.12** *Un automate cellulaire non-déterministe est un quatruplet*

$$A = (Q, \Sigma, \delta, F)$$

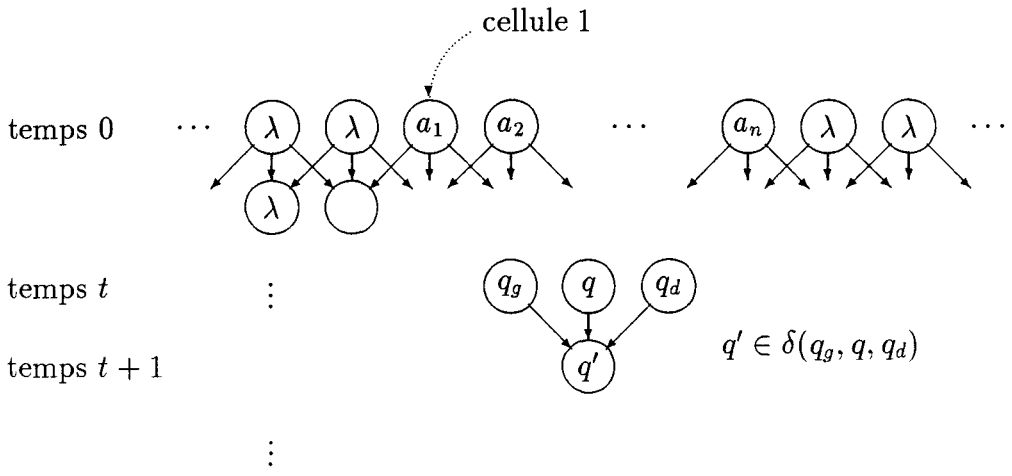



FIG. 5.1 - *Calcul d'un automate cellulaire sur le mot* $a_1 a_2 \dots a_n$

*où :*

- $Q$, *un ensemble fini d'états (l'ensemble* $Q$ *contient un état « quiescent » noté* $\lambda$*) ;*
- $\Sigma \subset Q$, *l'alphabet d'entrée (l'alphabet* $\Sigma$ *ne contient pas l'état quiescent) ;*
- $\delta : Q^3 \to 2^Q$, *la fonction de transition (pour l'état quiescent, on impose* $\delta(\lambda, \lambda, \lambda) = \lambda$*) ;*
- $F \subseteq Q$, *l'ensemble des états finaux.*

Le calcul de $A$ sur un mot $\omega = a_1 \dots a_n \in \Sigma^*$ ($a_i \in \Sigma$) consiste à créer un alignement supposé bi-infini de cellules (sur $\mathbb{Z}$). Au temps 0, toutes les cellules sont dans l'état quiescent $\lambda$ excepté les cellules 1 à $n$ : la $i^{\text{ème}}$ cellule $c(0,i)$ pour $1 \leq i \leq n$ est dans l'état $a_i$. À partir de la configuration au temps $t$, on utilise la fonction de transition $\delta$ pour calculer la configuration au temps $t+1$ (voir exemple Figure 5.1) :

$$\forall i \in \mathbb{Z} \quad c(t+1,i) \in \delta(c(t,i-1), c(t,i), c(t,i+1))$$

Le mot $\omega$ est accepté par l'automate cellulaire $A$ s'il existe un calcul partant de $\omega$ qui, à un temps $t$, a un état final sur la cellule $c(t,1)$. L'ensemble des mots acceptés par $A$ est noté $\mathcal{L}(A)$. La famille des langages de $\Sigma^*$ reconnus par automates cellulaires non-déterministes est notée $\mathrm{ACN}(\Sigma^*)$.

La plupart du temps on ne considère que les automates cellulaires déterministes et complètement spécifiés, c'est-à-dire que la fonction de transition renvoie exactement un état. La famille des langages reconnus par automates cellulaires (déterministes) est notée $\mathrm{AC}(\Sigma^*)$. Par définition, on a $\mathrm{AC}(\Sigma^*) \subseteq \mathrm{ACN}(\Sigma^*)$.

On peut s'obliger à limiter la taille de l'espace de calcul à la taille du mot d'entrée — en imposant par exemple $\delta(q_g, \lambda, \lambda) = \lambda$ et $\delta(\lambda, \lambda, q_d) = \lambda$. On a alors la classe des automates

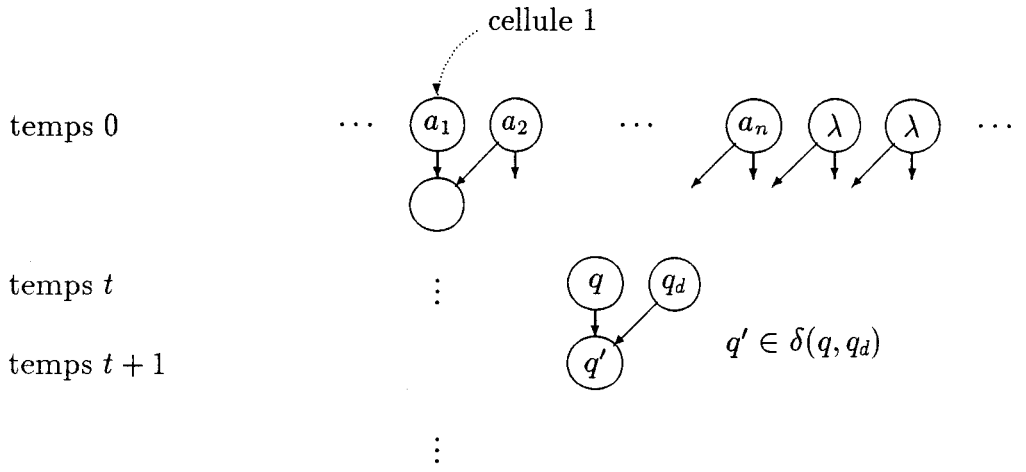



FIG. 5.2 - *Calcul d'un automate cellulaire unidirectionnel sur le mot* $a_1 a_2 \dots a_n$

cellulaires bornés. La famille des langages de $\Sigma^*$ reconnus par automates cellulaires bornés non-déterministes, respectivement déterministes, est notée $\mathrm{ACN}_b(\Sigma^*)$, resp. $\mathrm{AC}_b(\Sigma^*)$ [DMT].

Dans tous ces types d'automates, l'état d'une cellule à un temps $t > 0$ dépend de l'état de ses deux voisins (et d'elle-même) au temps $t-1$, un automate cellulaire unidirectionnel est un automate pour lequel le nouvel état ne dépend que de l'état de la cellule et de celui de son voisin de droite: $\forall q_g, q'_g q, q_d \in Q$ on a $\delta(q_g, q, q_d) = \delta(q'_g, q, q_d)$. C'est pourquoi l'on considère $\delta$ comme une fonction de $Q^2$ dans $2^Q$ (voir Figure 5.2). Notons que les langages reconnus par un automates cellulaires unidirectionnels sont également reconnus par automates unidirectionnels bornés: $\forall q \in Q$, on a $\delta(q, \lambda, \lambda) = \delta(\lambda, \lambda, \lambda) = \lambda$; de plus, l'état de la cellule $c(t, 1)$ ne dépendant que des états des cellules $c(t', i)$ avec $t' < t$ et $i \geq 1$, on peut ignorer les cellules en position négative ou nulle. La classe des langages de $\Sigma^*$ reconnus par automates cellulaires unidirectionnels non-déterministes, respectivement déterministes, est notée $\mathrm{ACUN}(\Sigma^*)$, resp. $\mathrm{ACU}(\Sigma^*)$.

En s'inspirant des automates mosaïque en ligne, on peut établir la proposition suivante:

**Proposition 5.13** *Soit* $L \in \mathrm{Rec}(\Sigma^{**})$ *un langage de figures reconnaissable, il existe un automate cellulaire unidirectionnel non-déterministe* $A$ *tel que* $\mathcal{L}(A) = \mathrm{fr}(L)$.

*Preuve.* Le langage $L \in \mathrm{Rec}(\Sigma^{**})$ est l'image d'un langage de figures hv-local $K \in \text{hv-Loc}(\Sigma'^{**})$ par une projection $\pi : \Sigma' \to \Sigma$. On peut supposer que $\Sigma$ et $\Sigma'$ sont disjoints.

Le principe est indentique à celui utilisé dans la preuve de la Proposition 4.3, pour un mot $\omega$ donné en entrée l'automate tente de manière non-déterministe de construire une figure de $K$ dont $\omega$ est l'image de la frontière.

L'automate cellulaire unidirectionnel non-déterministe $A$ est défini par $(Q, \Sigma, \delta, F)$ avec:

- $Q = \Sigma \cup \Sigma' \cup \{\#\} \cup ((\Sigma' \cup \{\#\} \times \{\#\}) \cup \{q_f, q_{erreur}\}$;

- $F = \{q_2\}$ ;
- la fonction de transition correspond à différentes étapes de la simulation :

1. générer un mot dont l'entrée est l'image par $\pi$ et vérifier que ce mot peut être sur la première ligne :

$$\forall q, q' \in \Sigma \quad \delta(q, q') = \left\{ q'', (q'', \#) \mid \boxed{\begin{array}{c}\#\\ q''\end{array}} \in \Delta \wedge q'' \in \pi^{-1}(q') \right\}$$

Dans un automate cellulaire unidirectionnel, la première cellule ne peut être identifiée en un seul pas de calcul. Pour ceci, chaque état de la première ligne peut se déclarer « première cellule » en prenant une valeur dans $\Sigma' \times \{\#\}$ et au pas de calcul suivant, une cellule ayant un voisin-droit qui s'est désigné première cellule se place dans un l'état $q_{erreur}$ ce qui empêche la suite du calcul d'aboutir.

$$\begin{array}{rrcl} \forall q, q' \in \Sigma' & \delta(q, (q', \#)) & = & \{q_{erreur}\} \\ \forall q \in Q & \delta(q_{erreur}, q) = \delta(q, q_{erreur}) & = & \{q_{erreur}\} \end{array}$$

2. vérifier les dominos horizontaux et générer la ligne suivante en vérifiant les connexions verticales :

$$\forall q, q' \in \Sigma' \quad \delta((q, \#), q') = \begin{cases} \left\{ (q'', \#) \mid \boxed{\begin{array}{c}q\\ q''\end{array}} \in \Delta \right\} & \text{si } \boxed{\#\,|\,q} \in \Delta \text{ et } \boxed{q\,|\,q'} \in \Delta \\ \{q_{erreur}\} & \text{sinon} \end{cases}$$

$$\forall q, q' \in \Sigma' \qquad \delta(q, q') = \begin{cases} \left\{ q'' \mid \boxed{\begin{array}{c}q\\ q''\end{array}} \in \Delta \right\} & \text{si } \boxed{q\,|\,q'} \in \Delta \\ \{q_{erreur}\} & \text{sinon} \end{cases}$$

$$\forall q \in \Sigma' \qquad \delta(q, \lambda) = \begin{cases} \left\{ q'' \mid \boxed{\begin{array}{c}q\\ q''\end{array}} \in \Delta \right\} & \text{si } \boxed{q\,|\,\#} \in \Delta \\ \{q_{erreur}\} & \text{sinon} \end{cases}$$

Ainsi une cellule peut décider de se placer dans l'état # qui signifie qu'elle anticipe la fin de la figure. Il faut donc vérifier que toute la ligne ne contient que des #. La dernière cellule (dont le voisin droit est $\lambda$) se place dans l'état acceptant $q_f$ et le propage à son voisin gauche jusqu'à la première cellule s'il n'y a pas eu d'erreurs :

$$\begin{aligned}
\delta(\#,\#) &= \{\#\} \\
\delta((\#,\#),\#) &= \{(\#,\#)\} \\
\forall q \in \Sigma' \quad \delta(q,\#) = \delta(\#,q) = \delta((\#,\#),q) &= \{q_{erreur}\} \\
\delta(\#,\lambda) = \delta((\#,\#),\lambda) &= \{q_f\} \\
\delta(\#,q_f) = \delta((\#,\#),q_f) &= \{q_f\} \\
\delta(q_f,q_f) &= \{q_f\}
\end{aligned}$$

L'automate simule ainsi la génération d'une figure de $K$ à partir du mot $\omega$ et il est clair que $\mathcal{L}(A) = \mathrm{fr}(L)$. □

La réciproque est également vraie ; pour tout automate cellulaire borné non-déterministe $A$, on peut construire un langage de figures reconnaissable dont la frontière est $\mathcal{L}(A)$. Pour ceci, on construit un langage de figures qui simule le fonctionnement de l'automate.

**Proposition 5.14** *Soit $A$ un automate cellulaire borné non-déterministe, il existe un langage de figures reconnaissable $L$ tel que $\mathrm{fr}(L) = \mathcal{L}(A)$. De plus pour tout mot dans $\mathcal{L}(A)$ pour tout calcul de l'automate on a une figure $f \in L$ telle que $\mathrm{fr}(f) = \omega$ et qui simule ce calcul.*

*Preuve.* Soit $A = (Q, \Sigma, \delta, F)$, on construit un langage de figures $(2,3)$-localement testable qui simule les calculs de $A$.

$$\Delta = \Delta_1 \cup \Delta_2 \cup \Delta_3$$

- l'ensemble $\Delta_1$ qui vérifie que la première ligne est bien un mot de $\Sigma^*$ :

$$\Delta_1 = \left\{ \begin{array}{|c|c|c|}\hline \# & \# & \# \\ \hline a & b & c \\ \hline \end{array} \mid a, c \in \Sigma \cup \{\#\}\ b \in \Sigma \right\}$$

- l'ensemble $\Delta_2$ qui simule les transitions de l'automate :

$$\begin{aligned}
\Delta_2 = & \left\{ \begin{array}{|c|c|c|}\hline \# & a & b \\ \hline \# & c & d \\ \hline \end{array} \mid a,b,c,d \in Q \wedge c \in \delta(\lambda,a,b) \right\} \\
& \cup \left\{ \begin{array}{|c|c|c|}\hline a & b & c \\ \hline d & e & f \\ \hline \end{array} \mid a,b,c,d,e,f \in Q \wedge e \in \delta(a,b,c) \right\} \\
& \cup \left\{ \begin{array}{|c|c|c|}\hline a & b & \# \\ \hline c & d & \# \\ \hline \end{array} \mid a,b,c,d \in Q \wedge d \in \delta(a,b,\lambda) \right\} \\
& \cup \left\{ \begin{array}{|c|c|c|}\hline \# & a & \# \\ \hline \# & b & \# \\ \hline \end{array} \mid a,b \in Q \wedge b \in \delta(\lambda,a,\lambda) \right\}
\end{aligned}$$

- l'ensemble $\Delta_3$ qui vérifie que la dernière ligne contient bien un état final sur la première cellule :

$$\Delta_3 = \left\{ \begin{array}{|c|c|c|}\hline \# & a & b \\ \hline \# & \# & \# \\ \hline \end{array} \mid a \in F, b \in Q \cup \# \right\} \cup \left\{ \begin{array}{|c|c|c|}\hline a & b & c \\ \hline \# & \# & \# \\ \hline \end{array} \mid a, b \in Q, c \in Q \cup \# \right\}$$

Le langage de figures $L$ localement testable qui correspond à $\Delta$ simule bien tous les calculs de $A$ qui aboutissent à une acceptation du mot de départ. □

Ainsi, par le biais des langages de figures, on peut aisément déduire du Théorème 4.6 et des Propositions 5.13 et 5.14 le corollaire suivant :

**Corollaire 5.15** $\mathrm{ACUN}(\Sigma^*) = \mathrm{ACN}_b(\Sigma^*) = \mathrm{fr}(\mathrm{Rec}(\Sigma^{**})) = \mathrm{CS}(\Sigma^*)$.

Alors que dans les automates cellulaires bornés on limite l'espace de travail, on peut également limiter le temps de calcul. On donne une fonction $f : \mathbb{N}^* \to \mathbb{N}^*$ et un calcul sur un mot de longueur $n$ est réussi si la première cellule contient un état final au temps $f(n)$ [MK93]. Par exemple, pour un automate cellulaire linéaire, on se fixe une fonction linéaire $f$ de $n$ et un mot $\omega$ n'est accepté que si au temps $f(|\omega|)$ la première cellule contient un état d'acceptation. La classe des langages sur $\Sigma$ reconnus par un automate cellulaire linéaire non-déterministe, respectivement déterministe, est noté $\mathrm{ACN}_l(\Sigma^*)$, resp. $\mathrm{AC}_l(\Sigma^*)$ [BC84, Sei77].

Un automate cellulaire temps réel est un automate cellulaire où la fonction de temps est $f(n) = n$. La classe des langages sur $\Sigma$ reconnus par un automate cellulaire temps réel non-déterministe, respectivement déterministe, est noté $\mathrm{ACN}_{tr}(\Sigma^*)$, resp. $\mathrm{AC}_{tr}(\Sigma^*)$ [Col69, Smi72, CC84, Ter93, Ter94, DMT].

Notons que par « pliage » on peut montrer que les langages reconnus par automates cellulaires linéaires et temps réel peuvent être reconnus en espace borné (et toujours en temps linéaire ou temps réel).

De même, on peut définir les automates cellulaires unidirectionnels linéaires et temps réel. On note les différentes classes de langages de $\Sigma^*$ reconnus par ces automates $\mathrm{ACUN}_l$, $\mathrm{ACU}_l$, $\mathrm{ACUN}_{tr}$ et $\mathrm{ACU}_{tr}$ [Pec83, Ter95, Ter96].

En utilisant les mêmes arguments que dans la preuve de la Proposition 5.14 et les fonctions reconnaissables par langages de figures reconnaissables [Gia93] — dont les fonctions polynomiales sont un exemple —, on montre la proposition suivante.

**Proposition 5.16** *Soit $A$ un automate cellulaire linéaire non-déterministe dont la fonction de temps est $f$, il existe un langage de figures reconnaissable $K$ tel que* $\mathrm{fr}(K) = \mathcal{A}$ *et :*

$$\forall p \in K \quad \mathrm{lig}(p) = f(\mathrm{col}(p)) + 1$$

De même d'après la Proposition 5.13, on obtient facilement :

**Proposition 5.17** *Soit $L$ un langage de figures reconnaissable pour lequel il existe une fonction linéaire $f : \mathbb{N}^* \to \mathbb{N}^*$ :*

$$\forall p \in L \quad \mathrm{lig}(p) = f(\mathrm{col}(p))$$

*il existe un automate cellulaire unidirectionnel linéaire non-déterministe dont la fonction de temps est $f(n) + n$.*

Ce facteur $n$ est ajouté en raison de la propagation de l'état $q_f$ de la $n^{\text{ème}}$ cellule vers la première cellule. Néanmoins, si pour tout $n \in \mathbb{N}^*$, $f(n) \geq n$ on peut par pliage éviter ce supplément de calcul.

**Proposition 5.18** *Soit $L$ un langage de figures reconnaissable pour lequel il existe une fonction linéaire $f : \mathbb{N}^* \to \mathbb{N}^*$ telle que pour tout $n \in \mathbb{N}^*$, $f(n) \geq n$ :*

$$\forall p \in L \quad \mathrm{lig}(p) = f(\mathrm{col}(p))$$

*il existe un automate cellulaire unidirectionnel linéaire non-déterministe dont la fonction de temps est $f(n)$.*

L'idée est de replier le carré inférieur de la figure de la manière suivante :

| | | | | |
|---|---|---|---|---|
| $a$ | $b$ | $b$ | $c$ | $a$ |
| $b$ | $c$ | $a$ | $a$ | $c$ |
| $a$ | $c$ | $a$ | $c$ | $c$ |
| $b$ | $a$ | $c$ | $a$ | $b$ |
| $a$ | $b$ | $a$ | $a$ | $b$ |
| $a$ | $c$ | $a$ | $b$ | $c$ |
| $c$ | $a$ | $c$ | $b$ | $a$ |

$\to$

| | | | | |
|---|---|---|---|---|
| $a$ | $b$ | $b$ | $c$ | $a$ |
| $b$ | $c$ | $a$ | $a$ | $c$ |
| $(a,a)$ | $(c,c)$ | $(a,b)$ | $(c,b)$ | $c$ |
| $(b,b)$ | $(a,b)$ | $(c,a)$ | $a$ | \$ |
| $(a,c)$ | $(b,a)$ | $a$ | \$ | \$ |
| $(a,a)$ | $c$ | \$ | \$ | \$ |
| $c$ | \$ | \$ | \$ | \$ |

**Proposition 5.19** *Soit $L$ un langage de figures reconnaissable pour lequel il existe une fonction $f : \mathbb{N}^* \to \mathbb{N}^*$ et une constante $k \in \mathbb{N}$ tel que pour tout $n \in \mathbb{N}^*$, $f(n) \geq n$ :*

$$\forall p \in L \quad \mathrm{lig}(p) = f(\mathrm{col}(p)) + k$$

*il existe un langage de figures reconnaissable $K$ tel que $\mathrm{fr}(K) = \mathrm{fr}(L)$ et :*

$$\forall p \in K \quad \mathrm{lig}(p) = f(\mathrm{col}(p))$$

*Preuve.* Il s'agit de regrouper les $k+1^{\text{èmes}}$ dernières lignes du langage hv-local $A \in \text{hv-Loc}(\Sigma'^{**})$ dont $L$ est l'image par une projection $\pi$. On considère l'alphabet $\Sigma'' = \Sigma' \cup \Sigma'^{k+1}$ et on définit l'ensemble de dominos autorisés $\Delta'$ en fonction de $\Delta$ qui est associé à $A$ :

$$\Delta' = \left(\Delta \setminus \left\{ \begin{array}{|c|}\hline a \\ \hline \# \\ \hline \end{array} \mid a \in \Sigma' \right\}\right)$$

$$\cup \left\{ \begin{array}{|c|}\hline a \\ \hline b_1 \ldots b_{k+1} \\ \hline \end{array} \mid \begin{array}{|c|}\hline a \\ \hline b_1 \\ \hline \end{array} \in \Delta \wedge \left( \forall 1 \leq i \leq k \quad \begin{array}{|c|}\hline b_i \\ \hline b_{i+1} \\ \hline \end{array} \in \Delta \right) \right\} \quad a \in \Sigma' \cup \{\#\} \quad \forall 1 \leq i \leq k+1 \; b_i \in \Sigma'$$

$$\cup \left\{ \begin{array}{|c|c|}\hline \# & a_1 \ldots a_{k+1} \\ \hline \end{array} \mid \begin{array}{l} \forall 1 \leq i \leq k+1 \; a_i \in \Sigma' \\ \forall 1 \leq i \leq k+1 \quad \begin{array}{|c|c|}\hline \# & a_i \\ \hline \end{array} \in \Delta) \end{array} \right\}$$

$$\cup \left\{ \begin{array}{|c|c|}\hline a_1 \ldots a_{k+1} & b_1 \ldots b_{k+1} \\ \hline \end{array} \mid \begin{array}{l} \forall 1 \leq i \leq k+1 \; a_i, b_i \in \Sigma' \\ \forall 1 \leq i \leq k+1 \quad \begin{array}{|c|c|}\hline a_i & b_i \\ \hline \end{array} \in \Delta) \end{array} \right\}$$

$$\cup \left\{ \begin{array}{|c|c|}\hline a_1 \ldots a_{k+1} & \# \\ \hline \end{array} \mid \begin{array}{l} \forall 1 \leq i \leq k+1 \; a_i \in \Sigma' \\ \forall 1 \leq i \leq k+1 \quad \begin{array}{|c|c|}\hline a_i & \# \\ \hline \end{array} \in \Delta) \end{array} \right\}$$

$$\cup \left\{ \begin{array}{|c|}\hline a_1 \ldots a_{k+1} \\ \hline \# \\ \hline \end{array} \mid \left( \forall 1 \leq i \leq k \quad \begin{array}{|c|}\hline b_i \\ \hline b_{i+1} \\ \hline \end{array} \in \Delta) \right) \wedge \begin{array}{|c|}\hline b_{k+1} \\ \hline \# \\ \hline \end{array} \in \Delta \right\} \quad \forall 1 \leq i \leq k+1 \; a_i \in \Sigma'$$

Considérons la projection $\pi'$ qui associe $\pi(a)$ à toute lettre $a$ de $\Sigma'$ et $\pi(a_1)$ à une lettre $a_1 \ldots a_{k+1}$ de $\Sigma'^{k+1}$. Notons $K$ l'image par $\pi'$ du langage hv-local associé à $\Delta'$. Les figures de $K$ sont les figures de $L$ tronquées des $k$ dernières lignes, on a donc bien $\mathrm{fr}(K) = \mathrm{fr}(L)$. □

**Proposition 5.20** *Soit $L$ un langage de figures reconnaissable pour lequel il existe une fonction $f : \mathbb{N}^* \to \mathbb{N}^*$ et une constante $k \in \mathbb{N}$ telles que pour tout $n \in \mathbb{N}^*$, $f(n) \geq n$ :*

$$\forall p \in L \quad \mathrm{lig}(p) = k.f(\mathrm{col}(p))$$

*il existe un langage de figures reconnaissable $K$ tel que $\mathrm{fr}(K) = \mathrm{fr}(L)$ et :*

$$\forall p \in K \quad \mathrm{lig}(p) = f(\mathrm{col}(p))$$

*Preuve.* Le langage $L \in \mathrm{Rec}(\Sigma^{**})$ est l'image d'un langage de figures hv-local $A \in \text{hv-Loc}(\Sigma'^{**})$ par une projection $\pi : \Sigma' \to \Sigma$. Cette fois, nous regroupons les lignes par paquets de $k$ lignes. Nous travaillons donc sur l'alphabet $\Sigma'' = \Sigma'^k$.

$$\Delta' = \left\{ \begin{array}{|c|c|}\hline \# & a_1 \ldots a_k \\ \hline \end{array} \mid \forall 1 \leq i \leq k \quad \begin{array}{|c|c|}\hline \# & a_i \\ \hline \end{array} \in \Delta \right\}$$

$$\cup\left\{ \boxed{a_1 \ldots a_k \mid b_1 \ldots b_k} \mid \forall 1 \leq i \leq k \quad \boxed{a_i \mid b_i} \in \Delta \right\}$$

$$\cup\left\{ \boxed{a_1 \ldots a_k \mid \#} \mid \forall 1 \leq i \leq k \quad \boxed{a_i \mid \#} \in \Delta \right\}$$

$$\cup\left\{ \boxed{\begin{array}{c}\#\\ a_1 \ldots a_k\end{array}} \mid \boxed{\begin{array}{c}\#\\ a_1\end{array}} \in \Delta \wedge \left( \forall 1 \leq i < k \quad \boxed{\begin{array}{c}a_i\\ a_{i+1}\end{array}} \in \Delta \right) \right\}$$

$$\cup\left\{ \boxed{\begin{array}{c}a_1 \ldots a_k\\ b_1 \ldots b_k\end{array}} \mid \left( \forall 1 \leq i < k \quad \boxed{\begin{array}{c}a_i\\ a_{i+1}\end{array}}, \boxed{\begin{array}{c}b_i\\ b_{i+1}\end{array}} \in \Delta \right) \wedge \boxed{\begin{array}{c}a_k\\ b_1\end{array}} \in \Delta \right\}$$

$$\cup\left\{ \boxed{\begin{array}{c}a_1 \ldots a_k\\ \#\end{array}} \mid \left( \forall 1 \leq i < k \quad \boxed{\begin{array}{c}a_i\\ a_{i+1}\end{array}} \in \Delta \right) \wedge \boxed{\begin{array}{c}a_k\\ \#\end{array}} \in \Delta \right\}$$

On considère la projection $\pi'$ qui associe $\pi(a)$ à toute lettre $a$ de $\Sigma'$ et $\pi(a_1)$ à une lettre $a_1 \ldots a_{k+1}$ de $\Sigma'^{k+1}$. Notons $K$ l'image par $\pi'$ du langage hv-local associé à $\Delta'$. Les figures de $K$ sont les figures de $L$ dont on n'a retenu qu'une ligne sur $k$. On a ainsi $\mathrm{fr}(K) = \mathrm{fr}(L)$. □

Ces deux propositions permettent de déduire facilement le résultat suivant où $C(\Sigma^{**})$ est la classe des langages de figures sur $\Sigma$ qui ne contiennent que des carrés.

**Proposition 5.21** $\mathrm{ACN}_l(\Sigma^*) = \mathrm{ACN}_{tr}(\Sigma^*) = \mathrm{ACUN}_l(\Sigma^*) = \mathrm{ACUN}_{tr}(\Sigma^*) = \mathrm{fr}(\mathrm{Rec}(\Sigma^{**}) \cap C(\Sigma^{**}))$.

Ces résultats ainsi que d'autres inclusions bien connues des classes de langages sont repris Figure 5.3. Dans ce DAG les flêches indiquent une inclusion au sens large, les inclusions strictes étant signalées par un signe « $\neq$ ». L'alphabet est supposé contenir plus d'une lettre et la classe $Q(\Sigma^*)$ représente la classe des langages quasi-temps réel [BG70] qui sont reconnus en temps linéaire par machine de Turing non-déterministe.

Enfin, notons que les langages reconnus par automates cellulaires peuvent être caractérisés par des transductions rationnelles comme nous l'indique le Lemme 4.8.

# Références

[BC84] W. Bucher and K. Culik II. On real time and linear time cellular automata. *RAIRO Informatique Théorique et Applications/Theoretical Informatics and Applications*, 81:307–325, 1984.

[BG70] R. V. Book and S. A. Greibach. Quasi-realtime languages. *Mathematical Systems Theory*, 4(2):97–111, 1970.

[CC84] C. Choffrut and K. Culik II. On real-time cellular automata and trellis automata. *Acta Informatica*, 21:393–407, 1984.

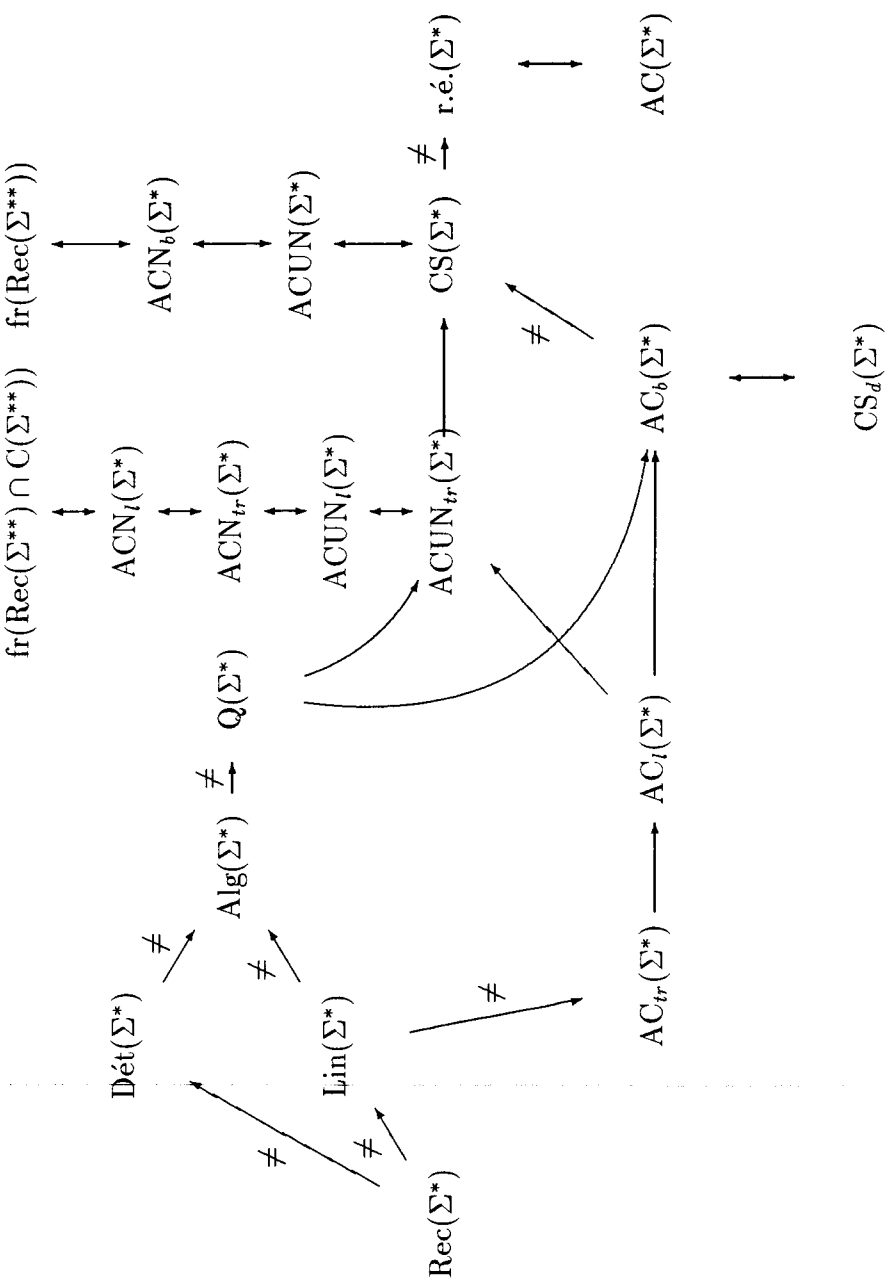



FIG. 5.3 - *Liens entre les différentes classes de langages*

[Col69] S. N. Cole. Real-time computation by n-dimensional iterative arrays of finite-state machine. *IEEE Transactions on Computing*, 18:349–365, 1969.

[DMT] M. Delorme, J. Mazoyer, and V. Terrier. Communications personnelles.

[Gia93] D. Giammarresi. Two-dimensional languages and recognizable functions. In G. Rozenberg and A. Salomaa, editors, *Proc. Developments in Language Theory*, pages 290–301, Turku, Finland, 1993. World Scientific Publishing Co.

[GR92] D. Giammarresi and A. Restivo. Recognizable picture languages. *International Journal Pattern Recognition and Artificial Intelligence*, pages 31–46, 1992. Special Issue on *Parallel Image Processing*. M. Nivat, A. Saoudi and P.S.P. Wangs (Eds.).

[GR96] D. Giammarresi and A. Restivo. Two-dimensional languages. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3, pages 215–267. Springer-Verlag, Berlin, 1996.

[IN77] K. Inoue and A. Nakamura. Some properties of two-dimensional on-line tesselation acceptors. *Information Sciences*, 13:95–121, 1977.

[IT91] K. Inoue and I. Takanami. A characterization of recognizable picture languages. In *Proc. 2nd International Colloquium on Parallel Image Processing*, volume 654 of *Lecture Notes in Computer Science*, pages 133–143. Springer-Verlag, Berlin, 1991.

[MK93] M. Mahajan and K. Krithivasan. Language classes defined time-bounded relativised cellular automate. *Informatique Théorique et Applications/Theoretical Informatics and Applications*, 27(5):403–432, 1993.

[Pec83] J. Pecht. On the real-time recognition of formal languages in cellular automata. *Acta Cybernetica*, 6(1):33–53, 1983.

[Sei77] J. I. Seiferas. Linear-time computation by nondeterministic multidimensional iterative arrays. *SIAM J. Comput.*, 6(3):487–504, 1977.

[Smi72] A. R. Smith III. Real-time language recognition by one-dimensional cellular automata. *Journal of the Assoc. Comput. Mach.*, 6:233–253, 1972.

[Ter93] V. Terrier. Real time recognition with cellular automata: a meaningful example. *Informatique Théorique et Applications/Theoretical Informatics and Applications*, 27(2):97–120, 1993.

[Ter94] V. Terrier. Language recognizable in real time by cellular automata. *Complex Systems*, 8:325–336, 1994.

[Ter95] V. Terrier. On real time one way cellular array. *Theoretical Computer Science*, 141(1/2):331–335, 1995.

[Ter96] V. Terrier. Language not recognizable in real time by one-way cellular automata. *Theoretical Computer Science*, 156(1/2):281–287, 1996.

# Conclusion

# Conclusion

Dans la première partie nous avons donné une manière de représenter le monoïde des figures connexes $\mathcal{F}_p^c$ à l'aide d'un monoïde libre. L'ensemble $\mathcal{F}_p^c$ nous paraît être un exemple pertinent de monoïde inversif — c'est-à-dire que l'on peut s'en inspirer pour déduire des propriétés des monoïdes inversifs.

Ainsi dans le Chapitre 2, nous donnons un système de réécriture générique qui engendre la congruence associée à un système générateur de tout monoïde inversif vérifiant une propriété donnée. Une limitation, déjà signalée, de ce système de réécriture est qu'il ne peut être appliqué si l'élément neutre du monoïde inversif possède des diviseurs. Par exemple, dans le monoïde des figures pointées $\mathcal{F}_p$, l'élément neutre $f_\varepsilon$, la boucle vide, possède une infinité de diviseurs (toute figure de taille nulle). Néanmoins, si l'on considère un système de réécriture $R$ où toutes les règles conservent l'élément décrit, dont la clôture transitive contient le système de réécriture $S$ donné Définition 2.6 (page 21) et qui vérifie la condition $(P)$ du théorème 2.8, il est clair que $R$ engendre la congruence. Il n'est d'ailleurs pas étonnant dans le cas des figures de segments avec traits invisibles que l'on puisse retrouver le système de réécriture de [Slo93].

Par contre, pour étendre la notion de complexité descriptive, étudiée Chapitre 3, aux systèmes générateurs, nous avons besoin de définir le poids des éléments du monoïde, définition qui devrait coïncider avec celle utilisée dans les figures. Une idée serait d'utiliser l'ordre naturel dans les monoïdes inversifs pour calculer un poids. On peut aussi considérer les monoïdes du type $A \times F$ où $A$ est l'ensemble des parties finies d'un ensemble $E$ et $F$ un groupe de bijections de $E$ dans $E$. Par exemple si l'on prend $E = \mathbb{Z}^2$ et $F$ l'ensemble des translations dans $\mathbb{Z}^2$, le monoïde $A \times F$ est exactement $\mathcal{F}_p$. Tous les monoïdes de ce type sont inversifs et le poids d'un élément $(X, f) \in A \times F$ peut, de manière naturelle, être défini comme le cardinal de $X$. Ces monoïdes, et surtout leurs sous-monoïdes finiment engendrés, semblent très analogues à $\mathcal{F}_p^c$ et méritent certainement notre attention.

Dans la seconde partie, nous avons considéré les langages de mots à deux dimensions. Nous donnons dans le Chapitre 2 une caractérisation des langages de figures reconnaissables utilisant le recouvrement par des dominos, ce qui permet de les définir par le « croisement » de deux langages de mots locaux.

L'un des problèmes les plus importants dans cette théorie est d'obtenir un équivalent du théorème de Kleene, c'est-à-dire de définir les reconnaissables à partir d'expressions régulières. Ceci paraît difficile et les différentes tentatives ne laissent présager que des résultats partiels ou très techniques [Mat97].

Nous avons vu dans les Chapitres 4 et 5 que les langages de figures peuvent être considérés comme modèle de calcul sur les mots. Notamment, les langages de figures reconnaissables peuvent être vus comme des calculs d'automates cellulaires. De ce point de vue, il subsiste des problèmes ouverts comme la comparaison des différentes classes d'automates cellulaires. Par exemple, on ne sait pas à l'heure actuelle si les automates cellulaires déterministes temps réel, linéaires ou bornés reconnaissent la même classe de langages. Enfin, on peut également s'interroger sur les inclusions éventuelles entre la classe des langages reconnus par automates cellulaires non-déterministes temps réel et la famille des langages reconnus par automates cellulaires déterministes bornés (voir Figure 5.3 page 144).

# Références

[Mat97] O. Matz. Regular expressions and contextfree grammars for picture languages. In *Proc. STACS'97*, volume 1200 of *Lecture Notes in Computer Science*, pages 283–294, Lübeck, Germany, 1997. Springer-Verlag, Berlin.

[Slo93] K. Slowinski. Picture words with invisible lines. *Theoretical Computer Science*, 108(2):357–363, 1993.

# Index

# Index

Les numéros de pages en chiffres romains (ex : vi) se rapportent à l'introduction, ceux en chiffres romains gras (ex : **xxv**) font référence au chapitre Notations. Les pages en roman droit (ex : 23) font référence à la partie sur les mots de figures tandis que les numéros de pages en italique (ex : *93*) se réfèrent à la partie sur les mots à deux dimensions. Enfin, les noms propres sont notés en italique (ex : *Autebert*).