50376
1993
183

50376
1993
183

N⁰ d'ordre : 1107

présentée à

L'UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE

pour obtenir

LE TITRE DE DOCTEUR DE L'UNIVERSITÉ

SPÉCIALITÉ : INFORMATIQUE THÉORIQUE

par

Jeanne DEVOLDER-MUCHEMBLED

Codes, mots infinis et bi-infinis

UNIVERSITE DES Sc. et Techn. LILLE
BIBLIOTHEQUE UNIVERSITAIRE

Soutenue le 5 mai 1993 devant la Commission d'Examen :

Président : J. BERSTEL, Université de Paris VI
Rapporteurs : D. BEAUQUIER, Université de Paris XII
J. BERSTEL, Université de Paris VI
Examinateurs : V. BRUYÈRE, Université de Mons (Belgique)
M. LATTEUX, Université de Lille I
I. LITOVSKY, ENSERB, Université de Bordeaux I
E. TIMMERMAN, Université de Lille I

U.F.R. Mathématiques Pures et Appliquées, bât. M2
Laboratoire d'Informatique Fondamentale de Lille, bât. M3
UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE
59655 VILLENEUVE D'ASCQ Cedex, FRANCE

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

A une époque de transition, j'ai trouvé dans une petite équipe d'informaticiens l'environnement idéal, à la fois stimulant et chaleureux, nécessaire pour me lancer dans une nouvelle voie de recherche. Des questions, d'apparence anodine, posées par Erick et Michel lors de la thèse d'Igor Litovsky sont à l'origine de ce travail.

Je tiens à exprimer à Erick, qui a dirigé ce travail, toute ma reconnaissance pour sa disponibilité, pour ses questions, suggestions et conseils enrichissants, pour ses exemples, son aide en général.

Je dois aussi à Michel toute ma gratitude, parce qu'il me fut un encouragement permanent, suivant avec beaucoup d'intérêt l'évolution du travail, ne ménageant ni son temps, ni son soutien, ni sa participation.

Mes plus vifs remerciements s'adressent également à Jean Berstel et Danièle Beauquier pour l'intérêt qu'ils ont montré pour mon travail, en acceptant très spontanément d'être rapporteurs de cette thèse.
J'apprécie tout particulièrement l'honneur que me fait Jean Berstel en présidant le jury.

Igor Litovsky faisait partie de l'équipe lilloise quand je me suis lancée dans ce domaine de recherche ; il est pour beaucoup dans l'intérêt que je porte aux mots infinis ; je l'en remercie. Je le remercie aussi pour sa participation au jury.

Je remercie également Véronique Bruyère pour le plaisir qu'elle me fait en participant au jury, et aussi pour l'organisation de diverses manifestations sur le thème des codes, au cours desquelles j'ai pu exposer quelques résultats.

Parvenir au terme de ce travail m'aurait été difficile sans le secours de mon entourage. Il m'est impossible de citer tous ceux qui m'ont aidée concrètement ou soutenue moralement. Je pense en particulier à Daniel et Carlos pour leur patience devant mon incompétence à utiliser latex, postcript..., à plusieurs collègues de proba-stat, à Nelly, à ceux de la piscine, ..., et à Pascal tout spécialement.
Je remercie finalement tous ceux qui, avec gentillesse, m'ont aidée, dans le travail quotidien, ou dans la réalisation matérielle de cette thèse.

Introduction

Les travaux présentés dans cette thèse constituent une étude des relations existant entre diverses classes de **codes** et les **factorisations de mots infinis**. Dans cette introduction, le terme de « mot infini » est utilisé comme un terme générique représentant à la fois les mots **infinis à droite** (suites de lettres indicées par l'ensemble $\boldsymbol{N}$ des entiers naturels), les $\boldsymbol{Z}$**-mots** (suites de lettres indicées par l'ensemble $\boldsymbol{Z}$ des entiers relatifs) et les mots **bi-infinis** (classes d'équivalence de Z-mots pour la relation de décalage).

La **théorie des codes** s'est développée dans plusieurs directions, à partir des années 50, à la suite des résultats de Shannon (1948) sur le codage des messages en vue de leur transmission [60]. L'une d'entre elles a mené à la théorie des codes correcteurs d'erreurs, théorie fructueuse qui a fait l'objet de nombreux travaux (pour une étude approfondie sur ces questions, on pourra se reporter à [48] et à sa bibliographie) ; mais ceci n'est pas l'objet de notre propos.

Le travail présenté ici est lié à la théorie des codes à longueur variable introduite par Schutzenberger en 1955 [58]. Un **code** (à longueur variable) est un langage de mots finis qui assure l'unicité du décodage des mots finis. L'étude des codes a été essentiellement menée dans l'optique du décodage des mots finis. On peut trouver dans [8] une partie des résultats de cette théorie. Nous nous intéressons ici au comportement des codes vis-à-vis des mots infinis.

L'évolution des états d'un système dynamique pendant un intervalle de temps discret I peut être représentée par une suite d'états indicée par I. L'histoire du système dynamique constitue donc un mot sur l'alphabet des états, mot fini, infini ou bi-infini suivant la nature de I. L'ensemble des histoires possibles d'un système dynamique constitue donc un langage sur l'alphabet des états. Dans ce travail, on s'intéressera surtout aux **langages infinitaires**, c'est-à-dire aux langages constitués de mots infinis de même genre.

Parmi les systèmes les plus simples, ceux qui sont modélisés par un automate fini intéressent plus particulièrement les informaticiens. Cela a donné naissance à la notion de **langage reconnaissable**.

La classe des **langages rationnels** finitaires est la plus petite famille de langages de mots finis stable par union finie, concaténation et étoile, et contenant tous les langages finis. La classe des langages reconnaissables finitaires coïncide avec celle des langages rationnels finitaires, c'est le théorème de Kleene (1954).

En ce qui concerne les mots infinis, Büchi (dans une préoccupation qui relevait de la logique) s'est le premier intéressé à leur reconnaissance par des automates finis [12]. D'autres travaux ont suivi, en particulier ceux de Mac Naughton [49], qui établissent que les langages de mots infinis reconnaissables (au sens de Büchi, ou au sens de Muller) sont aussi les langages rationnels: ce sont les unions finies de langages de la forme $B_iC_i^\omega$ où les B_i et les C_i sont des langages rationnels de mots finis. Parmi de nombreux autres travaux, ceux de Landweber sont de nature topologique et situent les langages reconnaissables dans une hiérarchie de familles d'ensembles mesurables [33]. Un aperçu des résultats concernant les langages reconnaissables de mots infinis peut être trouvé dans [52].

La notion de **mot bi-infini** constitue une extension très naturelle de celle de mot infini . M. Nivat et D. Perrin [50], puis D. Beauquier [4] ont proposé des modes de reconnaissance des langages de mots bi-infinis (biautomates, automates de Büchi, automates de Muller). Ces modes de reconnaissance sont équivalents et mènent à la même caractérisation: les langages reconnaissables sont les ensembles rationnels en ce sens que ce sont les unions finies de langages ${}^\omega A_iB_iC_i^\omega$ où les A_i, B_i, C_i sont des langages rationnels de mots finis.

Pour obtenir un mot infini, il suffit de concaténer les éléments d'une suite de mots finis de longueur non nulle; si la concaténation se fait dans l'ordre, nous dirons que la suite de mots finis utilisée constitue une **factorisation** du mot infini obtenu. Si l'on exige que les mots utilisés appartiennent à un même langage G, on a une G-factorisation. Dans notre travail, nous nous intéressons essentiellement au nombre de G-factorisations des mots infinis quand le langage G est un code. Nos résultats seront relatifs aux codes quelconques, mais nous verrons comment s'améliorent les résultats obtenus quand les codes considérés sont rationnels.

Les propriétés des **codes** vis-à-vis des mots infinis ne semblent pas avoir été beaucoup étudiées. En fait, il s'avère seulement que de nombreux auteurs s'intéressant aux mots bi-infinis ([2] [9] [10] [56]) ont surtout pris en considération les **codes circulaires** (définis par J. L. Lassez [34]), mais ceci sans définir une quelconque notion de code ou de factorisation pour les mots bi-infinis. Il se trouve qu'en réalité les codes circulaires, *quand ils sont rationnels*, factorisent de façon unique les mots bi-infinis. On peut d'autre part remarquer que certains codes, introduits dans d'autres buts, se révèlent d'un grand intérêt vis-à-vis des

mots infinis. C'est le cas par exemple des **codes faiblement préfixes** (définis par R. M. Cappocelli [17]) qui, *lorsqu'ils sont rationnels*, assurent l'unicité du décodage des mots infinis à droite. C'est le cas *a fortiori* des codes à délai de déchiffrage borné, introduits par Gilbert et Moore en 1959, en particulier des codes préfixes, et des codes à délai de synchronisation borné, introduits par Golomb et Gordon en 1965, qui assurent en plus un décodage facile [39], [46]. En fait, on verra que certaines notions de code « collent » vraiment aux mots infinis en ce sens qu'elles peuvent être caractérisées en termes de mots infinis. Même la notion usuelle de code peut être caractérisée – et de plusieurs façons – en termes de mots infinis.

Etant donné un alphabet A, tout langage de mots finis G inclus dans A^+, engendre (par concaténation infinie de mots de G) un langage L de mots infinis que nous noterons G^ω s'il s'agit de mots infinis à droite, ${}^\omega G^\omega$ s'il s'agit de mots bi-infinis. Plusieurs sortes de questions se posent alors :
- quelles sont les propriétés des factorisations sur G des mots de L ?
- que dire du nombre de factorisations sur G des mots de L ?
- L possède-t'il des générateurs G intéressants, en particulier selon les points de vue des questions précédentes ?

Cette thèse a pour but d'essayer de répondre aux deux premières questions. Le dernier chapitre est, dans le cadre bi-infinitaire, une ébauche d'étude de la troisième question.

Voyons maintenant ce qu'apporte chacune des parties de ce travail.

L'objet du premier chapitre est de caractériser les codes usuels en termes de mots infinis. Trois points de vue nous intéressent :
- étudier l' ensemble des mots infinis qui se factorisent de façon unique en mots du code ;
- étudier, pour des mots infinis particuliers (par exemple pour les mots périodiques ou ultimement périodiques), la forme des factorisations en mots du code ;
- étudier, pour des mots infinis particuliers, le nombre de leurs factorisations en mots du code.

Cela permet d'obtenir des caractérisations pour les codes. Le premier point de vue ne donnera de résultat que pour les mots infinis d'un seul côté. L'ensemble des mots infinis à droite qui se factorisent de façon unique en mots d'un code C contient tous les mots de la forme u^ω où $u \in C^+$. Cela donne une première caractérisation. Ici réside la différence essentielle entre les mots infinis d'un côté et les mots bi-infinis : il n'y a pas de résultat de ce type pour les mots bi-infinis.

Le second point de vue donne un résultat qui peut se résumer ainsi : toute factorisation sur un code d'un mot infini ultimement périodique est ultimement périodique. Cela donne une seconde caractérisation. Ici encore une différence entre les mots infinis à droite et les mots bi-infinis : les factorisations sur un code d'un mot bi-infini *périodique* sont *périodiques*, celles d'un mot infini à droite *périodique* sont *ultimement périodiques*, et peuvent ne pas être périodiques.

L'étude des codes selon le troisième point de vue montre que tous les mots infinis rationnels (c'est-à-dire les mots ultimement périodiques) n'ont qu'un nombre fini de factorisations sur un code donné. Cela donne la troisième caractérisation des codes.

Ces diverses caractérisations seront largement utilisées dans la suite du travail.

Les deux sections suivantes sont consacrées à la définition et à l'étude de notions de codes pour les mots bi-infinis. Pour le codage des mots infinis, un choix s'avère nécessaire entre d'une part un code finitaire (et par conséquent des factorisations infinies), et d'autre part un code infinitaire (et des factorisations finies ou des factorisations infinies). Certains auteurs ont fait le choix des codes infinitaires. Nous ne ferons pas ce choix, en partie parce que nous voulons que les codes pour mots infinis soient aussi des codes au sens usuel. Une notion de code finitaire pour les mots infinis à droite a été introduite et étudiée par L. Staiger [61] sous le nom de « ifl-codes » : un code pour les mots infinis à droite est un langage de mots finis qui assure l'unicité du décodage des mots infinis à droite. Nous appellerons **ω-codes** ces ifl-codes car ce sont vraiment, à notre avis, les codes pour les mots infinis à droite.

C'est cette même optique que nous prendrons pour définir des notions de codes pour les mots bi-infinis. Une remarque, simple en apparence, nous a fait mettre le doigt sur une particularité des mots bi-infinis périodiques : certains mots bi-infinis périodiques peuvent admettre une seule factorisation, et cependant être décomposés par cette factorisation de plusieurs façons.

En réalité, cette remarque ne s'exprime bien qu'en termes de Z-mots, pour lesquels nous distinguerons factorisations et décompositions. Une factorisation d'un Z-mot w sera une suite de mots finis, indicée par $\boldsymbol{Z}$, qui, concaténée dans l'ordre, donne w. Une décomposition d'un Z-mot w sera une suite strictement croissante d'indices, indicée par $\boldsymbol{Z}$. Une décomposition d'un Z-mot w représente un découpage de w, définissant une factorisation de w. A une factorisation peut correspondre plusieurs décompositions. Ce fait peut être illustré par l'exemple – très simple – suivant : à la factorisation $(aa)_{n\in Z}$ de a^Z correspond deux décompositions :

$$\ldots a_1a_2a_3a_4a_5a_6\ldots = \ldots a_1a_2.a_3a_4.a_5a_6\ldots = \ldots a_1.a_2a_3.a_4a_5.a_6\ldots$$

(où $a_i = a$ pour tout i).

Ceci va nous mener à définir deux notions de codes pour les mots bi-infinis : celle de **biω-code**, relative aux mots bi-infinis proprement dits, qui assurera l'unicité de la factorisation d'un mot bi-infini en mots du code, et celle de **Z-code**, relative aux Z-mots, qui assurera l'unicité de la décomposition d'un Z-mot en mots du code.

Dans le deuxième chapitre, nous étudions les **codes circulaires** et les **codes précirculaires**. Nous démontrons que les codes qui assurent l'unicité de la décomposition des mots périodiques sont les codes circulaires. Par conséquent, les Z-codes sont des codes circulaires.

Nous définissons et étudions une classe de codes un peu plus large que celle des codes circulaires : celle des codes précirculaires. Leur définition, en termes de mots finis, exige que les mots (finis) écrits sur un cercle ait au plus une factorisation (celle des codes circulaires exige que les mots écrits sur un cercle ait au plus un découpage ; un mot écrit sur un cercle peut avoir une seule factorisation mais plusieurs découpages). Nous obtenons plusieurs caractérisations des codes précirculaires, en particulier une caractérisation en termes de morphisme de codage et une caractérisation par rapport aux **codes presque circulaires** introduits par M. Leconte [36], [37]. Mais la principale propriété des codes précirculaires, et c'est la raison pour laquelle nous les avons introduits, est la caractérisation très intéressante suivante : les codes qui assurent l'unicité de la factorisation des mots périodiques sont les codes précirculaires.

Dans le troisième chapitre, nous définissons et caractérisons les ω-codes, les biω-codes et les Z-codes. Le résultat intéressant sur les Z-codes dit que ceux-ci sont les biω-codes purs, de même que les codes circulaires étaient les codes précirculaires purs.

Les biω-codes sont, comme les codes usuels et les ω-codes, caractérisés par une propriété d'injectivité des morphismes de codage. Cela nous mène à penser que la notion de biω-code est la bonne notion de code pour les mots bi- infinis. Les biω-codes sont à la fois des codes précirculaires et des ω-codes ; mais les ω-codes précirculaires ne sont pas toujours des biω-codes. Nous donnons des conditions nécessaires et suffisantes pour qu'un code précirculaire soit un biω-code.

Les trois points de vue précédemment définis pour étudier les codes usuels permettent, non seulement d'obtenir des caractérisations pour les codes, mais aussi de caractériser d'autres familles de codes, souvent déjà définies et étudiées sous d'autres points de vue. C'est le cas par exemple des codes **faiblement préfixes**, des codes **suffixes** et des codes **circulaires**.

L'objet du quatrième chapitre est de donner ces caractérisations et de définir en termes de mots finis d'autres familles de codes qui ont de belles propriétés

vis-à-vis des mots infinis.

Nous nous intéressons d'abord aux langages permettant de factoriser (ou de décomposer) de façon unique certains types de mots infinis. Si l'on exige l'unicité de la factorisation des mots infinis à droite ultimement périodiques, on obtient la famille déjà définie des **codes faiblement préfixes**. Si on exige l'unicité de la factorisation des mots infinis à droite périodiques, on obtient la famille des ***Π*-codes**. Si on exige l'unicité de la factorisation des mots bi-infinis ultimement périodiques, on obtient la famille des codes **fortement précirculaires**. Ces notions sont caractérisées en termes de mots finis.

Si l'on s'intéresse à la forme des factorisations, on peut se poser la question suivante : « quels sont les langages tels que toute factorisation d'un mot infini à droite périodique soit périodique? ». La réponse est un joli résultat inattendu : ces langages sont les **codes suffixes**.

En conclusion de la quatrième partie, des tableaux résument les résultats quantitatifs. Ils donnent, en fonction de la nature du code considéré, le nombre maximal de factorisations (ou de décompositions dans le cas des Z-mots) d'un mot de nature donnée (mot périodique, mot ultimement périodique ou mot quelconque). Des exemples permettent de voir qu'un mot peut avoir une infinité non dénombrable de factorisations, même si l'on factorise sur un code intéressant (un code faiblement préfixe ou un code fortement précirculaire par exemple).

Pour terminer cette partie, un organigramme présente les relations existant entre les différentes classes de codes considérées.

Le cinquième chapitre est consacré à l'étude du **cas rationnel**. On a vu qu'on peut trouver des codes C et des mots infinis qui ont une infinité non dénombrable de C-factorisations. Ce n'est plus possible si l'on s'intéresse aux codes rationnels. Mieux encore : tout mot infini a un nombre fini de factorisations sur un code rationnel.

Dans le cas général, nous n'avions pas de caractérisation en termes de mots finis pour les notions de code pour les mots infinis (ω-code, biω-code, Z-code). Dans le cas rationnel, nous obtenons de telles caractérisations. En effet, l'ensemble des mots ayant plusieurs factorisations est alors rationnel ; cela permet de démontrer que les codes rationnels faiblement préfixes sont des ω-codes, les codes rationnels fortement précirculaires des biω-codes, les codes circulaires rationnels des Z-codes. Ces résultats sont en accord avec le fait que l'ensemble des mots ultimement périodiques d'un langage rationnel spécifie complètement ce langage [12].

Nous démontrons la décidabilité de l'appartenance d'un langage rationnel à l'une quelconque des classes de codes étudiées.

Pour terminer nous donnons une caractérisation des biω-codes dans le cas rationnel.

Le sixième chapitre de ce travail est consacré à l'étude des **codes à délai de déchiffrage fini.**

Il est intéressant de coder les mots infinis par un ω-code puisque le décodage est alors unique. Il est plus intéressant encore que ce décodage puisse se réaliser concrètement. En d'autres termes, peut-on être assuré qu'un mot infini, codé à l'aide d'un code C, peut être décodé à la lecture (à partir du début du mot et vers la droite), avec retard éventuellement. Pour cela, il vaut mieux que C soit à délai de déchiffrage fini (ou mieux: borné). En effet, la factorisation du début d'un mot infini en $(m+1)$ mots d'un code à délai m nous assure que le premier mot de cette factorisation est bon.

Le cas rationnel est étudié plus particulièrement. Nous présentons des conditions nécessaires et suffisantes pour qu'un ω-code rationnel soit à délai borné. L'une de ces conditions est nouvelle et semble réellement intéressante. De cette condition, on peut déduire qu'un langage C^ω engendré par un code à délai fini C (même non rationnel) est un langage de type G_δ. En particulier, si C est un code à délai fini rationnel, C^ω est un langage rationnel déterministe.

De même, il est intéressant de coder les mots bi-infinis par un biω-code puisque le décodage est alors unique. Si l'on considère un mot bi-infini (ou un mot quelconque dont on ne possède qu'un facteur) codé et que l'on veuille le décoder à la lecture (à partir d'un certain point, vers la droite ou (et) vers la gauche), il vaut mieux que le code soit à délai de synchronisation borné.

Le septième chapitre de ce travail est consacré à l'étude des **codes à délai de synchronisation borné.** De tels codes sont circulaires. Des conditions nécessaires et suffisantes pour qu'un code circulaire rationnel ait un délai de synchronisation borné sont rappelées ou démontrées. L'une d'entre elles est nouvelle et permet d'obtenir des propriétés de ${}^\omega C^\omega$ quand C a un délai de synchronisation borné. Par exemple, ${}^\omega C^\omega$ est une bilimite. De plus, quand l'alphabet est fini, ${}^\omega C^\omega$ ne peut être une biadhérence que si C est fini.

L'objet du huitième chapitre est de rappeler les résultats obtenus dans le cas des **codes finis**, et d'étudier ce qu'apporte l'hypothèse supplémentaire que le code a trois ou deux éléments. Pour chaque famille étudiée, nous caractérisons les **codes à deux éléments** qui en font partie.

Quelques compléments sont présentés dans le neuvième chapitre.

Nous étudions d'abord le type des codes obtenus par composition de codes appartenant à l'une des familles étudiées.

Ensuite, nous rappelons ou démontrons des résultats concernant le plongement d'un code donné d'une certaine famille dans un code maximal dans cette famille, ou dans un code de cette famille, maximal en tant que code.

Le dixième chapitre est une ouverture sur l'étude des **générateurs** d'un langage bi-infinitaire L donné. Le mot « générateur » signifie ici « langage $G \subset A^*$ tel que ${}^\omega G^\omega = L$ ». Peu de résultats existent dans ce domaine. Nous rappelons d'abord que l'on sait décider si un langage bi-infinitaire rationnel L donné a des générateurs. Si c'est le cas, L a des générateurs maximaux, et ceux-ci sont rationnels, constructibles et en nombre fini.

Nous nous intéressons ensuite aux générateurs minimaux.

Les mots périodiques se factorisent de façon unique sur un code précirculaire. A l'aide de cette propriété, nous montrons que les codes précirculaires sont des générateurs minimaux.

Nous ne savons pas si tous les codes sont des générateurs minimaux. Mais nous prouvons que tous les codes rationnels sont des générateurs minimaux.

Afin d'approfondir le lien avec les générateurs des langages infinitaires, nous définissons la notion de **bissection générative**. Une bissection générative est un couple $({}^\omega G, G^\omega)$ où G est un générateur de L. Mais, pour l'instant, cette notion ne paraît pas débloquer les problèmes. C'est le cas, en particulier, dans la recherche de générateurs finis.

Pour rechercher des générateurs finis, nous sommes donc amenés à travailler directement sur le langage bi-infinitaire donné. Finalement, nous montrons que l'on peut décider si un langage bi-infinitaire rationnel L donné a des générateurs finis.

Cette partie comporte donc deux résultats intéressants : le fait que les codes rationnels sont des générateurs minimaux, et la décidabilité de l'existence de générateurs finis. Par ailleurs, le travail réalisé montre à quel point le cas bi-infinitaire se révèle très différent du cas infinitaire.

Une bonne partie du contenu de cette thèse a déjà été écrite ailleurs et se retrouve publiée [23], [24], ou à paraître [21], [20], ou fait l'objet de publications internes [22], [19].

Préliminaires

0.1 Définitions et notations

Précisons les notations utilisées dans ce travail. Si $\boldsymbol{A}$ désigne un alphabet dénombrable (fini ou non), $\boldsymbol{A^*}$ représente l'ensemble des **mots finis** sur A, $\boldsymbol{\varepsilon}$ désigne le mot vide et $\boldsymbol{A^+}$ le langage $A^* - \varepsilon$. La longueur d'un mot u est $\boldsymbol{|u|}$. Le symbole $\leq$ (resp. $<$) représente la relation « être un facteur gauche (resp. facteur gauche strict) de ». Deux mots x et x' sont dits **conjugués** s'il existe u et v tels que $x = uv$ et $x' = vu$. Un mot $z \in A^+$ est **primitif** si $z = u^n$ seulement pour $n = 1$, **imprimitif** dans le cas contraire. Tout mot $z \in A^+$ possède une unique racine notée $\sqrt{z}$ qui est le mot primitif vérifiant $\sqrt{z}^{\,n} = z$.

L'ensemble des entiers relatifs (resp. naturels ; resp. naturels non nuls) est noté $\boldsymbol{Z}$ (resp. $\boldsymbol{N}$; resp. $\boldsymbol{N^*}$). $\boldsymbol{A^Z}$ (resp. $\boldsymbol{A^\omega}$; resp. $\boldsymbol{{}^\omega A}$) est l'ensemble des **Z-mots** (resp. **mots infinis (à droite)** ; resp. mots infinis à gauche), c'est-à-dire des suites de lettres de A indicées par Z (resp. N ; resp. $-N$). Pour un Z-mot $w = (w_n)_{n \in Z}$ on notera $w[p, (p+h)[$ le facteur $w_p w_{p+1} ... w_{p+h-1}$, w_+ le mot infini $(w_n)_{n \in N}$ et w_- le mot infini à gauche $(w_n)_{n \in -N^*}$. Les **mots bi-infinis** [4] [50] sur A sont les classes d'équivalence pour la relation sur A^Z :

$$(w_n)_{n \in Z} \sim (v_n)_{n \in Z} \Leftrightarrow \exists p \in Z \; \forall n \in Z \; w_n = v_{n+p} \cdot$$

Le mot bi-infini défini par $(w_n)_{n \in Z}$ sera noté $...w_{-1}w_0w_1...$. L'ensemble des mots bi-infinis sur A est noté $\boldsymbol{{}^\omega A^\omega}$. Nous noterons $\boldsymbol{\pi}$ la projection canonique de A^Z sur ${}^\omega A^\omega$ et $\boldsymbol{\sigma}$ la fonction de décalage de A^Z sur lui-même : $\sigma(w_n)_{n \in Z} = (w_{n+1})_{n \in Z}$.

Etant donné un langage $C \subset A^+$, le sous-monoïde engendré par C est le langage obtenu par concaténation finie de mots de C : $\boldsymbol{C^*} = \{v_1 ... v_n \mid n \geq 0 \;\; v_i \in C \text{ pour } 1 \leq i \leq n\}$ et $\boldsymbol{C^\omega}$ (resp. $\boldsymbol{{}^\omega C^\omega}$) est l'ensemble des mots infinis (resp. bi-infinis) obtenus par concaténation d'une infinité de mots de C :
$C^\omega = \{v_0 v_1 v_2 ... \mid v_i \in C \text{ pour } i \geq 0\}, {}^\omega C^\omega = \{...v_{-1} v_0 v_1 v_2 ... \mid v_i \in C, i \in Z\}$.

Un mot infini w est dit **ultimement périodique** s'il existe des mots finis u et v tels que $w = uv^\omega$. Il est **périodique** si u peut être choisi égal à ε. Un mot bi-infini w est dit **ultimement périodique à droite** s'il existe un mot

infini à gauche u et un mot fini v tels que $w = uv^\omega$. Il est **périodique** s'il peut s'écrire $w = {}^\omega v^\omega$ où v est un mot fini. Un Z-mot est **ultimement périodique à droite** (resp. **périodique**) si le mot bi-infini correspondant est ultimement périodique à droite (resp. périodique). On définit de façon analogue les mots ultimement périodiques à gauche. Un Z-mot (resp. mot bi-infini) **ultimement périodique** est un Z-mot (resp. mot bi-infini) qui est à la fois ultimement périodique à droite et à gauche. Le qualificatif « rationnel » sera parfois utilisé à la place de « ultimement périodique » du fait que, parmi les singletons, seuls ceux formés d'un mot ultimement périodique sont rationnels [50].

Etant donnés deux langages U et V ($U \subset A^*$ et $V \subset A^*$, $U \subset {}^\omega A$ et $V \subset {}^\omega A$ ou $V \subset A^*$ et $U \subset A^\omega$), le **quotient à gauche** de U par V: $\{w \in A^* \mid \exists u \in U\ \exists v \in V\ u = vw\}$ est noté $V^{-1}U$. On définit de même le quotient à droite.

0.2 Factorisations et décompositions

Une **C-factorisation** d'un mot $v \in C^*$ est une suite finie de mots de C: $(v_1, \ldots, v_n)$ telle que $v = v_1 \ldots v_n$. Une **C-factorisation** d'un mot $v \in C^\omega$ est une suite infinie de mots de C: $(v_0, v_1, v_2, \ldots)$ telle que $v = v_0 v_1 v_2 \ldots$. Une **C-factorisation** d'un mot $v \in {}^\omega C^\omega$ est une classe d'équivalence, pour la relation de décalage définie ci-dessous, d'une suite $(v_i)_{i \in Z}$ de mots de C, vérifiant $v = \ldots v_{-1} v_0 v_1 v_2 \ldots$. La relation de décalage est définie sur C^Z par:

$$(v_i)_{i \in Z} \sim (u_i)_{i \in Z} \Leftrightarrow \exists p \in Z\ \forall i \in Z\ v_i = u_{i+p}\ .$$

Une **C-factorisation** d'un Z-mot v est une C-factorisation de la classe de v dans ${}^\omega A^\omega$.

Etant donné un langage $C \subset A^+$, un **morphisme de codage pour C** est un morphisme $\varphi : X^* \to C^*$ (même non injectif), extension à X^* d'une bijection φ entre un alphabet X et C. On peut aussi considérer les extensions: $\varphi : X^\omega \to C^\omega$ définie par $\varphi(z_0 z_1 \ldots) = \varphi(z_0)\varphi(z_1)\ldots$ et $\varphi : {}^\omega X^\omega \to {}^\omega C^\omega$ définie par $\varphi(\ldots z_{-1} z_0 z_1 \ldots) = \ldots \varphi(z_{-1})\varphi(z_0)\varphi(z_1)\ldots$. Ces extensions sont compatibles avec la composition des fonctions. Il existe une bijection naturelle entre X^* (resp. X^ω, resp. ${}^\omega X^\omega$) et l'ensemble des C-factorisations de mots de C^* (resp. C^ω, resp. ${}^\omega C^\omega$). On représentera donc une C-factorisation d'un mot de C^* (resp. C^ω, resp ${}^\omega C^\omega$) par un élément de X^* (resp. X^ω, resp. ${}^\omega X^\omega$).

Une C-factorisation représentée par un mot $u \in X^\omega$ ou $u \in {}^\omega X^\omega$ sera dite **périodique** (resp. **ultimement périodique**) si u l'est.

Une **C-décomposition d'un Z-mot** $u = (u_i)_{i \in Z}$ est une suite strictement croissante d'entiers relatifs $(d_n)_{n \in Z}$ telle que $d_0 \geq 0, d_{-1} < 0$ et pour tout $k \in Z$ $u[d_k, d_{k+1}[\in C$.

Soit u un Z-mot et w la classe de u dans ${}^\omega A^\omega$. On appelle τ_u l'application qui à une C-décomposition $(d_n)_{n \in Z}$ de u fait correspondre la C-factorisation

de w de représentant $(u[d_n, d_{n+1}[)_{n\in Z}$. Par définition, il est clair qu'une C-factorisation de w provient d'une C-décomposition de u, donc τ_u est surjective. Par contre, τ_u n'est pas toujours injective. Par exemple, si $C = \{aa\}$, le mot ${}^\omega a^\omega$ a une seule C-factorisation : ${}^\omega a^\omega = {}^\omega(aa)^\omega$. Mais à cette C-factorisation correspond deux C-décompositions de a^Z :

$$...a_1a_2a_3a_4a_5a_6... = ...a_1a_2.a_3a_4.a_5a_6... = ...a_1.a_2a_3.a_4a_5.a_6...$$

(où $a_i = a$ pour tout i).

Si c désigne l'unique C-factorisation de ${}^\omega a^\omega$, $\tau_{a^Z}^{-1}(c)$ a deux éléments: $(2n)_{n\in Z}$ et $(2n+1)_{n\in Z}$.

On dira qu'une C-factorisation c d'un mot bi-infini w est **chevauchante** si il existe un représentant u de w pour lequel $\tau_u^{-1}(c)$ a plus d'un élément.

Remarquons que les propriétés des factorisations: « être périodique » et « être chevauchante » sont incomparables. Par exemple, si φ désigne un morphisme de codage, les factorisations notées: ${}^\omega x^\omega$ et ${}^\omega(xy)^\omega$ où $\varphi(x) = ab$ et $\varphi(y) = a$ sont périodiques et non chevauchantes. Les factorisations notées: ${}^\omega x^\omega$ et ${}^\omega(zy)^\omega$ où $\varphi(x) = aa$, $\varphi(z) = aba$ et $\varphi(y) = b$ sont périodiques et chevauchantes.

La factorisation non périodique de ${}^\omega a^\omega$ de représentant: $(a^{|n|+1})_{n\in Z}$ est chevauchante .

On a les résultats suivants:

Lemme 0.2.1 - *Soient u et u' deux Z-mots représentant le même mot bi-infini w, et c une C-factorisation de w. Il y a une bijection canonique entre $\tau_u^{-1}(c)$ et $\tau_{u'}^{-1}(c)$.*
- Si le Z-mot u n'est pas périodique, l'application τ_u est bijective.

Preuve
- Si u et u' sont deux Z-mots représentant w, fixons p tel que $u'_i = u_{i+p}$ pour tout i. Soit d une C-décomposition de u. Il existe un unique k tel que $d_{k-1} < p \leq d_k$. Posons $d'_n = d_{n+k} - p$ pour tout n de $\boldsymbol{Z}$. d' est une C-décomposition de u', de plus, $\tau_u(d) = \tau_{u'}(d')$. Si l'on pose $f(d) = d'$, f est une bijection.
- Soit c une C-factorisation d'un mot bi-infini w de représentant u. Si $\tau_u^{-1}(c)$ a au moins deux éléments d et d', il existe q tel que $u[d_k, d_{k+1}[= u[d'_{q+k}, d'_{q+k+1}[$ pour tout k. On a alors $u_{d_0}u_{d_0+1}u_{d_0+2}... = u_{d'_q}u_{d'_q+1}u_{d'_q+2}...$ et $...u_{d_0-3}u_{d_0-2}u_{d_0-1} = ...u_{d'_q-3}u_{d'_q-2}u_{d'_q-1}$. Si $d_0 < d'_q$, u est périodique de période $u[d_0, d'_q[$. Si $d_0 > d'_q$, u est périodique de période $u[d'_q, d_0[$. On ne peut avoir $d_0 = d'_q$, car alors $q = 0$ et $d = d'$. •

0.3 Langages rationnels, langages reconnaissables

Dans cette partie, A est un alphabet fini, tous les langages sont constitués de mots sur l'alphabet A, tous les automates sont définis sur l'alphabet A.

0.3.1 Reconnaissance

En ce qui concerne les langages infinitaires, nous nous limitons ici à la reconnaissance au sens de Büchi. Pour une étude détaillée des divers modes de reconnaissance, on peut se référer pour les mots infinis à [52] et pour les mots bi-infinis à [50] et [4].

Un **automate fini** est un quadruplet (Q, I, F, R) où Q (l'ensemble des états) est un ensemble fini , I (l'ensemble des états initiaux) et F (l'ensemble des états finaux) sont des parties de Q, R (qui caractérise les transitions) est une application de $Q \times A$ dans 2^Q. L'application R se prolonge à $Q \times A^*$ de la façon suivante : pour tout $q \in Q$, $R(q, \varepsilon) = q$, $R(q, au) = \cup_{q' \in R(q,a)} R(q', u)$ si $a \in A$ et $u \in A^*$.
L'automate est dit **déterministe** si R est à valeurs dans $\{\emptyset\} \cup \{\{q\} | q \in Q\}$.

Un mot fini $u = u_1 u_2 ... u_p$ est reconnu par l'automate (Q, I, F, R) si et seulement si il existe une suite d'états $(q_n)_{1 \leq n \leq p+1}$ tels que: $q_1 \in I$, $\forall n \in \{1, ..., p\}$ $q_{n+1} \in R(q_n, u_n)$ et $q_{p+1} \in F$.
On dira alors que la suite $(q_n, u_n, q_{n+1})_{1 \leq n \leq p}$ est une lecture réussie de u.
Un langage $L \subset A^*$ est dit **reconnaissable** s'il est l'ensemble des mots finis reconnus par un automate fini.

Un mot infini $u = u_0 u_1 ...$ est reconnu (au sens de Büchi) par l'automate (Q, I, F, R) si et seulement si il existe une suite d'états $(q_n)_{n \in N}$ tels que : $q_0 \in I$, $\forall n \in N$ $q_{n+1} \in R(q_n, u_n)$ et $Card\{n \geq 0 \mid q_n \in F\}$ est infini.
On dira alors que la suite $(q_n, u_n, q_{n+1})_{n \in N}$ est une lecture réussie de u .
Un langage $L \subset A^\omega$ est dit **reconnaissable** s'il est l'ensemble des mots infinis reconnus (au sens de Büchi) par un automate fini.

Un mot bi-infini u est reconnu (au sens de Büchi) par l'automate (Q, I, F, R) si et seulement si il existe un représentant $...u_0 u_1 ...$ de u et une suite d'états $(q_n)_{n \in Z}$ tels que : $\forall n \in Z$ $q_{n+1} \in R(q_n, u_n)$, $Card\{n \leq 0 \mid q_n \in I\}$ et $Card\{n \geq 0 \mid q_n \in F\}$ sont infinis.
On dira alors que la suite $(q_n, u_n, q_{n+1})_{n \in Z}$ est une lecture réussie de u .
Un langage $L \subset {}^\omega A^\omega$ est dit **reconnaissable** s'il est l'ensemble des mots bi-infinis reconnus (au sens de Büchi) par un automate fini.

0.3.2 Rationalité

Les parties **rationnelles** de A^* sont les éléments de la plus petite famille de sous-ensembles de A^* close par union finie, concaténation finie et étoile, et contenant tous les sous-ensembles finis de A^*.

Le théorème de Kleene dit que, sur un alphabet fini A, l'ensemble des langages inclus dans A^* et reconnaissables coïncide avec l'ensemble des langages rationnels inclus dans A^*.

Les parties **rationnelles** de A^ω sont les unions finies de langages de la forme BC^ω où B et C sont des parties rationnelles non vides de A^+.

Les théorèmes de Büchi et de Mac Naughton établissent que l'ensemble des langages inclus dans A^ω et reconnaissables coïncide avec l'ensemble des langages rationnels inclus dans A^ω, et que c'est la clôture booléenne (dans A^ω) de l'ensemble des langages reconnus (au sens de Büchi) par des automates finis déterministes.

Les parties **rationnelles** de ${}^\omega A^\omega$ sont les unions finies de langages de la forme ${}^\omega DBC^\omega$ où D, B et C sont des parties rationnelles non vides de A^+.

Les travaux de M. Nivat, D. Perrin et D. Beauquier montrent qu'ici encore l'ensemble des langages inclus dans ${}^\omega A^\omega$ et reconnaissables coïncide avec l'ensemble des langages rationnels inclus dans ${}^\omega A^\omega$. Il est fermé pour l'union finie, l'intersection finie et le passage au complémentaire.

0.3.3 Automates *ad hoc*

Soit C inclus dans A^+ un langage rationnel, considérons un automate fini déterministe reconnaissant C. A partir de cet automate, on peut construire un automate fini reconnaissant C, $\Omega_0 = (Q_0, q_0, q_F, R_0)$, ayant un seul état initial q_0, un seul état final q_F, tel qu'aucune flèche ne parte de q_F et aucune flèche ne rentre dans q_0. On peut faire en sorte que cet automate Ω_0 soit émondé (c'est-à-dire que, pour tout état q, il existe un chemin menant de q_0 à q et un chemin menant de q à q_F) et non ambigu (c'est-à-dire que les mots de C ont une seule lecture réussie). Un automate Ω_0 qui a ces propriétés sera appelé dans la suite **automate** *ad hoc* pour C. A partir d'un automate *ad hoc* pour C, on peut considérer l'automate $\Omega = (Q, q_0, q_0, R)$ obtenu par identification de q_0 et q_F. Cet automate sera désigné par la suite comme **automate dérivé de** Ω_0. Le langage de mots finis reconnu par cet automate Ω est C^* ; le langage de mots infinis reconnu (au sens de Büchi) par l'automate Ω est C^ω ; le langage de mots bi-infinis reconnu (au sens de Büchi) par l'automate Ω est ${}^\omega C^\omega$.

Si C est un code, l'automate Ω est non ambigu pour la reconnaissance des mots de C^* [8].

Proposition 0.3.1 *Pour tout mot w de A^ω, il y a une bijection entre l'en-*

semble des lectures réussies de w sur Ω et l'ensemble des C- factorisations de w. Pour tout mot w de A^Z, il y a une bijection entre l'ensemble des lectures réussies de w sur Ω et l'ensemble des C-décompositions de w.

Preuve : La bijection f peut être définie de la façon suivante :
- Pour w appartenant à A^ω et $l = (q_{j(i)}, w_i, q_{j(i+1)})_{i\geq 0}$ une lecture réussie de w sur Ω, il suffit de définir $f(l)$ comme la suite des mots de C définis par les passages successifs par q_0 : $f(l) = (w[h(i), h(i+1)[)_{i\geq 0}$ où h est la suite strictement croissante d'entiers naturels vérifiant $h(N) = \{i \in N \mid q_{j(i)} = q_0\}$.
- Pour w appartenant à A^Z, et $l = (q_{j(i)}, w_i, q_{j(i+1)})_{i\in Z}$ une lecture réussie de w sur Ω, il suffit de définir $f(l)$ comme la suite strictement croissante d'entiers relatifs d satisfaisant à $d_0 \geq 0$, $d_{-1} < 0$ et $d(Z) = \{i \in Z \mid q_{j(i)} = q_0\}$. •

0.4 Langages infinitaires

0.4.1 Langages infinitaires particuliers

On note :
- $\boldsymbol{F(G)}$ l'ensemble des facteurs finis de G, si G est un langage inclus dans A^+, A^ω, ${}^\omega A$ ou ${}^\omega A^\omega$,
- $\boldsymbol{Fg(G)}$ l'ensemble des facteurs gauches finis de G, si G est inclus dans A^+ ou A^ω,
- $\boldsymbol{Fd(G)}$ l'ensemble des facteurs droits finis de G, si G est inclus dans A^+ ou ${}^\omega A$.

La **limite** d'un langage G ($\subset A^*$) : $\boldsymbol{Lim(G)}$ est l'ensemble des mots infinis ayant une infinité de facteurs gauches dans G. Landweber a démontré que les langages limites sont exactement les intersections dénombrables de langages de la forme WA^ω où $W \subset A^*$ [33].

L'**adhérence** d'un langage G ($\subset A^*$) : $\boldsymbol{Adh(G)}$ est l'ensemble des mots infinis w tels que tout facteur gauche de w soit facteur gauche d'un mot de G : $Adh(G) = \{w \in A^\omega \mid Fg(w) \subset Fg(G)\}$.

La **topologie** usuelle sur A^ω est la topologie produit des topologies discrètes sur chaque facteur A. A^ω est ainsi un espace métrique (compact si A est fini). On peut choisir comme distance la fonction d suivante :
$d(w, w') = 2^{-inf\{n \mid w_n \neq w'_n\}}$ si $w \neq w'$.

Les ouverts sont les sous-ensembles de la forme $E = WA^\omega$ où $W \subset A^*$, et les fermés sont les adhérences [11]. Nous utiliserons aussi les classes suivantes de la hiérarchie borélienne. Un F_σ-langage est une union dénombrable de fermés et un G_δ-langage une intersection dénombrable d'ouverts. Les langages limites sont les G_δ-langages [33]. Rappelons que, dans un espace métrique, les fermés sont des G_δ-langages.

Les langages limites et rationnels sont les limites des langages rationnels finitaires ; ce sont aussi les langages rationnels déterministes.

Les adhérences rationnelles sont les adhérences des langages rationnels finitaires [11].

On peut trouver les propriétés de ces langages infinitaires particuliers dans [52],[11], par exemple.

De même qu'on a défini la limite $Lim(G)$ et l'adhérence $Adh(G)$ d'un langage G inclus dans A^*, on en définit la **limite gauche** et l'**adhérence gauche** :
- $\boldsymbol{Llim(G)}$ est l'ensemble des mots infinis à gauche ayant une infinité de facteurs droits dans G,
- $\boldsymbol{Ladh(G)}$ est l'ensemble des mots infinis à gauche dont tous les facteurs droits sont facteurs droits de G: $Ladh(G) = \{w \in {}^{\omega}A \mid Fd(w) \subset Fd(G)\}$.

0.4.2 Langages bi-infinitaires particuliers

Nous dirons qu'une suite de mots finis $(u_n)_{n\geq 0}$ est **strictement croissante** si :

$$\forall n \geq 0 \ \exists v_n, v'_n \in A^+ \ u_{n+1} = v_n u_n v'_n \quad [4].$$

Un mot w de A^Z est une bilimite de la suite strictement croissante $(u_n)_{n\geq 0}$ si il existe une suite strictement croissante $(p_n)_{n\geq 0}$ et une suite strictement décroissante $(q_n)_{n\geq 0}$ d'entiers relatifs telles que, pour tout n, $u_n = w[q_n, p_n[$. Un mot bi-infini w est une **bilimite** de la suite strictement croissante $(u_n)_{n\geq 0}$ s'il a un représentant qui est bilimite de cette suite, ou encore si tous ses représentants sont bilimites de cette suite.

La **bilimite** dans A^Z d'un langage finitaire G: $\boldsymbol{Bilim(G)}$ est l'ensemble des Z-mots bilimites des suites strictement croissantes de mots de G. Sa bilimite dans ${}^{\omega}A^{\omega}$ (notée de la même façon) est l'ensemble des mots bi-infinis bilimites des suites strictement croissantes de mots de G, c'est aussi le quotient de la bilimite dans A^Z par la relation de décalage [4].

La **biadhérence** d'un langage finitaire G est la bilimite de $F(G)$:
$\boldsymbol{Biadh(G)}$= $\{w \in {}^{\omega}A^{\omega} \mid F(w) \subset F(G)\}$ [31]. L'ensemble des Z-mots représentant les mots bi-infinis de $Biadh(G)$ sera appelé biadhérence de G dans A^Z et noté aussi $Biadh(G)$.

Nous utiliserons certains résultats sur les bilimites et sur les biadhérences. Ces résultats, de D. Beauquier [4], de I. Litovsky [42] ou de J. Devolder et I. Litovsky [23], seront rappelés au moment de leur utilisation. Rappelons cependant certains faits principaux :
- un langage L est une biadhérence si et seulement si c'est la biadhérence de $F(L)$,
- un langage de la forme ${}^{\omega}G^{\omega}$ est une biadhérence si et seulement si c'est la

biadhérence de G^*,
- quand l'alphabet est fini, $Biadh(G)$ est vide si et seulement si G est fini,
- si G est fini, ${}^\omega G^\omega$ est la biadhérence de G^*,
- les biadhérences rationnelles sont les biadhérences des langages rationnels finitaires,
-les bilimites des langages rationnels finitaires sont des rationnels déterministes particuliers: ce sont ceux qui sont reconnus par un biautomate déterministe [4].

Nous utiliserons aussi les sous-ensembles de ${}^\omega A^\omega$ de la forme:
$K(G) = Ladh(G).G^*.Adh(G) \bigcup {}^\omega G.Adh(G) \bigcup Ladh(G).G^\omega \bigcup Biadh(G)$, où G est un sous-ensemble de A^+. Le langage $K(G)$ intervient dans le calcul de $Biadh(G^*)$: $Biadh(G^*) = {}^\omega G^\omega \cup K(G)$

La **topologie** usuelle sur A^Z est la topologie produit des topologies discrètes sur chaque facteur A. A^Z est ainsi un espace métrique (compact si A est fini). On peut choisir comme distance la fonction d suivante:
$d(w,w') = 2^{-inf(inf\{n\geq 0 | w_n \neq w'_n\}, -sup\{n<0 | w_n \neq w'_n\})}$ si $w \neq w'$.

Notons N^* l'ensemble des entiers strictement positifs. Considérons l'application ρ de A^Z dans $A^{N^*} \times A^N$ qui à $w = (w_i)_{i\in Z}$ associe $(\tilde{w_-}, w_+)$ où $\tilde{w_-} = (w_{-i})_{i\in N^*}$ et $w_+ = (w_i)_{i\in N}$. De cette façon, on peut considérer la topologie de A^Z comme la topologie produit des topologies usuelles sur A^{N^*} et A^N.

Etant donné des sous-ensembles G_1 et G_2 de A^{N^*} et A^N, nous noterons $G_1 \# G_2$ le sous-ensemble de A^Z: $\rho^{-1}(G_1 \times G_2)$.

Nous utiliserons les F_σ-langages et les G_δ-langages de A^Z ainsi que les fermés suivants:

Proposition 0.4.1 *Si G est un langage de A^*, l'adhérence de G dans A^Z est un fermé de A^Z σ-invariant. Si F_1 est un fermé de A^{N^*} et F_2 un fermé de A^N, $F_1 \# F_2$ est un fermé de A^Z.*

On peut munir ${}^\omega A^\omega$ de la **topologie quotient**: $U \subset {}^\omega A^\omega$ est un ouvert si et seulement si $\pi^{-1}(U)$ est un ouvert de A^Z. La projection canonique π de A^Z sur ${}^\omega A^\omega$ est alors ouverte et continue. Mais il existe des singletons non fermés. Par exemple, si u est un mot bi-infini tel que $F(u) = A^*$, la fermeture de $\{u\}$ est ${}^\omega A^\omega$ et donc:
- l'espace ${}^\omega A^\omega$ n'est pas séparé,
- l'image par π d'un fermé de A^Z n'est pas toujours fermée,
- l'ensemble $F(u)$ des facteurs d'un mot u ne caractérise pas toujours u.

Les fermés de ${}^\omega A^\omega$ sont les biadhérences. Les ouverts sont les sous-ensembles de la forme ${}^\omega AWA^\omega$ où $W \subset A^*$.

Il s'ensuit que les seuls ensembles à la fois ouverts et fermés sont $\emptyset$ et ${}^{\omega}A^{\omega}$, et donc l'espace ${}^{\omega}A^{\omega}$ est connexe.

Les fermés (resp. ouverts) σ-invariants de A^Z sont les images réciproques par π des biadhérences (resp. des ouverts de ${}^{\omega}A^{\omega}$).

Nous donnerons au chapitre 7 un exemple d'une bilimite d'un langage rationnel qui est un G_δ-langage de A^Z et dont l'image par π n'est pas un G_δ-langage de ${}^{\omega}A^{\omega}$. Cela montre qu'il n'y a pas coïncidence entre la classe des bilimites de ${}^{\omega}A^{\omega}$ et celle des G_δ-langages de ${}^{\omega}A^{\omega}$, même dans le cas rationnel.

Dans le cas infinitaire, Landweber a montré que tous les langages rationnels sont à la fois des intersections dénombrables de F_σ-langages et des unions dénombrables de G_δ-langages. Dans le cas bi-infinitaire, la situation est très différente : tout langage (rationnel ou non) contenant un mot w tel que $F(w) = A^*$ et faisant partie de la hiérarchie borélienne définie à partir des fermés, est égal à ${}^{\omega}A^{\omega}$. Et pourtant, il existe des langages rationnels, (par exemple ${}^{\omega}A^{\omega} - {}^{\omega}a^{\omega}$), différents de ${}^{\omega}A^{\omega}$ contenant un mot w tel que $F(w) = A^*$.

Preuve : Si $R = \bigcap_i \bigcup_j F_{i,j}$ contient un mot w tel que $F(w) = A^*$, on a : $\forall i \ \exists j(i) \ w \in F_{i,j(i)}$. Si les $F_{i,j}$ sont des fermés, on a alors, pour tout i, $F_{i,j(i)} = {}^{\omega}A^{\omega}$. Donc $R = {}^{\omega}A^{\omega}$. Pour un langage R faisant partie de la hiérarchie borélienne définie à partir des fermés, on démontre le résultat par récurrence sur le nombre de couples $(\bigcap, \bigcup)$. •

La topologie de ${}^{\omega}A^{\omega}$ étant peu agréable, nous n'y aurons pas recours, sauf par commodité de langage.

Chapitre 1

Caractérisations des codes

Un code est un langage C de mots finis tel que tout mot de C^+ a une seule C-factorisation. Le but de cette section est d'obtenir en termes de mots infinis des conditions nécessaires et suffisantes pour qu'un langage C de mots finis soit un code.

L'une des caractérisations (proposition 1.1.2) est relative à l'unicité de la C-factorisation de certains mots infinis (les mots infinis à droite de la forme u^ω pour $u \in C^+$), les autres concernent les mots infinis rationnels et sont relatives à la forme et au nombre de C-factorisations de ces mots.

On obtient très facilement des conditions suffisantes pour qu'un langage soit un code. En effet, si $C \subset A^+$ n'est pas un code, il existe un élément u de C^+ ayant deux C-factorisations distinctes. Si l'on code ces deux factorisations par 0 et par 1, l'ensemble des C-factorisations de u^ω (et de ${}^\omega u^\omega$) a la puissance de $\{0,1\}^N$, et u^ω (ainsi que ${}^\omega u^\omega$) a des C-factorisations non ultimement périodiques. Les conditions nécessaires sont moins évidentes, et finalement font la différence entre mots infinis et mots bi-infinis. C'est ainsi que les mots infinis d'un seul côté, plus contraignants, donnent une caractérisation des codes très intéressante (les mots de la forme u^ω pour $u \in C^+$ ont une seule C-factorisation) dont l'équivalent n'existe pas pour les mots bi-infinis.

Les diverses caractérisations des codes seront illustrées dans les prochaines sections, par exemple dans la recherche de caractérisations des codes précirculaires ou des codes à délai borné, ou dans l'étude des biω-codes rationnels. D'autres applications de la proposition 1.1.2 peuvent être trouvées dans [42] et dans [22] .

La notion usuelle de code admet des caractérisations bien connues [8].

Définition 1.0.1 *Soit $C \subset A^+$ et $\varphi : X \to C$ une bijection. On dit que C est un* **code** *s'il vérifie l'une des propriétés (équivalentes) suivantes :*

- pour tous u, v de C on a :

$$uC^* \cap vC^* \neq \emptyset \Rightarrow u = v \,,$$

- *tout mot de C^+ a une seule C-factorisation,*
- *$\varphi : X^* \to C^*$ est injective,*
- *C^* est un monoïde libre de base C,*
- *C^* est un monoïde stable de base C.*

1.1 Codes et factorisation unique

Le lemme suivant permet d'avoir des conditions, exprimées en termes de mots infinis, suffisantes pour avoir un code.

Lemme 1.1.1 *Si $C \subset A^+$ n'est pas un code, il existe $u \in C^+$ tel que :*
- *l'ensemble des C-factorisations de u^ω (et de ${}^\omega u^\omega$) a la puissance du continu,*
- *u^ω ainsi que ${}^\omega u^\omega$ ont des C-factorisations non ultimement périodiques.*

Preuve : Soit $\varphi : X^* \to C^*$ un morphisme de codage pour C. Si C n'est pas un code, il existe $u \in C^+$, $y, z \in X^+$, $y \neq z$, tels que $\varphi(y) = \varphi(z) = u$. Sans perte de généralité, on peut supposer que les premières lettres de y et z diffèrent. Les éléments de $\{y, z\}^\omega$ sont des C-factorisations distinctes de u^ω. Le mot u appartient à C^+ et u^ω ainsi que ${}^\omega u^\omega$ ont une infinité non dénombrable de C-factorisations. Les mots non ultimement périodiques de $\{y, z\}^\omega$ sont des C-factorisations non ultimement périodiques de u^ω. De même ${}^\omega u^\omega$ a des C-factorisations non ultimement périodiques. •

Théorème 1.1.2 *[42] Si C est inclus dans A^+, on a équivalence entre les propriétés :*
- *C est un code,*
- *pour tout $u \in C^+$, u^ω a une seule C- factorisation.*

Preuve : La condition suffisante est donnée par le lemme 1.1.1.

Pour démontrer la condition nécessaire, considérons $v \in C^+$, $v = v_1 v_2 ... v_n$ (où $\forall i\ v_i \in C$) tel que v^ω ait deux C-factorisations différentes : $v^\omega = (v_1 v_2 ... v_n)^\omega = u_1 u_2 ... v_p ...$ (où $\forall i\ u_i \in C$). Sans perte de généralité, on peut supposer que $v_1 \neq u_1$. v étant de longueur finie, il existe un facteur gauche w de v et $i, j, k, m \geq 1$ tels que $k < m$, $u_1 ... u_k = v^i w$ et $u_1 ... u_m = v^{i+j} w$. Le mot $v^{i+j} w = u_1 ... u_m = (v_1 v_2 ... v_n)^j u_1 ... u_k$ a deux C-factorisations distinctes donc C n'est pas un code. •

Cette proposition n'a pas d'équivalent dans le cas bi-infini : par exemple, pour le code $C = \{ab, ba\}$, ${}^\omega(abba)^\omega$ n' a qu'une seule C-factorisation alors que ${}^\omega(ab)^\omega$ en a deux.

1.2 Forme des factorisations des mots rationnels

Les trois lemmes suivants se basent sur le même raisonnement que la démonstration précédente.

Lemme 1.2.1 *Soient $u \in A^+$ primitif, $u' \in A^+$, $v, v' \in A^*$ ou ${}^\omega A$ (resp. A^ω). Si $vu^\omega = v'u'^\omega$ (resp.${}^\omega uv = {}^\omega u'v'$) alors u' est une puissance d'un conjugué de u.*

Preuve: Une infinité de u sont coupés par les u'. Or u ne peut être coupé qu'en un nombre fini d'endroits, donc il existe deux u' coupant u au même endroit. Il existe donc une puissance de u' qui est une puissance d'un conjugué de u. On peut conclure car u est primitif. •

Lemme 1.2.2 *Soit $G \subset A^+$. Tout mot ultimement périodique de G^ω ou de ${}^\omega G^\omega$ (resp. périodique de ${}^\omega G^\omega$) admet une G-factorisation ultimement périodique (resp. périodique).*
Autrement dit, *si $vu^\omega \in G^\omega$ (resp. ${}^\omega wvu^\omega \in {}^\omega G^\omega$; resp. ${}^\omega u^\omega \in {}^\omega G^\omega$), il existe v' et $u' \in G^+$ (resp. $w', v', u' \in G^+$; resp. $u' \in G^+$) tels que $vu^\omega = v'u'^\omega$ (resp. ${}^\omega wvu^\omega = {}^\omega w'v'u'^\omega$; resp. ${}^\omega u^\omega = {}^\omega u'^\omega$).*

Preuve: Une infinité de u sont coupés par les mots de G. u ne peut être coupé qu'en un nombre fini d'endroits. Il existe donc $v', u' \in G^+$ tels que $vu^\omega = v'u'^\omega$. •

Le lemme suivant est une conséquence immédiate du lemme 1.2.2.

Lemme 1.2.3 *Soient $G \subset A^+$, $\varphi : X^* \to G^*$ un morphisme de codage pour G et $x \in X^\omega$ tel que $\varphi(x)$ soit ultimement périodique. Il existe $y, z \in X^*, t \in X^\omega$ tels que $x = yzt$ et $\varphi(t) = \varphi(z)^\omega$.*

Preuve: Si $\varphi(x) = vu^\omega$, comme dans la preuve du lemme 1.2.2, on peut trouver y, z, t tels que $x = yzt$, $\varphi(t) = \varphi(z)^\omega$ et $\varphi(z)$ est une puissance d'un conjugué de u. •

Lemme 1.2.4 *Soient $C \subset A^+$ un code et $\varphi : X^* \to C^*$ un morphisme de codage pour C. Pour tout $u \in A^+, u^+ \cap C^+$ est vide ou de la forme $(u^n)^+$. Dans $\varphi^{-1}(u^+)$ seul le mot $x = \varphi^{-1}(u^n)$ peut être primitif et on a : $\varphi^{-1}(u^+) = x^+$. Si u est primitif, x est primitif.*

Preuve :
- Si $u^+ \cap C^+$ n'est pas vide, appelons n le plus petit entier tel que $u^n \in C^+$. Considérons $u^p \in C^+$ et h, $0 \leq h < n$, $h \equiv p$ (*mod.* n). Comme C est un code, les C-factorisations $(u^p)^n$ et $(u^n)^p$ de u^{pn} coïncident. Donc u^{p-n} appartient à C^*, et par induction u^h aussi. Donc $h = 0$.
- Si $\varphi(x) = u^n$ et $\varphi(y) = u^p$, p est un multiple de n et, puisque C est un code, y un multiple de x. y n'est primitif que si $p = n, y = x$ et x est primitif.
- Si $x = y^p$, on a $u^n = \varphi(y)^p$. Si u est primitif, $\varphi(y)$ est dans $u^+ \cap C^+$, donc $y \in x^+$ et par conséquent $p = 1$. Donc x est primitif. •

Dans les caractérisations suivantes, on remarquera la différence entre le cas infini et le cas bi-infini, ainsi que la plus grande difficulté de la preuve dans le cas des mots bi-infinis périodiques.

Théorème 1.2.5 *Si C est inclus dans A^+, les propriétés suivantes sont équivalentes :*

- C est un code,

- toute C-factorisation d'un mot infini ultimement périodique est ultimement périodique,

- toute C-factorisation d'un mot infini **périodique** *est* **ultimement périodique,**

- toute C-factorisation d'un mot bi-infini ultimement périodique à droite est ultimement périodique à droite,

- toute C-factorisation d'un mot bi-infini ultimement périodique est ultimement périodique,

- toute C-factorisation d'un mot bi-infini **périodique** *est* **périodique.**

Preuve : Les conditions sont suffisantes d'après le lemme 1.1.1.

Réciproquement, considérons un code C et $\varphi : X^* \to C^*$ un morphisme de codage pour C.

Soit $x \in X^\omega$ tel que $\varphi(x)$ soit ultimement périodique. Il existe $y, z \in X^*, t \in X^\omega$ tels que $x = yzt$ et $\varphi(t) = \varphi(z)^\omega$ (lemme 1.2.3). C étant un code, comme $\varphi(z)$ est dans C^+, d'après la proposition 1.1.2, $t = z^\omega$ et x est ultimement périodique.

Le cas des mots bi-infinis ultimement périodiques d'un côté ou ultimement périodiques se résout de la même façon.

Il reste à étudier le cas des mots bi-infinis périodiques. Soit $x \in {}^\omega X^\omega$ tel que $\varphi(x)$ soit périodique. Comme on vient de le démontrer x est ultimement périodique. Posons $x = {}^\omega zyz'^\omega$. On peut prendre z et z' primitifs. Soient v et w primitifs tels que $\varphi(z) = v^n$ et $\varphi(z') = w^p$. $\varphi(x) = {}^\omega v\varphi(y)w^\omega$ est périodique donc v et w sont conjugués et $v = v_1v_2$, $w = v_2v_1$ et $\varphi(y) = v^q v_1$ car v est primitif. On a : $\varphi(z^p y) = v^{np+q}v_1 = \varphi(yz'^n)$, donc $z^p y = yz'^n$ (φ est injectif), z^p et z'^n sont conjugués, z et z' étant primitifs le sont aussi. Donc x est de

la forme $x = {}^{\omega}zy'z^{\omega}$. $\varphi(x) = {}^{\omega}v\varphi(y')v^{\omega}$ est périodique et v primitif donc $\varphi(y')$ appartient à v^+; d'après le lemme 1.2.4, z étant primitif également, y' appartient à z^+ et $x = {}^{\omega}z^{\omega}$. Les C-factorisations des mots bi-infinis périodiques sont périodiques. •

Même si C est un code, il se peut qu'un mot infini périodique ait une C-factorisation non périodique. Par exemple, le mot périodique $(ab)^{\omega}$ possède une factorisation non périodique: $(a, ba, ba, ...)$ sur le code {a,ba}.

1.3 Nombre de factorisations des mots rationnels

Théorème 1.3.1 *Si C est inclus dans A^+, les propriétés suivantes sont équivalentes :*
- C est un code,
- tout mot infini ultimement périodique a un nombre fini de C-factorisations,
- tout mot infini périodique a un nombre fini de C-factorisations,
- tout mot bi-infini ultimement périodique a un nombre fini de C-factorisations,
- tout mot bi-infini périodique a un nombre fini de C-factorisations.

Preuve: Les conditions sont suffisantes d'après le lemme 1.1.1.

Regardons la condition nécessaire dans le cas des mots bi-infinis. Soit C un code et $\varphi : X^* \to C^*$ un morphisme de codage pour C. Considérons un mot ultimement périodique $w = {}^{\omega}u'vu^{\omega}$ de ${}^{\omega}C^{\omega}$. On peut supposer que u et u' sont primitifs. Par la proposition 1.2.5, toutes les C-factorisations de w sont de la forme ${}^{\omega}x'yx^{\omega}$ où, si x et x' sont choisis primitifs, $\varphi(x) = v^n$ et $\varphi(x') = v'^p$ pour certains entiers n et p et des conjugués v de u et v' de u' (lemme 1.1.1). Fixons de tels x et x', et considérons $F_{x,x'}$ l'ensemble des C-factorisations de w de la forme ${}^{\omega}x'yx^{\omega}$. Considérons deux éléments ${}^{\omega}x'yx^{\omega}$ et ${}^{\omega}x'y'x^{\omega}$ de $F_{x,x'}$. Fixons w_0 un représentant de w, et des C-décompositions de w_0 associées à ${}^{\omega}x'yx^{\omega}$ et à ${}^{\omega}x'y'x^{\omega}$. Quitte à allonger y', on peut supposer que $\varphi(y)$ coïncide avec un facteur de $\varphi(y')$ dans ces décompositions de w_0. On a alors $\varphi(y') = v'^{\alpha}\varphi(y)v^{\beta}$ puisque v et v' sont primitifs. Si $\alpha \equiv 0$ $(mod.\ p)$ et $\beta \equiv 0$ $(mod.\ n)$, $\varphi(y') \in \varphi(x'^*yx^*)$; C étant un code, $y' \in x'^*yx^*$ et donc ${}^{\omega}x'y'x^{\omega} = {}^{\omega}x'yx^{\omega}$.

Le nombre d'éléments de $F_{x,x'}$ est par conséquent majoré (strictement) par le produit $n \cdot p$. Pour u primitif, posons

$$U_u = \{v \text{ conjugué de } u \mid \exists n > 0\ \exists x \in X^*,\ x \text{ primitif}, \varphi(x) = v^n\}.$$

Pour v appartenant à U_u, l'unicité du mot primitif $x \in X^*$ vérifiant $\varphi(x) = v^n$ est assurée par le lemme 1.2.4. Cet x sera noté x_v, la puissance n correspon-

dante sera notée n_v. Les ensembles U_u et $U_{u'}$ sont finis. L'ensemble F des C-factorisations de w est donc fini.

Dans le cas des mots infinis, on peut reprendre la démonstration précédente en la simplifiant, ou encore considérer que à toute C-factorisation d'un mot vu^ω on peut faire correspondre au moins une C-factorisation du mot ${}^\omega uvu^\omega$, à condition d'avoir pris u dans C^+, ce qui est possible d'après le lemme 1.2.2 . •

Le lemme suivant permet d'obtenir une majoration moins forte du cardinal de F. Cette majoration servira dans la preuve de la proposition 4.4.2. Les notations sont celles de la démonstration précédente. On définit sur U_u et sur $U_{u'} \times U_u$ les relations d'équivalence suivantes :
pour $v, w \in U_u$, $v', w' \in U_{u'}$:

$$v \simeq w \Leftrightarrow x_v \text{ et } x_w \text{ sont conjugués.}$$

$$(v', v) \cong (w', w) \Leftrightarrow v \simeq w \text{ et } v' \simeq w' \ .$$

Lemme 1.3.2 *Le nombre de factorisations sur un code C*
- d'un mot $u''u^\omega$ de C^ω est majoré par $\sum n_v$, où l'on somme sur les $\simeq$-classes d'équivalence de U_u,
- d'un mot ${}^\omega u'u''u^\omega$ de ${}^\omega C^\omega$ est majoré par $\sum n_{v'} \cdot n_v$, où l'on somme sur les $\cong$-classes d'équivalence de $U_{u'} \times U_u$,
- d'un mot ${}^\omega u^\omega$ de ${}^\omega C^\omega$ est égal au nombre de $\simeq$-classes d'équivalence de U_u.

Preuve : On a : $(v', v) \cong (w', w)$ si et seulement si $\{ {}^\omega x_{v'} y x_v{}^\omega \mid y \in X^* \} = \{ {}^\omega x_{w'} y' x_w{}^\omega \mid y' \in X^* \}$. On en déduit que le cardinal de F est majoré par $\sum n_{v'} \cdot n_v$, où l'on peut sommer uniquement sur les $\cong$-classes d'équivalence de couples (v', v). Dans le cas d'un mot périodique ${}^\omega u^\omega$, toute C-factorisation étant périodique (proposition 1.2.5), le cardinal de F est égal au nombre de $\simeq$-classes d'équivalence. •

Grâce au lemme suivant, la proposition 1.3.1 a un corollaire immédiat en termes de Z-mots.

Lemme 1.3.3 *Si un Z-mot w a une C-factorisation périodique c, w est périodique et $\tau_w^{-1}(c)$ est fini.*

Preuve : Soient $c_1, c_2, ..., c_n \in C$ tels que ${}^\omega(c_1 c_2 ... c_n)^\omega$ soit un représentant de c. Appelons u la racine primitive de $c_1 c_2 ... c_n$; on a $c_1 c_2 ... c_n = u^p$. Le Z-mot w est un représentant de ${}^\omega u^\omega$, il est donc périodique. Puisque u est primitif, la seule façon de positionner $c_1 c_2 ... c_n$ sur w est de faire coïncider les débuts de $c_1 c_2 ... c_n$ et de u. D'après le lemme 0.2.1, on peut supposer que les lettres

w_0 et u_0 coïncident. On voit alors facilement que les divers positionnements de $c_1c_2...c_n$ sur w donnent au plus p décompositions de w. •

Corollaire 1.3.4 *Si C est inclus dans A^+, les propriétés suivantes sont équivalentes :*
- C est un code,
- tout Z-mot ultimement périodique a un nombre fini de C-décompositions,
- tout Z-mot périodique a un nombre fini de C-décompositions.

Preuve : Si w est un Z-mot non périodique, la correspondance naturelle τ_w entre ses C-décompositions et ses C-factorisations est bijective. D'où le résultat pour les mots ultimement périodiques non périodiques.

Si w est un Z-mot périodique représentant un mot bi-infini ${}^\omega u^\omega$, d'après le lemme précédent, à chaque C-factorisation périodique de ${}^\omega u^\omega$ correspond un nombre fini de C-décompositions de u^Z. Or si C est un code, toutes les C-factorisations d'un mot périodique sont périodiques. Tout Z-mot périodique a donc un nombre fini de C-décompositions. •

On pourrait croire que, si C est un code, tout mot infini a « peu » de C-factorisations. Les exemples suivants prouvent qu'il n'en est rien en général. Nous verrons au chapitre 6 que, dans le cas des codes rationnels, tous les mots infinis ont un nombre fini de factorisations.

Proposition 1.3.5 *Le nombre de factorisations d'un mot infini sur un code peut être fini, infini dénombrable ou infini non dénombrable.*

Exemple : Posons $C = C_1 \cup C_2$ où $C_1 = \{aba^2b^2a^3b^3...a^nb^na^{n+1} \mid n \geq 0\}$ et $C_2 = \{b^pa^qb^p \mid q \geq 2, 0 < p < q\}$. Le langage C est un code (suffixe).

Le mot infini $w_0 = aba^2b^2...a^nb^na^{n+1}b^{n+1}...$ a une infinité dénombrable de C-factorisations. Ce sont les C-factorisations de la forme :
$(ab...a^{n-1}b^{n-1}a^n)(b^na^{n+1}b^n)(ba^{n+2}b)...(b^{n+h}a^{n+2h+1}b^{n+h})(b^{h+1}a^{n+2h+2}b^{h+1})...$
où $n \geq 1$. •

Exemple : Soit $A = \{a, b\}$ et $C = \{uab^n \mid |u| = n, n \geq 0\}$. Le langage C est un code (suffixe). Nous allons montrer que le mot : $w_0 = ab^{i_0}ab^{i_1}...ab^{i_n}...$, où $i_0 = 0, i_1 = 1, i_{n+2} = i_{n+1} + i_n + 1$ pour tout $n \geq 0$, a une infinité non dénombrable de C-factorisations. Nous montrons pour cela que, pour tout découpage de w_0 : $w_0 = uv$, u est dans C^* et v a au moins deux C-factorisations.

Soit u un facteur gauche de w_0. Clairement, si $|u| \leq 6$, u est dans C^*. On continue par induction. Supposons que pour tout $n \leq N$ et tout p, $0 \leq p \leq i_n$, le mot $ab^{i_0}ab^{i_1}...ab^{i_{n-1}}ab^p$ appartienne à C^*. Alors le mot $u = ab^{i_0}ab^{i_1}...ab^{i_N}ab^q$ vérifie :
- si $0 \leq q \leq i_N$, $u = (ab^{i_0}ab^{i_1}...ab^{i_{N-1}}ab^p) \cdot (b^qab^q)$ où $p = i_N - q$ vérifie :

$0 \leq p \leq i_N$,
- si $i_N < q \leq i_{N+1}$, $u = (ab^{i_0}ab^{i_1}...ab^{i_{N-2}}ab^p) \cdot (b^r ab^{i_N} ab^q)$ où $r = q - i_N - 1$ et $p = i_{N-1} - r$ vérifie $0 \leq p \leq i_{N-1}$. Le mot u appartient donc à C^*.

Considérons un découpage de w_0 : $w_0 = uv$. Montrons qu'il existe deux mots de C différents $x(v)$ et $y(v)$ tels que v appartienne à $x(v)C^\omega \cap y(v)C^\omega$. Posons $v = b^{j_0}ab^{i_n}ab^{i_{n+1}}...$ où $0 \leq j_0 \leq i_{n-1}$. Alors $v = (b^{j_0}ab^{j_0})(b^{j_1}ab^{j_1})...(b^{j_h}ab^{j_h})...$ où $j_{h+1} = i_{n+h} - j_h$ et j_h vérifie $0 \leq j_h \leq i_{n+h}$ pour tout $h \geq 0$; on peut poser $x(v) = b^{j_0}ab^{j_0}$. Le mot v a aussi la C-factorisation :
$v = (b^{j_0}ab^{i_n}ab^{k_0})(b^{k_1}ab^{k_1})...(b^{k_h}ab^{k_h})...$ où $k_0 = j_0 + i_n + 1$, $k_{h+1} = i_{n+1+h} - k_h$ et k_h vérifie $0 \leq k_h \leq i_{n+1+h}$ pour tout $h \geq 0$; on peut poser $y(v) = b^{j_0}ab^{i_n}ab^{k_0}$.

On peut alors définir une application injective δ de $\{0,1\}^N$ dans l'ensemble des C-factorisations de w_0 de la façon suivante : soit $\beta = (\beta_n)_n \in \{0,1\}^N$, on pose $\delta(\beta) = (z_n)_n$ où $z_0 = x(w_0)$ si $\beta_0 = 0$ et $z_0 = y(w_0)$ si $\beta_0 = 1$, $z_n = x((z_0z_1...z_{n-1})^{-1}w_0)$ si $\beta_n = 0$ et $z_n = y((z_0z_1...z_{n-1})^{-1}w_0)$ si $\beta_n = 1$. Le mot w_0 a une infinité non dénombrable de C-factorisations. •

Chapitre 2

Codes précirculaires

Les **codes circulaires** (ou encore codes très purs) ont été introduits par J. L. Lassez, A. Restivo et d'autres vers 1976 [34] [46]. Dans cette section, nous verrons que les codes circulaires sont intéressants du point de vue des mots bi-infinis. En effet, les Z-mots périodiques ont au plus une décomposition sur ces codes ; de plus, ce fait est caractéristique des codes circulaires. Cette propriété ne s'étend hélas pas aux Z-mots quelconques, sauf dans le cas des codes rationnels, ce que nous verrons dans la sixième partie.

Dans cette section, nous introduisons la notion de **code précirculaire**, un peu plus générale que celle de code circulaire. Les codes précirculaires sont aux factorisations ce que les codes circulaires sont aux décompositions. Cela signifie que les codes précirculaires sont caractérisés par l'unicité des factorisations des mots bi-infinis périodiques.

Une autre notion généralise celle de code circulaire : c'est la notion de code **presque circulaire** introduite par M. Leconte [36], [37]. Dans la dernière partie de cette section, nous montrons qu'un code précirculaire est presque circulaire et nous obtenons une condition nécessaire et suffisante pour qu'un code presque circulaire soit un code précirculaire.

Nous aurons besoin dans cette partie de la notion d'interprétation. Soit C un langage inclus dans A^+. On définit :
$A^-C = \{x \in A^* \mid \exists w \in C\ \exists v \in A^+\ w = vx\}$,
$CA^- = \{x \in A^* \mid \exists w \in C\ \exists v \in A^+\ w = xv\}$.
On dit que la suite $(v', v_1, ..., v_n, v'')$ est une **C-interprétation** de u si : $n \geq 0, u = v'v_1...v_nv''$, $v' \in A^-C$, $v'' \in CA^-$ et pour tout i, $1 \leq i \leq n$, $v_i \in C$ [8]. Une C-interprétation est dite **cyclique** si $v''v' \in C \cup \{\varepsilon\}$. Pour toute C-factorisation $(v_1, ..., v_n)$ de u, $(\varepsilon, v_1, ..., v_n, \varepsilon)$ est une C-interprétation cyclique de u, et réciproquement. Une C-interprétation $(v_0, v_1, ..., v_n, v_{n+1})$ et une C'-interprétation $(u_0, u_1, ..., u_p, u_{p+1})$ d'un même mot sont dites **adjacentes** s'il existe $k, 0 \leq k \leq n$ et $h, 0 \leq h \leq p$ tels que $v_0v_1...v_k = u_0u_1...u_h$. Une C-factorisation $(v_1, ..., v_n)$ et une C'-interprétation $(u_0, u_1, ..., u_p, u_{p+1})$

du même mot sont dites adjacentes si les interprétations $(\varepsilon, v_1, ..., v_n, \varepsilon)$ et $(u_0, u_1, ..., u_p, u_{p+1})$ le sont.

Donnons quelques exemples quand le langage C est : $\{ab, ba, a\}$.
Les suites (ε, ba, ba, b) et (b, ab, ab, ε) sont des C-interprétations non cycliques et non adjacentes du mot *babab*.
La suite (b, ab, a) est une C-interprétation cyclique du mot *baba*.
Les suites $(\varepsilon, ba, ab, \varepsilon)$ et (b, a, a, b) sont deux C-interprétations adjacentes du mot *baab*, la première est cyclique, la seconde ne l'est pas.

2.1 Notion de code précirculaire

La définition des langages précirculaires s'inspire de la définition des codes circulaires. Nous verrons plus loin que les langages précirculaires sont aussi des codes.

Définition 2.1.1 *Soit $C \subset A^+$. C est un* **langage précirculaire** *si et seulement si*

$$\forall\, n, p \geq 1\ \forall u_0, \ldots, u_{n-1}, v_0, \ldots, v_{p-1} \in C\ \forall\, t, s \in A^*\ \textit{tels que}\ v_0 = ts$$

$$(\ u_0 ... u_{n-1} = s v_1 ... v_{p-1} t\ \Rightarrow\ n = p\ \textit{et}\ \exists h\ \forall i\ u_i = v_{(i+h)\ mod\ p}\ )\ .$$

Autrement dit : pour tout $u \in C^*$, pour toute C-factorisation $(u_0, ..., u_{n-1})$ et toute C-interprétation cyclique $(s, v_1, ..., v_{p-1}, t)$ de u, si l'on pose $v_0 = ts$, on a $n = p$ et $(v_0, ..., v_{p-1})$ est une permutation circulaire de $(u_0, ..., u_{n-1})$. Ou encore, tout mot de C^* écrit sur un cercle a seulement une C-factorisation (mais peut-être plusieurs découpages suivant cette factorisation).

Pour simplifier les notations, dans la suite nous écrirons v_i (resp. u_i) au lieu de $v_{i\ mod\ p}$ (resp. $u_{i\ mod\ n}$).

Tout sous-ensemble d'un langage précirculaire est précirculaire ; un singleton $\{u\}$ est un langage précirculaire si $u \neq \varepsilon$; $\{a, bab\}$, $\{ba^3, ba^2ba, a^2ba^2\}$, $\{b^2\} \cup ab(a^2b)^*$ sont des langages précirculaires ; nous verrons plus loin que les codes circulaires sont des langages précirculaires. Si deux mots distincts ont des puissances conjuguées, ils ne peuvent faire partie d'un même langage précirculaire.

La notion de langage précirculaire peut s'exprimer en termes de morphisme de codage. Cela permettra de voir qu'un langage précirculaire est un code, ce qui justifie l'appellation « code précirculaire ».

Proposition 2.1.1 *Soit $C \subset A^+$ et $\varphi : X^* \to C^*$ un morphisme de codage pour C. C est un langage précirculaire si et seulement si*

$$\forall\, x, x' \in X^+\ (\ \varphi(x)\ \textit{et}\ \varphi(x')\ \textit{conjugués}\ \Rightarrow x\ \textit{et}\ x'\ \textit{conjugués}\ )\ .$$

Preuve : Considérons $p, q \geq 1$, $x_i \in X$ $(0 \leq i \leq p-1)$, $y_j \in X$ $(0 \leq j \leq q-1)$. Posons $x = x_0...x_{p-1}$, $y = y_0...y_{q-1}$, $u_i = \varphi(x_i)$ $(0 \leq i \leq p-1)$, $v_j = \varphi(y_j)$ $(0 \leq j \leq q-1)$.

La proposition découle des remarques suivantes :

$$\varphi(x) \text{ et } \varphi(y) \text{ sont conjugués } \Leftrightarrow \exists k \; u_0...u_{p-1} = sv_{1+k}...v_{q-1+k}t \text{ et } ts = v_k$$

$$x \text{ et } y \text{ sont conjugués } \Leftrightarrow q = p \text{ et } \exists h \; \forall i \; u_i = v_{i+h} \; .\bullet$$

Corollaire 2.1.2 : *Un langage précirculaire est un code.*

Preuve : Soit $\varphi : X^* \to C^*$ un morphisme de codage pour le langage précirculaire C. Soient $x, y \in X^*$ tels que $\varphi(x) = \varphi(y)$. $\varphi(xx) = \varphi(xy)$, donc $\varphi(xx)$ et $\varphi(xy)$ sont conjugués ainsi que xx et xy. D'où $x = y$. Le morphisme φ est injectif et donc C est un code. •

Mais tous les codes ne sont pas précirculaires : le code $\{ab, ba\}$ ne l'est pas.

2.2 Codes précirculaires et mots bi-infinis

On peut remarquer qu'un mot bi-infini périodique peut avoir plusieurs C-factorisations, même si C est un code. Par exemple, $\{ab, ba\}$ est un code, et cependant on a : ${}^{\omega}(ab)^{\omega} = {}^{\omega}(ba)^{\omega}$. C'est dans le but d'exclure cette possibilité que nous avons introduit la notion de code précirculaire. En fait, cela donne une caractérisation : nous démontrons que les langages qui factorisent de façon unique les mots bi-infinis périodiques sont les codes précirculaires. La seconde caractérisation des codes intervient dans la preuve de ce résultat.

Lemme 2.2.1 *Soit C un code précirculaire inclus dans A^+ et $\varphi : X^* \to C^*$ un morphisme de codage pour C. Si xz^{ω} $(x \in {}^{\omega}X, \; z \in X^+)$ est une C-factorisation d'un mot de ${}^{\omega}Av^{\omega}$ $(v \in A^+)$, on a : $\varphi^{-1}({}^{\omega}Av^{\omega}) = {}^{\omega}Xz^{\omega}$.*

Preuve : Il s'agit de montrer que toute C-factorisation d'un mot de ${}^{\omega}Av^{\omega}$ appartient à ${}^{\omega}Xz^{\omega}$. Soit $wv^{\omega}(w \in {}^{\omega}A, v \in A^+)$ l'image de xz^{ω} par φ, et considérons une C-factorisation d'un mot $w'v^{\omega}$ de ${}^{\omega}Av^{\omega}$. Sans perte de généralité on peut supposer que v et z sont primitifs. D'après la proposition 1.2.5, puisque C est un code, cette C-factorisation est de la forme $x'z'^{\omega}(x' \in {}^{\omega}X, z' \in X^+, z'$ primitif). Pour obtenir le résultat, il suffit de prouver que z et z' sont conjugués. Soient u et u' les racines primitives de $\varphi(z)$ et $\varphi(z')$. On a $\varphi(x')\varphi(z')^{\omega} = w'v^{\omega}$ donc u' et v sont conjugués (lemme 1.2.1). De même $\varphi(x)\varphi(z)^{\omega} = wv^{\omega}$ donc u et v sont conjugués, et donc u et u' sont conjugués. Posons $\varphi(z) = u^n$ et $\varphi(z') = u'^p$; $\varphi(z)^p$ et $\varphi(z')^n$ sont conjugués. Le code C

étant précirculaire, z^p et z'^n sont conjugués (proposition 2.1.1) et donc z et z' sont conjugués. •

Théorème 2.2.2 *Un langage C inclus dans A^+ est un code précirculaire si et seulement si tout mot bi-infini périodique a au plus une C-factorisation.*

Preuve : Supposons que tout mot périodique de ${}^\omega A^\omega$ a au plus une C-factorisation. Pour tous $n, p \geq 1$ $u_0, ..., u_{n-1}, v_1, ..., v_{p-1} \in C$ $t, s, u \in A^*$ tels que $v_0 = ts \in C$ et $u = u_0...u_{n-1} = sv_1...v_{p-1}t$, le mot périodique ${}^\omega u^\omega$ a les C-factorisations $(u_{i \bmod n})_{i\in\mathbf{Z}}$ et $(v_{i \bmod p})_{i\in\mathbf{Z}}$. Par hypothèse il existe donc i tel que: $(u_0, ..., u_{n-1}) = (v_{(i+1) \bmod p}, ..., v_{(i+n) \bmod p})$. D'après l'égalité des longueurs :
$|u_0...u_{n-1}| = |v_0...v_{p-1}| = |v_{(i+1) \bmod p}...v_{(i+p) \bmod p}|$, on a $p = n$ et C est un code précirculaire.

Réciproquement, considérons un code précirculaire C et $\varphi : X^* \to C^*$ un morphisme de codage pour C. Soit ${}^\omega u^\omega$ un mot périodique quelconque et deux C-factorisations de ${}^\omega u^\omega$. Ces factorisations sont périodiques puisque C est un code (proposition 1.1.2) : notons-les ${}^\omega z^\omega$ et ${}^\omega z'^\omega$ où z et z' sont des mots primitifs de X^+. D'après le lemme 2.2.1, la factorisation ${}^\omega z'^\omega$ appartient en fait à ${}^\omega z X^* z^\omega$ et donc, puisqu'ils sont primitifs, z et z' sont conjugués. Donc ${}^\omega z^\omega = {}^\omega z'^\omega$ et ${}^\omega u^\omega$ n'a qu'une seule C- factorisation. •

Dans cette caractérisation, on ne peut pas remplacer le terme « périodique» par le terme « ultimement périodique », même dans le cas d'un code fini. Par exemple, le code $C = \{a^2, ab, b^2\}$ est précirculaire mais le mot ${}^\omega ab^\omega$ a deux C-factorisations. Cependant, des codes plus particuliers ont cette propriété, c'est le cas des codes circulaires que nous étudions maintenant.

2.3 Codes circulaires et Z-mots

Les codes circulaires (ou encore codes très purs) ont été introduits vers l'année 1976. Ces codes ont été caractérisés de diverses manières [46], [8].

Rappelons d'abord quelques définitions.

Définitions 2.3.1 *Soit $C \subset A^+$.*
- C est un code **pur** *si et seulement si c'est un code vérifiant :*

$$\forall u \in A^+ \; \forall n \geq 1 \; (u^n \in C^* \Rightarrow u \in C^*) \; .$$

- C est un **code circulaire** *si et seulement si*

$$\forall \, n, p \geq 1 \;\; \forall u_0, \ldots, u_{n-1}, v_0, \ldots, v_{p-1} \in C$$

$$\forall t \in A^* \ \forall s \in A^+ \text{ tels que } v_0 = ts$$
$$(u_0 ... u_{n-1} = s v_1 ... v_{p-1} t \ \Rightarrow \ n = p,\ t = \varepsilon \text{ et } \forall i \ u_i = v_i) .$$

Les codes purs et les codes circulaires peuvent être caractérisés en termes de morphisme de codage.

Proposition 2.3.1 *[8] Soit $C \subset A^+$ et $\varphi : X^* \to C^*$ un morphisme de codage pour C. Les propriétés suivantes sont équivalentes :*
- C est un **code circulaire**,
- C est la base d'un monoïde très pur, c'est-à-dire que C^* vérifie :

$$\forall u, v \in A^* \ (uv, vu \in C^* \ \Rightarrow \ u, v \in C^*),$$

- C est un code pur et φ vérifie :

$$\forall \ x, x' \in X^+ \ (\ \varphi(x) \text{ et } \varphi(x') \text{ conjugués } \Rightarrow x \text{ et } x' \text{ conjugués }) .$$

Proposition 2.3.2 *Soit $\varphi : X^* \to C^*$ un morphisme de codage pour un code C. Le code C est pur si et seulement si*

$$\forall y \in X^+ \ y \text{ primitif } \Rightarrow \ \varphi(y) \text{ primitif } .$$

Preuve : Considérons $y \in X^+$ tel que $\varphi(y)$ soit imprimitif : $\varphi(y) = u^n$ où $n > 1$. Si C est un code pur, comme u^n appartient à C^+, u appartient à C^+ et il existe z dans X^+ tel que $u = \varphi(z)$. Puisque C est un code, l'égalité $\varphi(z^n) = \varphi(y)$ donne $y = z^n$. Donc y n'est pas primitif.

Inversement, si $u^n \in C^+$, il existe y dans X^+ tel que $u^n = \varphi(y)$. Soit z la racine primitive de y : $y = z^p$, $p > 1$. D'après l'hypothèse, le mot $\varphi(z)$ est primitif. On a : $\varphi(z)^p = u^n$, donc u est une puissance de $\varphi(z)$. Or $\varphi(z)$ appartient à C^+ donc u aussi. Le code C est pur. •

La relation existant entre les notions de code circulaire et celle de code précirculaire se déduit aisément des propositions 2.1.1 et 2.3.1 :

Proposition 2.3.3 *Un langage est un code circulaire si et seulement si c'est un code précirculaire pur.*

Les deux conditions « pur» et « précirculaire » sont indispensables : le code pur $\{ab, ba\}$ n'est pas précirculaire, le code précirculaire $\{a, bab\}$ n'est pas pur.

Etudions maintenant les liens reliant les codes circulaires et la décomposition des Z-mots. L'analogie entre le théorème suivant et le théorème 2.2.2 est remarquable.

Théorème 2.3.4 *Soit C un langage inclus dans A^+. Les propriétés suivantes sont équivalentes:*
1 - C est un code circulaire,
2 - tout Z-mot périodique a au plus une C-décomposition,
3 - tout Z-mot ultimement périodique a au plus une C-décomposition.

Preuve: 3 $\Rightarrow$ 2: Evident. 2 $\Rightarrow$ 1: La négation de la définition des codes circulaires assure de l'existence d'un mot u de C^+ tel que les Z-mots représentant ${}^\omega u^\omega$ ont deux C-décompositions distinctes. 1 $\Rightarrow$ 3: Soit C un code circulaire. Considérons ${}^\omega uwv^\omega$ et deux C-décompositions d'un représentant de ce mot. Puisque C est un code, les C-factorisations associées sont ultimement périodiques (proposition 1.2.5). Soit $\varphi : X^* \to C^*$ un morphisme de codage pour C, et ${}^\omega xyz^\omega (x, z$ primitifs) l'une de ces factorisations. Comme C est précirculaire, l'autre factorisation peut s'écrire ${}^\omega xy_1z^\omega$ (lemme 2.2.1). Puisque C est un code pur, $\varphi(x)$ est primitif et ne peut pas se chevaucher ($\varphi(z)^2 = s\varphi(z)t \Rightarrow s = \varphi(z)$ ou $s = \varepsilon$); il en est de même pour $\varphi(z)$. Donc les C-décompositions coïncident à droite et à gauche. On peut donc choisir y et y_1 tels que $\varphi(y) = \varphi(y_1)$. On a alors $y = y_1$ puisque C est un code. Les deux factorisations et aussi les deux décompositions sont identiques. •

Malheureusement, le théorème précédent ne s'étend pas aux mots bi-infinis quelconques. Par exemple, le code $C = \{ab^nab^{n+1} \mid n \geq 1\} \cup \{ab\}$ est un code circulaire, mais les représentants du mot ${}^\omega(ab)ab^2ab^3ab^4...$ admettent deux C-décompositions.

2.4 Codes presque circulaires et codes précirculaires

De même que les codes précirculaires, les codes presque circulaires définis par M. Leconte [36] généralisent les codes circulaires. Nous allons ici comparer les codes précirculaires et les codes presque circulaires.

Rappelons qu'un semi-groupe est dit **monogène** s'il est de la forme u^+, pour un mot u de A^+.

Définition 2.4.1 *Un langage $C \subset A^+$ est un code* **presque circulaire** *si c'est un code vérifiant: $\forall s, t \neq \varepsilon$*

$$(ts \in C \text{ et } sC^*t \cap C^* \neq \emptyset \Rightarrow sC^*t \cap C^* \text{ est un semi-groupe monogène }).$$

Lemme 2.4.1 *Soit C un code précirculaire, $s \neq \varepsilon$, $t \neq \varepsilon$ tels que $ts \in C$. Le semi-groupe $sC^*t \cap C^*$ est vide ou tous ses éléments sont imprimitifs.*

Preuve: Soit $x \in sC^*t \cap C^*$ et $(u_0, ..., u_{n-1})$ la C-factorisation de x. Posons $y = s^{-1}xs$. Sa C-factorisation est $(u_i, ..., u_{i+n-1})$ pour un certain i.

- si $i = 1$, on pose $u = t$ et $v = xt^{-1}$,
- si $i \neq 1$, on pose $u = u_0...u_{i-2}t$ et $v = xu^{-1}$.

Puisque dans chaque cas $uv = vu = x$, il existe p et q tels que $u = (\sqrt{x})^p$ et $v = (\sqrt{x})^q$. Donc x est imprimitif. •

Cependant la condition précédente ne suffit pas pour avoir un code précirculaire. Par exemple, le code $C = \{abab, ba\}$ vérifie $sC^*t \cap C^* = (baba)^+$ ou $(abab)^+$pour tous $s, t \neq \varepsilon$ tels que $ts \in C$, mais C n'est pas un code précirculaire.

Lemme 2.4.2 *Soit C un code. Si $sC^*t \cap C^*$ (avec $t, s \neq \varepsilon$, $ts \in C$) et $s'C^*t \cap C^*$ (avec $t, s' \neq \varepsilon$, $ts' \in C$) sont monogènes, alors $s = s'$.*

Preuve: Soit $sC^*t \cap C^* = w^+$ et $s'C^*t \cap C^* = w'^+$. On a: $ww'^* \subset w^*$, $w'w^* \subset w'^*$ et donc $w'^* \subset w^*$ et $w^* \subset w'^*$. Par conséquent $w = w'$. Puisque C est un code twt^{-1} a une seule C- factorisation, donc $ts = ts'$ et $s = s'$. •

Lemme 2.4.3 *Soit C un code. Si les mots $t, s, u \neq \varepsilon$ vérifient $ts \in C$ et $\emptyset \neq sC^*t \cap C^* \subset u^+$, alors $sC^*t \cap C^*$ est monogène.*

Preuve: Considérons $u^q \in sC^*t \cap C^*$. Soit $p = inf\{q \mid u^q \in sC^*t \cap C^*\}$ et $h = q \ mod \ p$, $0 \leq h < p$. Il suffit de prouver que $h = 0$. Supposons le contraire. Puisque C est un code, les C-factorisations of u^{pq} obtenues à partir de celles de u^p et u^q sont identiques. Nous obtenons ainsi une C-factorisation de u^h. u^p et u^q sont dans sC^*t donc nous avons deux C-factorisations de $s^{-1}u^{pq}t^{-1}$. Ces C- factorisations sont nécessairement les mêmes donc u^h appartient à sC^*t. Ceci contredit la minimalité de p. •

Proposition 2.4.4 *Tout code précirculaire est presque circulaire.*

Preuve: Considérons $s, t \neq \varepsilon$ tels que $ts \in C$ et $G = sC^*t \cap C^* \neq \emptyset$. Soient x et y appartenant à G, ils sont imprimitifs (lemme 2.4.1): $x = (\sqrt{x})^p$, $y = (\sqrt{y})^q$ avec p et $q \geq 2$. De même xy appartient à G et donc $xy = (\sqrt{xy})^k$ avec $k \geq 2$. L'égalité $(\sqrt{x})^p(\sqrt{y})^q = (\sqrt{xy})^k$ implique que $\sqrt{x} = \sqrt{y} = \sqrt{xy}$ car ces mots sont primitifs [47] [45]. Donc G est inclus dans $(\sqrt{x})^+$. D'après le lemme 2.4.3, G is monogène. •

La réciproque de la proposition précédente est fausse. Par exemple, $C = \{ab, ba\}$ est un code presque circulaire (car $bC^*a \cap C^* = (ba)^+$ et $aC^*b \cap C^* = (ab)^+$) mais C n'est pas précirculaire.

Théorème 2.4.5 *Un code presque circulaire $C \subset A^+$ est un code précirculaire si et seulement si il vérifie : pour tous $s, t, u \neq \varepsilon$ tels que $sC^*t \cap C^* = u^+$ et $v_1 = ts \in C$, si $(u_1, ..., u_n)$ est la C-factorisation de u, l'unique C-interprétation cyclique de u qui commence par s et se termine par t $(s, v_2, ..., v_k, t)$ vérifie :*

$$k = n \text{ et } \exists h \; \forall i \; u_i = v_{i+h} \quad .$$

Preuve : Clairement la condition est nécessaire.
Inversement, considérons $x \in C^+$ et une C-interprétation cyclique de x : $(s, v_2, ..., v_k, t)$. On note $v_1 = ts$. Puisque C est un code presque circulaire, il existe u tel que $sC^*t \cap C^* = u^+$, donc $x = u^n$ pour un certain n. Si $(u_1, ..., u_p)$ est la C-factorisation de u, celle de $x(= u^n)$ est $(w_1, ..., w_{np}) = (u_1, ..., u_p)^n$. Si on pose : $(s, v_2, ..., v_k, t) =$
$(s, v_2, ..., v_{k_1}, t_1 s_1, v_{k_1+2}, ..., v_{k_2}, t_2 s_2, ..., t_{n-1} s_{n-1}, v_{k_{n-1}+2}, ..., v_k, t)$ où
$u = sv_2...v_{k_1} t_1 = s_1 v_{k_1+2}...v_{k_2} t_2 = ... = s_{n-1} v_{k_{n-1}+2}...v_k t$. Puisque C est un code presque circulaire, $sC^*t \cap C^*$ et les $s_iC^*t \cap C^*$ sont monogènes. D'après le lemme 2.4.2, on a donc $s = s_1 = ... = s_{n-1}$; de même, $t = t_1 = ... = t_{n-1}$. Alors $v_2...v_{k1} = v_{k1+2}...v_{k2} = ... = v_{k(n-1)+2}...v_k$, et donc $v_2 = v_{k_1+2} = ... = v_{k_{n-1}+2}$, $v_3 = v_{k_1+3} = ... = v_{k_{n-1}+3}$,..., $v_{k_1} = v_{k_2} = ... = v_k$ puisque C est un code. Par hypothèse faite sur C, $k_1 = p$ et il existe $h, 1 \leq h \leq k_1$ tel que $u_i = v_{(i+h) \bmod p}$ pour $1 \leq i \leq p$, et donc $k = np$ et h vérifie $u_i = v_{(i+h) \bmod k}$ pour $1 \leq i \leq k$. Le code C est précirculaire. •

Nous verrons dans dans la cinquième partie que ce dernier théorème donne une méthode pour décider si un langage rationnel donné est un code précirculaire.

Chapitre 3

Codes pour les mots infinis

Pour coder les mots infinis, certains auteurs, en particulier Do Long Van ([26], [25], [28]) ont choisi de considérer des codes constitués de mots finis et infinis. Le fait d'avoir des mots infinis dans le code permet d'avoir des factorisations finies de mots infinis. En revanche, un tel « code » ne sera pas un code au sens usuel. De plus, il faut décider si l'on ne considère que les factorisations finies, ou si l'on exige qu'un mot donné ait au plus une factorisation, finie ou infinie, ce qui donne deux notions de « code » différentes. Ces notions ont été étudiées dans les articles cités plus haut.

Nous avons, pour notre part, fait le choix de coder les mots infinis par des codes finitaires. La notion de code (finitaire) pour les mots infinis à droite a été introduite par L. Staiger [61] : c'est celle d'« ifl-code », codes que nous appelons ici « **ω-codes** ». Nous venons de voir en deuxième section que, pour parler de « décoder de façon unique » les mots bi-infinis périodiques, il faut préciser ce que l'on entend par l'« unicité du décodage ». Si l'on parle d'unicité de décomposition, il faut que le code soit **circulaire**. Si l'on parle d'unicité de factorisation, il faut que le code soit **précirculaire**. Les notions de code circulaire et de code précirculaire diffèrent, et aucune des deux n'assure l'unicité du décodage des mots bi-infinis quelconques.

L'unicité du décodage des mots bi-infinis à l' aide d'un langage finitaire nous mène donc à définir deux notions différentes de code (finitaire) pour les mots bi-infinis selon que l'on considère les décompositions ou les factorisations. Nous appelons ces codes « Z-codes » et « biω-codes ».

La notion de **biω-code**, moins restrictive, est mieux appropriée aux mots bi-infinis. On peut en effet la caractériser en termes d'injectivité des morphismes de codage de la même façon que les codes usuels et les ω-codes. Rappelons qu'un morphisme de codage $\varphi : X^* \to C^*$ peut être étendu à X^ω et à ${}^\omega X^\omega$. Cela permet de représenter les C-factorisations par les mots de X^ω ou de ${}^\omega X^\omega$. Le langage C est un code si et seulement si $\varphi : X^* \to C^*$ est injectif ;

Le langage C est un ω-code si et seulement si $\varphi : X^\omega \to C^\omega$ est injectif; Le langage C est un biω-code si et seulement si $\varphi : {}^\omega X^\omega \to {}^\omega C^\omega$ est injectif.

Les **Z-codes** sont quant à eux mieux adaptés au codage des Z-mots. Caractériser les Z-codes se ramène à caractériser les biω-codes, car , de même que les codes circulaires sont les codes précirculaires purs, nous montrons que les Z-codes sont les biω-codes purs.

Les biω-codes sont des codes précirculaires particuliers. Nous donnons des conditions nécessaires et suffisantes pour qu'un code précirculaire C soit un biω-code. Ces conditions exigent que certains langages infinitaires soient monogènes.

3.1 Notions de ω-code, de biω-code et de Z-code

Définition 3.1.1 *[61] Soit $C \subset A^+$ et $\varphi : X^* \to C^*$ un morphisme de codage pour C. On dit que C est un* **ω-code** *s'il vérifie l'une des propriétés (équivalentes) suivantes:*

- pour tous u, v de C on a:

$$uC^\omega \cap vC^\omega \neq \emptyset \Rightarrow u = v \ ,$$

- tout mot de C^ω a une seule C-factorisation,
- $\varphi : X^\omega \to C^\omega$ est injective,
- $C^{-1}C \cap C^\omega (C^\omega)^{-1} = \{\varepsilon\}$.

Tout sous-ensemble d'un ω-code est un ω-code; un singleton $\{u\}$ est un ω-code si $u \neq \varepsilon$; tout code préfixe est un ω-code; $ab(a^2b)^*$, $ab^*c \cup \{a, b\}$ sont des ω-codes; $\{a, ab, b^2\}$ n'est pas un ω-code .

La définition que nous avons donné aux C-factorisations des mots bi-infinis nous mène à la notion suivante de biω-code:

Définition 3.1.2 *[24] Soit $C \subset A^+$ et $\varphi : X^* \to C^*$ un morphisme de codage pour C. On dit que C est un* **biω-code** *s'il vérifie l'une des propriétés (équivalentes) suivantes:*

- tout mot de ${}^\omega C^\omega$ a une seule C- factorisation,
- $\varphi : {}^\omega X^\omega \to {}^\omega C^\omega$ est injective.

Tout sous-ensemble d'un biω-code est un biω-code; un singleton $\{u\}$ est un biω-code si $u \neq \varepsilon$; $\{a, bab\}$, $\{ba^3, ba^2ba, a^2ba^2\}$, $\{b^2\} \cup ab(a^2b)^*$ sont des biω-codes. Si deux mots distincts ont des puissances conjuguées, ils ne peuvent faire partie d'un même biω-code. Par exemple, le code {ab, ba} n'est pas un biω-code.

D'après le théorème 2.2.2 relatif aux factorisations des mots bi-infinis périodiques sur un code précirculaire, on a:

Proposition 3.1.1 *Tout biω-code est un code précirculaire.*

La réciproque est fausse. Par exemple, le code $C = \{a^2, ab, b^2\}$ est précirculaire mais ce n'est pas un biω-code: le mot ${}^{\omega}ab^{\omega}$ a deux C- factorisations.

La notion de C-décomposition d'un Z-mot nous mène à la notion suivante de Z-code:

Définition 3.1.3 *[24] [25] Soit $C \subset A^+$. On dit que C est un* **Z-code** *si tout Z-mot a au plus une C-décomposition.*

D'après le lemme 0.2.1, C est un Z-code si et seulement si tout mot bi-infini a un représentant qui possède au plus une C-décomposition.

La définition des Z-codes ne peut pas être immédiatement traduite en termes de morphismes de codage. Cela sera fait un peu plus loin (corollaire 3.2.5).

Tout sous-ensemble d'un Z-code est un Z-code; tout élément d'un Z-code est primitif; un singleton $\{u\}$ est un Z-code si u est un mot primitif; deux mots primitifs conjugués ne peuvent faire partie d'un même Z-code; les codes $\{a, bab\}$, $\{ba^3, ba^2ba, a^2ba^2\}$ ne sont pas des Z-codes (il suffit de considérer des représentants de ${}^{\omega}(ab)^{\omega}$ et ${}^{\omega}(ba^3ba^2)^{\omega}$); $\{b^n\} \cup ab(a^2b)^*$ est un Z-code si et seulement si $n = 1$.

D'après le théorème 2.3.4 concernant les codes circulaires, on a:

Proposition 3.1.2 *Tout Z-code est un code circulaire.*

La réciproque est fausse. Par exemple, le code $C = \{ab^nab^{n+1} \mid n \geq 1\} \cup \{ab\}$ est un code circulaire, mais ce n'est pas un Z-code: les représentants du mot ${}^{\omega}(ab)ab^2ab^3ab^4...$ admettent deux C-décompositions.

Ces différentes notions de codes peuvent être comparées:

Proposition 3.1.3 *: Considérons un langage $C \subset A^+$ et les propriétés: (a): le langage C est un Z-code, (b): le langage C est un biω-code, (c): le langage C est un ω-code, (d): le langage C est un code. On a: $(a) \Rightarrow (b) \Rightarrow (c) \Rightarrow (d)$ et les implications contraires sont fausses.*

Preuve: $(a) \Rightarrow (b)$: Immédiat d'après les définitions.
$(c) \Rightarrow (d)$: Soit C un ω-code et $u, v \in C$ vérifiant $uC^* \cap vC^* \neq \emptyset$. Alors $uC^\omega \cap vC^\omega \neq \emptyset$ et $u = v$.
$(b) \Rightarrow (c)$: Soit C un biω-code.
- Si C est un singleton, c'est un ω-code.
- Si C est un code à deux éléments, c'est un ω-code (voir une preuve dans le chapitre 7).
- Si C a deux éléments: $C = \{u, v\}$ et n'est pas un code, $uv = vu$ et le mot ${}^\omega uv^\omega = {}^\omega uvuv^\omega$ a deux C-factorisations.
- Si C n'est pas un ω-code et a au moins trois éléments, considérons $u_0, u_1, u_2, ..., v_0, v_1, v_2... \in C$ tels que $u_0 \neq v_0$ et $u_0u_1u_2... = v_0v_1v_2...$ et choisissons w dans $C - \{u_0, v_0\}$. Le mot ${}^\omega wu_0u_1u_2...(= {}^\omega wv_0v_1v_2...)$ a deux C-factorisations distinctes, donc C n'est pas un biω-code. •

Les exemples suivants montrent que ces notions de codes sont distinctes.
- Le singleton $\{aa\}$ est un biω-code mais pas un Z-code.
- Le langage $\{ab, ba\}$ est un ω-code mais pas un biω-code.
- Le langage $\{a, ab, b^2\}$ est un code mais pas un ω-code.

3.2 Caractérisations des Z-codes

Nous donnons d'abord une caractérisation simple des Z-codes, qui aura comme conséquence immédiate la décidabilité dans le cas rationnel.

Proposition 3.2.1 *Soit C un langage $\subset A^+$, C est un Z-code si et seulement si*

$$(({}^\omega C)^{-1}({}^\omega C) \cap A^+).((C^\omega)(C^\omega)^{-1} \cap A^+) \cap C = \emptyset \ .$$

Dans la suite, nous précisons la relation existant entre les notions de biω-code et de Z-code. Dans ce but, rappelons que nous avons défini pour tout mot $w \in A^Z$ une application surjective τ_w qui associe à une C-décomposition d de w la C-factorisation représentée par $(w[d_n, d_{n+1}[)_{n \in Z}$. Clairement, un bi$\omega$-code est un Z-code si et seulement si pour tout mot $w \in A^Z$, τ_w est injective, c'est-à-dire qu'aucune C-factorisation n'est chevauchante.

Rappelons qu'une factorisation chevauchante factorise toujours un mot (bi-infini) périodique (lemme 0.2.1) et donc qu'une C-factorisation chevauchante est périodique quand C est un code (proposition 1.2.5). Le lemme suivant précise la structure des factorisations périodiques chevauchantes, et peut servir à démontrer le théorème 3.2.4.

Lemme 3.2.2 *: Considérons un langage C, $\varphi : X^* \to C^*$ un morphisme de codage pour C, un mot primitif y de X^* et la C-factorisation périodique $c = {}^\omega y^\omega \in {}^\omega X^\omega$. La factorisation c est chevauchante si et seulement si $\varphi(y)$ est imprimitif.*

Preuve: Considérons une C-factorisation $c = {}^{\omega}y^{\omega}$ chevauchante et w un représentant de ${}^{\omega}\varphi(y)^{\omega}$. Le Z-mot w a deux C-décompositions distinctes d et d' appartenant à $\tau^{-1}(c)$. Il existe q tel que $u[d_k, d_{k+1}[= u[d'_{q+k}, d'_{q+k+1}[$ pour tout k, et u est périodique de période $u[d_0, d'_q[$ (si $d_0 < d'_q$, ce qui peut toujours être réalisé en échangeant éventuellement d et d').
Puisque c est périodique de période y, le découpage de w par d' est périodique de période $\varphi(y)$. On peut donc supposer que $d_0 \leq d'_q < d_0 + |\varphi(y)|$. D'autre part $d'_q \neq d_0$ sinon $d = d'$. Alors w, qui a $\varphi(y)$ pour période, a aussi la période $w[d_0, d'_q[$ de longueur strictement plus petite que $|\varphi(y)|$. Donc $\varphi(y)$ est imprimitif.

Inversement, supposons que $\varphi(y) = u^n$ ($n \geq 1$ et $u \in A^+$). Alors $c = {}^{\omega}y^{\omega}$ factorise tout représentant w de ${}^{\omega}u^{\omega}$. Considérons une C-décomposition d'un représentant w de ${}^{\omega}u^{\omega}$: d, associée à c. Posons $y = y_0y_1...y_k$, où les y_i appartiennent à X. On peut choisir w tel que la C-décomposition d vérifie $\varphi(y_i) = w[d_id_{i+1}[$ pour tout i de 0 à k. A partir des n suites strictement croissantes : $(d_i + p|u|)_{i\in Z}$ $(p = 0, ..., n-1)$, on peut considérer n C-décompositions de w, associées à c. Si c n'est pas chevauchante, ces décompositions coïncident. Donc $u \in C^+$ et il existe z appartenant à X^+ tel que $\varphi(z) = u$, de plus $y = z^n$ car, par translation, la C-factorisation des u successifs est la même (même si C n'est pas un code). Alors $n = 1$ puisque y est primitif. •

Proposition 3.2.3 *Soit C un code pur. L'ensemble Y des mots bi-infinis dont les représentants ont plusieurs C-décompositions et une seule C-factorisation est vide.*

Preuve: Soit C un code tel que Y ne soit pas vide. Les éléments de Y sont périodiques (lemme 0.2.1), et leur factorisation aussi (proposition 1.2.5). Cette factorisation étant chevauchante, d'après le lemme 3.2.2, on voit, en utilisant un morphisme de codage pour C que le code C n'est pas pur. •

La réciproque de la proposition précédente est fausse. Malgré cela, nous pourrons quand même démontrer le théorème suivant.

Exemple: Considérons le code $C = \{abca, bc, cab\}$. Seuls les représentants du mot ${}^{\omega}(abc)^{\omega}$ ont plusieurs C-décompositions. Mais ce mot a deux C-factorisations, donc Y est vide. Le code C n'est pas pur car $(abc)^2 \in C^*$, alors que $abc \notin C^*$.

La caractérisation des Z-codes que nous donnons maintenant est plutôt algébrique et peut s'exprimer en termes de morphismes de codage , ce qui la rend d'utilisation commode (par exemple dans l'étude de la composition des Z-codes).

Théorème 3.2.4 *Soit C un langage inclus dans A^+, C est un Z-code si et seulement si C est un biω-code pur.*

Preuve : Un Z-code est un biω-code et aussi un code circulaire, donc un biω-code pur. Inversement, considérons un biω-code pur C, il s'agit de montrer qu'aucune C-factorisation n'est chevauchante. D'après le lemme 0.2.1, il suffit de considérer les mots bi-infinis périodiques. C est un code pur et précirculaire donc un code circulaire (proposition 2.3.3) ; donc tout mot bi-infini périodique n'a qu'une seule C-décomposition (théorème 2.3.4). •

Corollaire 3.2.5 *Soit $C \subset A^+$ et $\varphi : X^* \to C^*$ un morphisme de codage pour C. C est un Z-code si et seulement si $\varphi : {}^{\omega}X^{\omega} \to {}^{\omega}C^{\omega}$ est injective et vérifie :*

$$\forall y \in X^+ \ (\ y \textit{ primitif } \Rightarrow \varphi(y) \textit{ primitif}\ ) .$$

3.3 Caractérisations des biω-codes

Dans cette section nous donnons des caractérisations des biω-codes, en liaison avec certaines propriétés déjà obtenues des biω-codes. Tout biω-code C est un ω-code précirculaire, et le miroir de C : $\tilde{C}$ est aussi un ω-code (nous dirons que C est un $\tilde{\omega}$-code). Malheureusement, ces conditions ne sont pas suffisantes pour avoir un biω−code, même dans le cas simple d'un code bipréfixe fini. Par exemple, le langage $C = \{aa, ab, bb\}$ est un code bipréfixe fini, mais le mot ${}^{\omega}ab^{\omega}$ a deux C-factorisations.

Remarquons que la classe des codes précirculaires, celle des ω-codes et celle des $\tilde{\omega}$-codes ne sont pas comparables :
- le langage $\{ab, ba\}$ est un ω-code (bipréfixe) et un $\tilde{\omega}$-code, mais il n'est pas précirculaire .
- le langage $\{c, b^2, cb\}$ est un $\tilde{\omega}$-code précirculaire, mais n'est pas un ω-code.

Dans les caractérisations qui suivent nous utiliserons la notion de langage infinitaire monogène : un langage infinitaire (ou bi-infinitaire) est dit **monogène** s'il est de la forme u^{ω}, ${}^{\omega}u$ ou ${}^{\omega}u^{\omega}$, pour un mot u de A^+.

Théorème 3.3.1 *Soit C un code précirculaire inclus dans A^+. C est un biω-code si et seulement si C vérifie :*
(2) $\forall s, t \in A^+$

$$st \in C \textit{ et } ({}^{\omega}C \cap {}^{\omega}Cs)(C^{\omega} \cap tC^{\omega}) \neq \emptyset \Rightarrow \exists u \in A^+ \ ({}^{\omega}C \cap {}^{\omega}Cs)(C^{\omega} \cap tC^{\omega}) = {}^{\omega}u^{\omega} .$$

Notation : Par commodité, dans la suite , nous noterons : $M_s = {}^{\omega}C \cap {}^{\omega}Cs$ et $M'_s = C^{\omega} \cap sC^{\omega}$ (pour $s \in A^+$).

Preuve : Si C est un biω-code, c'est un code précirculaire. Considérons $v \in M_s$ et $w \in M'_t$ où $st \in C$. Les représentants du mot bi-infini vw ont deux C-décompositions distinctes. Si C est un biω-code, ces décompositions ont une même factorisation associée (chevauchante) et le mot vw est périodique. Il existe donc u tel que $v = {}^\omega u$ et $w = u^\omega$. Pour tout $v' \in M_s$, $v'w$ est périodique et donc $v' = {}^\omega u$. Par conséquent, $M_s = {}^\omega u$, et de même $M'_t = u^\omega$. Réciproquement, si un Z-mot représentant d'un mot bi-infini x a deux C-décompositions distinctes, il existe s et $t \in A^+$ tels que $x \in M_s.M'_t$ et $st \in C$. Par la propriété (2), x est périodique. Si C est un code précirculaire, x n'a qu'une C-factorisation (théorème 2.2.2). Donc tout mot a au plus une C-factorisation. •

Dans cette caractérisation :
- on ne peut omettre la condition (2). Par exemple, le code $C = \{a^2, ab, b^2\}$ est précirculaire mais n'est pas un biω-code. On a $M_a = {}^\omega a$ et $M'_b = b^\omega$.
- on ne peut remplacer la condition « code précirculaire » par la condition moins forte « code presque circulaire ». En effet, le langage $C = \{ab, c, bca\}$ est un code presque circulaire non précirculaire (donc C n'est pas un biω-code). Cependant, on a : $M_a = {}^\omega(bca)$ et $M'_b = (bca)^\omega$, $M_{bc} = {}^\omega(abc)$ et $M'_a = (abc)^\omega$, $M_b = {}^\omega(cab)$ et $M'_{ca} = (cab)^\omega$.

Pour affaiblir la condition (2) de la caractérisation précédente, nous utiliserons la notion de code presque circulaire qui a déjà été utilisée en section 2.4.

Lemme 3.3.2 : *Un code C vérifiant la propriété :*

(2') $\forall s, t \in A^+$ ($st \in C$ *et* $M_s.M'_t \neq \emptyset \Rightarrow M_s$ *et* M'_t *sont monogènes*)

est un code presque circulaire et vérifie la propriété (2).

Preuve : Remarquons d'abord que la propriété (2') nous assure que :

$$tC^*s \cap C^* \neq \emptyset,\ s,t \in A^+,\ st \in C \ \Rightarrow\ M_s.M'_t \text{ est monogène .}$$

En effet, considérons $w \in tC^*s \cap C^*$ (où $s, t \in A^+$ et $st \in C$). Le mot ${}^\omega w$ appartient à M_s . Mais $M_s = {}^\omega u$, où l'on peut supposer que u est primitif. Alors w appartient à u^*. De même, w appartient à v^* si $M'_t = v^\omega$ où v est primitif. Ainsi $u = v$ et $M_s.M'_t = {}^\omega u^\omega$. De plus, $tC^*s \cap C^* \subset u^*$, et donc, si C est un code, $tC^*s \cap C^*$ est monogène (lemme 2.4.3). Donc un code vérifiant (2') est un code presque circulaire.

Montrons maintenant la propriété (2). Considérons $s, t, u, v \in A^+$ tels que $st \in C$, $M_s = {}^\omega u$, $M'_t = v^\omega$, u et v primitifs. Il s'agit de prouver que $u = v$. Soit $\varphi : X^* \to C^*$ un morphisme de codage pour C. Considérons z (resp z')$\in X^\omega$

une C-factorisation de v^ω (resp $t^{-1}v^\omega$). Puisque C est un code, d'après la proposition 1.2.5, z et z' sont ultimement périodiques ($z = z_1 z_2^\omega, z' = z'_1 z'^\omega_2$). Il existe une portion droite du mot v^ω sur laquelle z et z' réalisent un découpage périodique. D'après les lemmes de la section 1.2, il existe des conjugués de v: v_1 et v_2 tels que $\varphi(z_2) = v_1^n, \varphi(z'_2) = v_2^{n'}$ pour certains n et n'. Il existe donc $p > 0$, $t' \in A^+$, $s' \in A^*$ tels que $s't' \in C$ et $v_1^p \in t'C^*s' \cap C^*$. On démontre de même qu'il existe un conjugué u_1 de u, $q > 0$, $t'' \in A^+$, $s'' \in A^*$ tels que $s''t'' \in C$ et $u_1^q \in t''C^*s'' \cap C^*$.

Plusieurs cas se présentent:
- $s' \neq \varepsilon$. Le mot ${}^\omega uv^\omega$ appartient à $M_{s'}.M'_{t'}$, langage qui est monogène d'après la remarque préliminaire. Donc $M_s.M'_t$ est monogène et $u = v$.
- $s'' \neq \varepsilon$. De même que précédemment le langage $M_s.M'_t$ est monogène et $u = v$.
- $s' = s'' = \varepsilon$. On est dans le cas où $z = z_3 z_2^\omega$, $z' = z'_3 z_2^\omega$ et $\varphi(z_3) = t\varphi(z'_3)$. De même, si s'' et t'' ont été construits à partir de deux C-factorisations y et y', on a: $y = {}^\omega y_2 y_3, y' = {}^\omega y_2 y'_3$ où $\varphi(y_3) = \varphi(y'_3)s$. Si C est un code, on a une contradiction puisque st appartient à C. Ce cas ne se présente pas. •

Remarques:
- Il existe des codes presque circulaires qui ne vérifient pas (2'). Par exemple le code précirculaire $C = \{aa, ab, bb\}$ est tel que $M'_a = a^*b^\omega + a^\omega$.
- Les conditions (2) et (2') ne suffisent pas pour avoir un code. Par exemple, $C = \{a^2, a^3\}$ n'est pas un code (et C^* n'est pas libre) et cependant $M_a = M_{a^2} = {}^\omega a$, $M'_a = M'_{a^2} = a^\omega$.

On déduit aisément de la proposition 3.3.1 et du lemme 3.3.2 la caractérisation suivante:

Théorème 3.3.3 *Un code précirculaire $C \subset A^+$ est un biω-code si et seulement si C vérifie la propriété:*

(2') $\forall s, t \in A^+$ ($st \in C$ *et* $M_s.M'_t \neq \emptyset \Rightarrow M_s$ *et* M'_t *sont monogènes*) .

Lemme 3.3.4 *Tout code C qui vérifie la propriété (2'), vérifie aussi la propriété:*

(3) $\forall s, t, t' \in A^+$ ($st \in C, st' \in C, M_s, M'_t$ *et* $M'_{t'} \neq \emptyset \Rightarrow t = t'$) .

Preuve: Considérons $s, t, t' \in A^+$ tels que $st \in C, st' \in C, M_s, M'_t$ et $M'_{t'} \neq \emptyset$. Le code C, vérifiant la propriété (2') vérifie la propriété (2). Il existe donc $u \in A^+$ tel que $M_s = {}^\omega u$ et $M'_t = M'_{t'} = u^\omega$. Soit $\varphi : X^* \to C^*$ un morphisme de codage pour C. Considérons $z \in {}^\omega X$, $y, y' \in X^\omega$ tels que $M_s = \varphi(z)s$ et $M'_t = t\varphi(y) = M'_{t'} = t'\varphi(y')$. Puisque C est un code, les deux C-factorisations

$z\varphi^{-1}(st)y$ et $z\varphi^{-1}(st')y'$ de ${}^{\omega}u^{\omega}$ sont périodiques. Elles commencent toutes deux par le même mot z, elles sont donc égales et $st = st'$ et $t = t'$. •

Corollaire 3.3.5 *Tout biω-code vérifie la propriété (3).*

La propriété (2') entraîne la propriété suivante:

(4) $\forall s, t \in A^+$ ($st \in C$ et $M_s.M'_t \neq \emptyset \Rightarrow$ M_s et M'_t *singletons*) .

Mais la condition (2') du théorème 3.3.3 ne peut être remplacée ni par la condition (3) ni par la condition (4). Par exemple, le langage $C = \{acab\} \cup \{ac^{n+1}ac^n \mid n \geq 1\} \cup \{ab^nab^{n+1} \mid n \geq 1\}$ est un ω-code circulaire mais n'est pas un biω-code. Le mot $...ac^4ac^3ac^2acabab^2ab^3ab^4...$ est le seul mot qui a deux C- factorisations. Tout (M_s, M'_t) tel que $st \in C$, M_s et $M'_t \neq \emptyset$ est formé de deux singletons. Ce code C vérifie les propriétés (3) et (4).

Chapitre 4

Classification des codes et factorisations des mots infinis

Nous venons d'étudier les codes qui assurent l'unicité du décodage des mots infinis : les ω-codes, les biω-codes et les Z-codes. D'autre part, nous avons vu que les codes usuels sont intéressants des points de vue suivants:
- certains mots infinis se factorisent de façon unique en mots du code ;
- la forme des factorisations en mots du code de certains mots infinis (les mots périodiques ou ultimement périodiques) est particulière (périodique ou ultimement périodique).

Cela a permis de caractériser les codes en termes de mots infinis. Ces caractérisations seront beaucoup utilisées dans cette partie où nous regardons les questions précédentes d'une autre façon. Nous déterminons les codes qui assurent l'unicité du décodage des mots d'un type particulier (périodique ou ultimement périodique), et ceux qui donnent aux factorisations des mots périodiques la forme périodique.

Nous avons déjà résolu en partie le premier problème en étudiant les codes circulaires et en introduisant les codes précirculaires. Nous achevons ici ce travail en étudiant d'autres codes.

Si l'on exige l'unicité de la factorisation des mots infinis ultimement périodiques, on obtient une caractérisation en termes de mots finis qui n'est autre que la définition de codes déjà étudiés : les codes **faiblement préfixes**. Si on exige l'unicité de la factorisation des mots infinis ultimement périodiques, on définit de nouveaux codes : les **π-codes**. Il en est de même si on exige l'unicité de la factorisation des mots bi-infinis ultimement périodiques, on définit alors les codes **fortement précirculaires**.

Dans la présentation, nous définissons d'abord ces codes en termes de mots finis, puis en termes de morphismes de codage, enfin nous les caractérisons en termes de mots infinis.

En ce qui concerne la forme des factorisations des mots ultimement pério-

diques, nous savons déjà qu'elle est ultimement périodique si l'on factorise sur un code. Dans les mêmes conditions, elle est périodique pour les mots bi-infinis périodiques. Si on exige que les factorisations des mots infinis périodiques soient périodiques, le code doit être **suffixe**; en fait, cette propriété caractérise les codes suffixes.

Les divers résultats concernant les codes étudiés et le nombre de factorisations des mots infinis sont exprimés dans des tableaux récapitulatifs. Un exemple de code circulaire montre qu'il peut exister des mots infinis ayant une infinité non dénombrable de factorisations, même si il y a unicité de la factorisation pour tous les mots ultimement périodiques.

Entre ces différentes familles de codes existent des relations d'inclusion. Nous les étudions et présentons un schéma résumant les résultats.

4.1 Définitions de codes particuliers

Les codes particuliers qui interviennent dans cette section sont introduits et étudiés parce qu'ils peuvent être caractérisés en termes de mots infinis. Certains viennent d'être étudiés, ce sont les codes circulaires et les codes précirculaires; d'autres ont été définis et étudiés par ailleurs, ce sont les codes faiblement préfixes définis par Capocelli [17]; afin de distinguer les autres nous leur donnons un nom inspiré par leurs propriétés.

Définition 4.1.1 *Soit $C \subset A^+$. C est un langage* **faiblement préfixe** *[17] si et seulement si C est la base du monoïde C^* vérifiant:*

$$\forall\, u, v, w \in A^* \; (w, wu, uv, vu \in C^* \;\Rightarrow\; u \in C^*)\,.$$

Proposition 4.1.1 *[17] Les langages faiblement préfixes sont des codes. Les* **codes préfixes** *et les* **codes circulaires** *sont des codes faiblement préfixes.*

Preuve: Soit C un langage faiblement préfixe, C est un code car on peut montrer que C^* est stable [8]. En effet, si w, wu, uv', v' appartiennent à C^*, les mots $w, wu, uv'w, v'wu$ appartiennent aussi à C^*. Si C est faiblement préfixe, u appartient à C^*.

Si un code C est préfixe, un mot u appartient à C^* dès que w et wu appartiennent à C^* donc tout code préfixe est faiblement préfixe.

Si un code C est circulaire, C^* est très pur, autrement dit:

$$\forall u, v \in A^* \; (uv, vu \in C^* \;\Rightarrow\; u, v \in C^*).$$

Donc C est faiblement préfixe. •

Les implications inverses sont fausses. Par exemple, $\{ab, ba\}$ est un code préfixe (donc faiblement préfixe) non circulaire ; $\{a, ab\}$ est un code circulaire (donc faiblement préfixe) non préfixe.

Définitions 4.1.2 *Soit $C \subset A^+$.*
- C est un **π-code** *[22] si et seulement si C est un code vérifiant :*

$$\forall\, u, v, w, \beta \in A^* \text{ tels que } wuvu < \beta^\omega \text{ et } |u| \geq |\beta|$$

$$(w, wu, uv, vu \in C^* \Rightarrow u \in C^*)\,.$$

- C est un **code fortement précirculaire** *si et seulement si C est un code précirculaire qui vérifie :*
pour tous mots primitifs u et u' de A^+, pour tous mots v et v' de C^+, pour tous n et $p > 0$ tels que $u^+ \cap C^+ = (u^n)^+$ et $u'^+ \cap C^+ = (u'^p)^+$, on a :

$$(\, v' = u'^j v u^i \;\Rightarrow\; \exists\, j_0 \in p \cdot N \;\; \exists\, i_0 \in n \cdot N \quad v' = u'^{j_0} v u^{i_0} \,)\,.$$

Dans la définition des π-codes, il est clair qu'on peut se restreindre à des β primitifs. Par contre, on ne peut supprimer la condition « être un code ». Par exemple, le langage $C = \{ba, b, abc, bc\}$ n'est pas un code mais l'autre condition est satisfaite.

D'après les définitions, on a :

Proposition 4.1.2 *Tout code faiblement préfixe est un π-code.*

Nous verrons en fin de section 4.3 que la réciproque est fausse et qu'il existe des codes qui ne sont pas des π-codes.

Nous verrons en section 4.2 qu'on ne peut pas supprimer la condition « être un code précirculaire » dans la définition des codes fortement précirculaires.

4.2 Caractérisations en termes de morphismes de codage

Les codes définis précédemment peuvent être caractérisés en termes de morphismes de codage. Ces caractérisations peuvent n'être qu'une traduction des définitions, c'est alors peu intéressant. C'est le cas des codes faiblement préfixes et des π-codes. C'est aussi le cas des codes fortement précirculaires, mais pour ceux-ci nous énonçons le résultat car cela servira dans la suite. Pour

les codes précirculaires, nous avons déjà donné cette caractérisation en section 2.1 .

Proposition 4.2.1 (Rappel 2.1.1) *Soit $C \subset A^+$ et $\varphi : X^* \to C^*$ un morphisme de codage pour C. Le langage C est un code précirculaire si et seulement si*

$$\forall\, x, x' \in X^+ \ (\ \varphi(x) \text{ et } \varphi(x') \text{ conjugués } \Rightarrow x \text{ et } x' \text{ conjugués }) \ .$$

Lemme 4.2.2 *Soit C un code inclus dans A^+ et $\varphi : X^* \to C^*$ un morphisme de codage pour C. Les propositions suivantes sont équivalentes :*
(1) : pour tous mots primitifs x et x' de X^+, pour tous mots y et y' de X^+, on a :

$$\varphi(y') = \sqrt{(\varphi(x'))}^j \varphi(y) \sqrt{(\varphi(x))}^i \ \Rightarrow \ y' \in x'^* y x^* \ .$$

(2) : pour tous mots primitifs u et u' de A^+, pour tous mots v et v' de C^+, pour tous n et $p > 0$ tels que $u^+ \cap C^+ = (u^n)^+$ et $u'^+ \cap C^+ = (u'^p)^+$, on a :

$$(\ v' = u'^j v u^i \ \Rightarrow \ \exists\, j_0 \in p \cdot N \ \ \exists\, i_0 \in n \cdot N \quad v' = u'^{j_0} v u^{i_0}\ ) \ .$$

Preuve :
- si la propriété (1) est vérifiée, considérons $u, u' \in A^+$ primitifs, $v, v' \in C^+$, $n, p > 0$ tels que $u^+ \cap C^+ = (u^n)^+$ et $u'^+ \cap C^+ = (u'^p)^+$. Il existe $y, y' \in X^+$, $x, x' \in X^+$ primitifs tels que $v' = \varphi(y'), v = \varphi(y), u'^p = \varphi(x'), u^p = \varphi(x)$. Si $v' = u'^j v u^i$, d'après (1), $y' \in x'^* y x^*$. Donc $\varphi(y') \in \varphi(x'^* y x^*)$. Donc v' est de la forme $u'^{j_0} v u^{i_0}$ où $j_0 \in p \cdot N$, $i_0 \in n \cdot N$.
- si la propriété (2) est vérifiée, considérons $x, x' \in X^+$ primitifs, $y, y' \in X^+$ tels que $\varphi(y') = \sqrt{(\varphi(x'))}^j \varphi(y) \sqrt{(\varphi(x))}^i$. Posons $v' = \varphi(y')$, $v = \varphi(y)$, $u' = \sqrt{(\varphi(x'))}$, $u = \sqrt{(\varphi(x))}$. On a : $v' = u'^j v u^i$ et, d'après (2), $v' = u'^{j_0} v u^{i_0}$ où $j_0 \in p \cdot N$ et $i_0 \in n \cdot N$. Donc $\varphi(y') \in \varphi(x'^* y x^*)$ et, puisque C est un code, $y' \in x'^* y x^*$. •

On déduit aisément de ce lemme que :

Proposition 4.2.3 *Soit $C \subset A^+$ et $\varphi : X^* \to C^*$ un morphisme de codage pour C. Le langage C est un code fortement précirculaire si et seulement si C est un code précirculaire qui vérifie :*
(1) : pour tous mots primitifs x et x' de X^+, pour tous mots y et y' de X^+, on a :

$$\varphi(y') = \sqrt{(\varphi(x'))}^j \varphi(y) \sqrt{(\varphi(x))}^i \ \Rightarrow \ y' \in x'^* y x^* \ .$$

On peut remarquer que la propriété (1) requise dans la proposition précédente ne suffit pas pour qu'un code soit précirculaire. Il existe en effet des codes purs non précirculaires (le code $\{ab, ba\}$ par exemple), et la propriété

(1) est vérifiée par tout code pur . En effet si φ est un morphisme de codage d'un code pur, $\varphi(x)$ est primitif dès que x est primitif, donc la propriété (1) s'exprime :

$$\varphi(y') = \varphi(x')^j \varphi(y)\varphi(x)^i \Rightarrow y' \in x'^* y x^*$$

ce qui est vérifié puisque φ est injectif.

Dans la définition des codes fortement précirculaires on ne peut donc pas supprimer la condition « être un code précirculaire ».

4.3 Caractérisations en termes de mots infinis

La proposition suivante caractérise les codes faiblement préfixes en termes de mots infinis.

Proposition 4.3.1 *Soit C un langage inclus dans A^+. Les propriétés suivantes sont équivalentes :*
1) le langage C est un code faiblement préfixe,
2) pour tous $u, v \in C^+$, uv^ω a une C-factorisation unique,
3) tout mot infini ultimement périodique a au plus une C-factorisation.

Preuve :
$1 \Rightarrow 2$: Supposons que u, u', v, v' appartenant à C^+ vérifient $uv^\omega = u'v'^\omega$ et $|u| \leq |u'|$. Si les deux C-factorisations sont distinctes, on peut supposer que $u = u_1 ... u_n$ et $u' = u'_1 ... u'_p$ où $u_1 \neq u'_1$. Si $u' = uw$ où $w \in C^*$, C n'est pas un code. On peut donc supposer que $u' = uw$ où $w \notin C^*$. En prenant des puissances appropriées de v et v', on peut supposer que $|v| = |v'| > |w|$. Alors $v = ww'$ et $v' = w'w$ pour un certain mot w'. Nous avons : $u, uw, ww', w'w \in C^*$ mais $w \notin C^*$, ce qui contredit le fait que C soit faiblement préfixe.
$2 \Rightarrow 1$: Si C n'est pas faiblement préfixe, il existe u, v, w tels que $u \notin C^*, w, wu$, $uv, vu \in C^*$. Alors le mot $w(uv)^\omega$ a deux C-factorisations distinctes.
$3 \Rightarrow 2$: Clair.
$2 \Rightarrow 3$: Clair d'après le lemme 1.2.2. •

On en déduit le corollaire :

Corollaire 4.3.2 *Tout ω-code est un code faiblement préfixe.*

La réciproque est fausse. Le code $\{ab\} \cup \{ab^n ab^{n+1} \mid n \geq 1\}$ est circulaire, donc faiblement préfixe, mais n'est pas un ω-code.

Etudions de même les π-codes.

Proposition 4.3.3 *Soit C un langage inclus dans A^+. C est un π-code si et seulement si tout mot infini périodique a au plus une C-factorisation.*

Preuve: Soit C un π-code, et β^ω un mot périodique ayant deux C-factorisations distinctes: $(w_1, w_2, ...)$ et $(w'_1, w'_2, ...)$. On peut supposer que $w_1 \neq w'_1$. Pour tout i, considérons (quand il existe) p_i tel que $w_1...w_{p_i-1} < \beta^{i-1} \leq w_1...w_{p_i} \leq \beta^i$. Il existe un mot α et une infinité de i tels que $w_1...w_{p_i} = \beta^{i-1}\alpha$. Les indices m et n seront de tels indices p_i. De même, il existe un mot α' et une infinité de i pour lesquels un indice qi vérifie $w'_1...w'_{q_i-1} < \beta^{i-1} \leq w'_1...w'_{q_i} \leq \beta^i$ et $w'_1...w'_{q_i} = \beta^{i-1}\alpha'$. Les indices m' et n' seront de tels indices q_i.

Choisissons m, m', n, n' tels que:
$w_1...w_m < w'_1...w'_{m'} < w_1...w_n < w'_1...w'_{n'}$. Posons $w = w_1...w_m, wu = w'_1...w'_{m'}, wuy = w_1...w_n, wuyz = w'_1...w'_{n'}$. On peut choisir m' tel que $|u| \geq |\beta|$. On a: $uy = \beta'^p \in C^+$ et $yz = \beta''^q \in C^+$, où β' et β'' sont des conjugués de β. Posons $v = y(uy)^{q-1}$. On a: $uv \in C^+$ et $vu = (yz)^p \in C^+$. Comme C est un π-code et w, wu, uv, vu appartiennent à C^*, u appartient à C^*. Cela contredit le fait que C est un code et que $w_1 \neq w'_1$.

Inversement, considérons u, v, w, β tels que $wuvu < \beta^*$, $|u| \geq |\beta|$ et w, wu, $uv, vu \in C^*$. On peut supposer que β est primitif, alors u a une seule interprétation sur β: $u = \beta'\beta^i\beta''$ pour un certain $i \geq 0$, β' (resp: β'') un facteur droit (resp: gauche) de β. Alors $uv = \beta'\beta^j\beta\beta'^{-1}$ et $vu = \beta''^{-1}\beta\beta^j\beta''$ pour un certain j. Il s'ensuit que $\beta^\omega = w(uv)^\omega = wu(vu)^\omega$. Si tout mot infini périodique a au plus une C-factorisation, u appartient à C^*, donc C est un π-code. •

Nous savons que tout code faiblement préfixe est un π-code. La proposition inverse est fausse. Le code $C = \{ab, aba, ba^2\}$ est un π-code qui n'est pas faiblement préfixe: seul le mot $ab(aba)^\omega$ a deux C-factorisations ne commençant pas par le même mot. Notons que ce code est un code suffixe.

Il existe des codes qui ne sont pas des π-codes. Par exemple, le code $C = \{a, a^2ba, aba^2ba\}$ n'est pas un π-code car le mot $(aab)^\omega$ a deux C-factorisations.

4.4 Caractérisations en termes de mots bi-infinis

La majoration du nombre de C-factorisations des mots bi-infinis ultimement périodiques donnée dans la preuve du corollaire 1.3.2 permet de retrouver une caractérisation des codes précirculaires et de caractériser les codes

fortement précirculaires.

Proposition 4.4.1 (Rappel 2.2.2) *Un langage $C \subset A^+$ est un code précirculaire si et seulement si tout mot bi-infini périodique a au plus une C-factorisation.*

Preuve : Les notations sont celles de la preuve du théorème 1.3.1 .
Pour tout u primitif de A^+ , ${}^\omega u^\omega$ a au plus une C-factorisation si et seulement si il existe au plus une $\simeq$-classe dans U_u ; c'est-à-dire si tous les mots primitifs x et x' dont les images par φ ont des racines conjuguées sont eux-mêmes conjugués. On en déduit que pour tout u, ${}^\omega u^\omega$ a au plus une C-factorisation si et seulement si

$$\forall\, x, x' \in X^+ \ (\ \varphi(x) \text{ et } \varphi(x') \text{ conjugués } \Rightarrow x \text{ et } x' \text{ conjugués })\ ,$$

ce qui caractérise les codes précirculaires. •

Proposition 4.4.2 *Un langage $C \subset A^+$ est un code fortement précirculaire si et seulement si tout mot bi-infini ultimement périodique a au plus une C-factorisation.*

Preuve : Les notations sont celles de la preuve du théorème 1.3.1. Soit C un code fortement précirculaire, w un mot bi-infini ultimement périodique, w_0 un de ses représentants, deux C-factorisations de w et deux C-décompositions de w_0 associées à ces factorisations. Le code C étant précirculaire, les $\simeq$-classes sont uniques et donc les C-factorisations considérées peuvent s'écrire ${}^\omega x'yx^\omega$ et ${}^\omega x'y'x^\omega$ avec x et x' primitifs. Posons $\varphi(x) = u^n$ et $\varphi(x') = u'^p$, où u et u' sont primitifs. On a $w = {}^\omega u'\varphi(y)u^\omega = {}^\omega u'\varphi(y')u^\omega$, $\varphi(y)$ et $\varphi(y')$ sont des facteurs de w_0. Quitte à allonger y' par des x ou (et) des x', $\varphi(y)$ peut être considéré comme un facteur de $\varphi(y')$ dans w_0, et donc on a $\varphi(y') = u'^i\varphi(y)u^j$ puisque u et u' sont primitifs. D'après l'hypothèse, $y' \in x'^*yx^*$, et donc ${}^\omega x'y'x^\omega = {}^\omega x'yx^\omega$. Le mot w n'a qu'une seule C-factorisation.
Réciproquement, soit C un langage tel que tout mot bi-infini ultimement périodique a au plus une C-factorisation. D'après la proposition précédente, C est un code précirculaire. Soient u et u' appartenant à A^+ et primitifs, soient x et x' primitifs appartenant à X^+ tels que $\varphi(x) = u^n$ et $\varphi(x') = u'^p$, soient y et y' appartenant à X^+ et soient i et j entiers positifs. Supposons que $\varphi(y') = u'^j\varphi(y)u^i$. Considérons le représentant w_0 de $w = {}^\omega u'\varphi(y)u^\omega$ de partie positive $w_+ = \varphi(y)u^\omega$, et sa C-décomposition coïncidant sur w_+ avec la C-factorisation yx^ω et sur w_- avec la C-factorisation ${}^\omega x'$. Il existe d'autre part une C-décomposition de w_0 coïncidant sur u'^jw_+ avec la C-factorisation $y'x^\omega$ et sur $w_-(u'^j)^{-1}$ avec la C-factorisation ${}^\omega x'$. Si $w = \varphi({}^\omega x'yx^\omega) = \varphi({}^\omega x'y'x^\omega)$ est périodique, puisque C est précirculaire on a ${}^\omega x'yx^\omega = {}^\omega x'y'x^\omega = {}^\omega x^\omega$, donc

x et x' sont conjugués : $x' = st, x = ts$ et $y, y' \in sx^*$ et puisque $|\varphi(y')| \geq |\varphi(y)|$, $y' \in yx^* = x'^*yx^*$. Si le mot w ultimement périodique n'est pas périodique, d'après l'hypothèse sur C, w_0 n'a qu'une seule C-décomposition, donc u'^j appartient à C^* et d'après le lemme 0.2.1, $u'^j \in \varphi(x'^*)$; de même $u^i \in \varphi(x^*)$ et par suite, $\varphi(y') \in \varphi(x'^*yx^*)$ et puisque φ est injective, $y' \in x'^*yx^*$. •

On en déduit le corollaire :

Corollaire 4.4.3 *Tout biω-code est un code fortement précirculaire. Tout code circulaire est un code fortement précirculaire.*

Les réciproques sont fausses. Le code $C = \{ab\} \cup \{ab^n ab^{n+1} \mid n > 0\}$ est fortement précirculaire, mais ce n'est pas un biω-code : le mot $^\omega(ab)abab^2ab^3ab^4...$ a deux C-factorisations. Le code $\{a, bab\}$ est fortement précirculaire (c'est un biω-code) mais n'est pas circulaire.

Un code fortement précirculaire est précirculaire, mais le contraire n'est pas vrai : le code précirculaire $C = \{c, abab, cab\}$ n'est pas fortement précirculaire, car le mot $c(ab)^\omega$ a deux C-factorisations.

4.5 Codes suffixes

En ce qui concerne la forme des factorisations, la seule question non encore étudiée est la suivante : quels sont les langages C tels que toute C-factorisation d'un mot infini périodique soit périodique? D'après le théorème (1.2.5), ces langages ne peuvent être que des codes. On a la jolie propriété suivante :

Proposition 4.5.1 *Si C est inclus dans A^+, on a équivalence entre les propriétés :*

- *C est un code suffixe,*
- *toute C-factorisation d'un mot infini périodique est périodique.*

Preuve : Si C n'est pas un code suffixe, il existe $u, v \in C$, $v' \in A^+$ tels que $v = v'u$. Le mot périodique $(uv')^\omega = uv^\omega$ a une C-factorisation non périodique.

Inversement, considérons $\varphi : X^* \to C^*$ un morphisme de codage d'un code suffixe C, u un mot primitif tel que u^ω appartienne à C^ω et une C-factorisation de u^ω. D'après le théorème 1.2.5, cette factorisation peut s'écrire yz^ω. De plus, il existe un conjugué u' de u tel que $\varphi(z) \in u'^+$ (lemme 1.2.1). Le mot u^ω s'écrit aussi vu'^ω où $v < u$, donc $\varphi(y)$ appartient à vu'^+. Le mot $\varphi(y)$ est un facteur droit de u'^+, et donc y est un facteur droit de z^+, puisque le code est suffixe. La factorisation considérée est donc périodique. •

Cette caractérisation des codes suffixes a une conséquence intéressante :

Corollaire 4.5.2 *Tout code suffixe est un π-code.*

Preuve : Avec les notations de la preuve précédente, il existe un mot w de C^+ tel que $u^\omega = w^\omega$, et donc, d'après le théorème 1.1.2, la C-factorisation de u^ω est unique. •

Remarques :
- La réciproque de ce corollaire est fausse : $C = \{a, ba\}$ est un code préfixe, donc un π-code, mais ce n'est pas un code suffixe.
- Les codes suffixes peuvent ne pas être faiblement préfixes. C'est le cas du code $\{c, ca, aba, ba^2\}$.
- On pourrait penser que les codes à délai de déchiffrage vers la gauche borné (cette notion est rappelée en section 6.1) sont aussi des π-codes, il n'en est rien. Le code $C = \{a, ab, cab, bca\}$ est de délai 1, mais le mot $(abc)^\omega$ a deux C-factorisations.

4.6 Tableaux récapitulatifs

Les tableaux de cette sous-section sont relatifs au cardinal de l'ensemble des C-factorisations d'un mot donné. L' exemple suivant montre qu'un cardinal non dénombrable (noté ∞) est possible. Dans cet exemple, le code est circulaire donc à la fois fortement précirculaire et faiblement préfixe, cela nous permet de remplir avec le symbole ∞ toutes les cases des tableaux non encore remplies.

Proposition 4.6.1 *Il existe un code circulaire C sur un alphabet à deux lettres et un mot ayant un ensemble non dénombrable de C-factorisations.*

Exemple : Soit $A = \{u_i \mid i \geq 1\}$ et

$$C = \{u_n ... u_{2n-1} \mid n \geq 1\} \cup \{u_n ... u_{2n} \mid n \geq 1\}.$$

On voit facilement que les mots $w_0 = u_1 u_2 ... u_n ...$ et $^\omega u_1 w_0$ ont une infinité non dénombrable de C-factorizations. Nous allons montrer que C est un **code circulaire**.

Considérons $n_1, m_1, n_2, m_2, p, q \in N$ tels que $n_1 < p \leq m_1$, $u_{n_1} ... u_{m_1}$, $u_{n_2} ... u_{m_2}$ et $u_p ... u_q \in C$ et $u_p ... u_q$ soit facteur de $u_{n_1} ... u_{m_1} u_{n_2} ... u_{m_2}$. On peut démontrer qu'alors $n_2 = m_1 + 1$ et que $n_2 \leq q < m_2$. Tout Z-mot qui admet deux C-décompositions est donc de la forme $v.(u_1 u_2 ... u_n)^{-1} w_0$ et n'est donc pas périodique. Le langage C est donc circulaire. En utilisant la composition des codes circulaires, il est facile d'obtenir sur un alphabet fini un exemple de code

ayant les mêmes propriétés. Considérons l'alphabet $B = \{a, b\}$ et $\varphi : A \to B^+$ defini par: $\varphi(u_i) = a^i b$. Le language $D = \varphi(A)$ est un code circulaire. Le language $C' = \varphi(C)$ est un code circulaire sur l'alphabet B tel que les mots $\varphi(w_0)$ et $\varphi({}^\omega u_1 w_0)$ ont une infinité non dénombrable de C'-factorizations. •

Les tableaux de la proposition suivante résultent des définitions et des résultats démontrés dans cette section ou dans les précédentes.

La différence entre les mots infinis et les mots bi-infinis tient à l'absence d'une proposition analogue au théorème (1.1.2) dans le cas des mots bi-infinis. La différence entre les mots bi-infinis et les Z-mots réside dans le fait que:

- pour les mots non périodiques, cela revient au même d'exiger une seule C-factorisation ou d'exiger une seule C-décomposition,
- par contre, pour les mots périodiques, on exige beaucoup plus en demandant une seule C-décomposition plutôt qu'une seule C-factorisation; en fait on exige en plus la pureté du code.

Théorème 4.6.2 *Les tableaux suivants donnent, en fonction de la nature du code C, le nombre maximal de C-factorisations (ou de C-décompositions dans le cas des Z-mots) d'un mot de la famille considérée. Un astérisque $*$ indique que la propriété caractérise les codes particuliers considérés. Le symbole ∞ désigne le cardinal du continu.*

codes	mots infinis			
	u^ω $(u \in C^+)$	u^ω	uv^ω	quelconques
ω-code	1	1	1	1 $*$
faiblement préfixe	1	1	1 $*$	∞
π-code	1	1 $*$	fini	∞
code	1 $*$	fini $*$	fini $*$	∞

codes	mots bi-infinis		
	${}^\omega u^\omega$	${}^\omega u' v u^\omega$	quelconques
biω-code	1	1	1 $*$
fortement précirculaire	1	1 $*$	∞
précirculaire	1 $*$	fini	∞
code	fini $*$	fini $*$	∞

codes	Z-mots		
	périodiques	ultimement périodiques	quelconques
Z-code	1	1	1 *
circulaire	1 *	1 *	∞
code	fini *	fini *	∞

Un code circulaire est faiblement préfixe et fortement précirculaire (Mais la réciproque est fausse: le code préfixe $\{a, bab\}$ est un biω-code non pur). Un biω-code est un ω-code. Cela suggère l'idée du résultat suivant:

Proposition 4.6.3 *Tout code fortement précirculaire est faiblement préfixe.*

Preuve: Soit C un code fortement précirculaire. Si C n'a qu'un élément, C est un biω-code donc le résultat est évident. Sinon, soient w, wu, uv, vu des mots de C^*. Il s'agit de montrer que u appartient à C^*. Soient α et β des mots de C. Par hypothèse, le mot ultimement périodique ${}^\omega\beta wu(vu)^\omega = {}^\omega\ \beta w(uv)^\omega$ n'a qu'une seule C-factorisation. S'il n'est pas périodique, il n'a qu'une seule C-décomposition, donc u appartient à C^*. S'il est périodique, on prend α au lieu de β. Si les deux sont périodiques, les racines de uv, α et β sont conjuguées. Mais alors, α et β ont des puissances conjuguées et C ne peut être précirculaire. •

La réciproque est fausse: le code préfixe (donc faiblement préfixe) $\{ab, ba\}$ n'est pas fortement précirculaire: le mot ${}^\omega(ab)^\omega$ a deux factorisations.

Le résultat de la proposition précédente ne peut être amélioré: un code précirculaire peut ne pas être faiblement préfixe. Le code $\{a, bab, abab\}$ est précirculaire mais n'est pas un π-code.

Un code circulaire est faiblement préfixe et fortement précirculaire. Un code est circulaire si et seulement si il est précirculaire et pur (proposition 2.3.3). Il est naturel de regarder s'il existe d'autres relations entre les notions de code pur (resp: précirculaire) et de code faiblement préfixe ou fortement précirculaire. Il n'y en a pas. En effet, on peut considérer:
- le code bipréfixe et pur $\{ab, ba\}$ non précirculaire,
- le code biω-code bipréfixe $\{a, bab\}$ non pur,
- le code pur $\{a, ab, cab, bca\}$ non faiblement préfixe (car ce n'est pas un π-code: le mot $(abc)^\omega$ a deux factorisations), et non précirculaire,
- le code précirculaire $\{a, a^2ba, aba^2ba\}$ non faiblement préfixe (car ce n'est pas un π-code: le mot $(a^2b)^\omega$ a deux factorisations).

De plus, il existe des codes circulaires qui ne sont pas des ω-codes: le code $C = \{ab\} \cup \{ab^nab^{n+1} \mid n > 0\}$ en est un.

Les exemples donnés jusqu'ici et certaines propositions permettent d'énoncer le théorème suivant qui résume les inclusions entre les différentes classes de codes étudiées.

Théorème 4.6.4 *Dans le schéma suivant, les implications sont strictes et il n'y a pas d'autres implications que celles indiquées :*

Z-code	$\Rightarrow$	*code circulaire*	$\Rightarrow$	*code pur*
$\Downarrow$		$\Downarrow$		
biω-code	$\Rightarrow$	*code fortement précirculaire*	$\Rightarrow$	*code précirculaire*
$\Downarrow$		$\Downarrow$		
ω-code	$\Rightarrow$	*code faiblement préfixe*	$\Rightarrow$	*π-code*
$\Uparrow$				$\Uparrow$
code préfixe	$\Leftarrow$	*code bipréfixe*	$\Rightarrow$	*code suffixe*

Chapitre 5

Etude du cas rationnel

Dans cette partie, nous examinons comment se modifient les résultats obtenus quand les codes utilisés pour coder les mots infinis sont des codes rationnels. Ces modifications sont en relation avec le fait qu'un langage rationnel infinitaire est complètement spécifié par l'ensemble de ses mots ultimement périodiques [12].

Les résultats s'appuient sur le fait que les ensembles suivants :
- l'ensemble des mots infinis ayant plusieurs factorisations sur un langage rationnel donné,
- l'ensemble des mots bi-infinis dont les représentants ont plusieurs décompositions sur un langage rationnel donné,
sont rationnels.

A ce propos, il est d'ailleurs étonnant de constater que l'ensemble des mots bi-infinis ayant plusieurs factorisations sur un langage rationnel donné peut ne pas être rationnel.

La première modification apportée par la rationalité du code porte sur le nombre de factorisations d'un mot infini : sur un **code rationnel**, tout mot infini a un nombre fini de factorisations.

La seconde modification est relative aux classes de codes étudiées. Les intersections de certaines de ces classes aves la classe des codes rationnels sont confondues : la classe des ω-codes rationnels avec celle des codes faiblement préfixes rationnels, celle des biω-codes rationnels avec celle des codes fortement précirculaires rationnels, celle des Z-codes rationnels avec celle des codes circulaires rationnels. Cela donne donc, dans le cas rationnel, des caractérisations des ω-codes, biω-codes et Z-codes **en termes de mots finis.**

Ces modifications sont résumées par un tableau en section 3.

La sous-section suivante est consacrée aux résultats de **décidabilité.** Nous donnons ici, pour chaque classe de codes étudiée, une preuve que l'appartenance d'un langage rationnel donné à cette classe est décidable. Nous nous

sommes limités à des méthodes découlant directement des théorèmes démontrés, sans nous attarder à chercher d'autres algorithmes plus performants et sans étudier ou comparer leur complexité à celle d'algorithmes existant. En particulier, nous ne les comparons pas aux algorithmes inspirés de celui de Sardinas et Patterson décrits par Do Long Van pour décider qu'un langage rationnel est un ω-code) [26] ou pour décider qu'un langage rationnel est un Z-code [27].

Nous donnons finalement une dernière caractérisation des biω-codes, caractérisation qui est pour nous historique. Tout biω-code est un ω-code, un $\tilde{\omega}$-code et un code précirculaire ; nous pensions que, au moins dans le cas rationnel, ces trois conditions suffiraient pour avoir un biω-code. Il n'en est rien, il a fallu ajouter une quatrième condition pour obtenir une réciproque.

5.1 Codes rationnels et mots infinis

Dans le cas rationnel, il est intéressant d'étudier l'ensemble des mots ayant plusieurs C-factorisations :

Proposition 5.1.1 *: Soit C un langage rationnel inclus dans A^+. L'ensemble des mots finis qui ont plusieurs C-factorisations, l'ensemble des mots infinis qui ont plusieurs C-factorisations, l'ensemble des mots bi-infinis dont les représentants ont plusieurs C-décompositions sont rationnels.*

Preuve : Soit $\Omega_0 = (Q_0, q_0, q_F, R_0)$, un automate *ad hoc* pour C et $\Omega = (Q, q_0, q_0, R)$ l'automate dérivé de Ω_0. Etant donné des états q et q' de Q, notons $A_{q,q'}$ l'ensemble des mots lus sur Ω de q à q', et $B_{q,q'}$ l'ensemble $A_{q,q_0}.A_{q_0,q'}$.

Soit D l'ensemble des mots finis qui ont plusieurs C-factorisations. On sait que D est rationnel (substraction lemma) [29] [8].
On peut remarquer que l'on a :

$$D = \bigcup_{q \neq q_0} (C^+ \cap A_{q_0,q}).(A_{q,q_0} \cap C^+) \ .$$

Considérons les langages rationnels suivants : $X_1 = {}^{\omega}CDC^{\omega}$,

$$X_2 = \bigcup_{(q,q') \in Q^2 - (q_0,q_0)} {}^{\omega}(B_{q',q'} \cap C^+).(B_{q',q} \cap C^+).(B_{q,q} \cap C^+)^{\omega} \ ,$$

$$D' = DC^{\omega} \cup (\bigcup_{q \neq q_0} (C^+ \cap A_{q_0,q}).(B_{q,q} \cap C^+)^{\omega}) \ .$$

Clairement, tout représentant w d'un mot de X_1 a plusieurs C-décompositions. Tout représentant w d'un mot de X_2 a deux lectures réussies sur

Ω puisque $(q, q') \neq (q_0, q_0)$, donc w a au moins deux C-décompositions. De même, tout mot de D' a au moins deux C-factorisations.

Inversement, si $w \in A^Z$ a deux C-décompositions distinctes, w a deux lectures réussies sur Ω $(q_{j(i)}, w_i, q_{j(i+1)})_{i\in Z}$ et $(q_{k(i)}, w_i, q_{k(i+1)})_{i\in Z}$. Considérons la suite maximale $(i_p)_{p\in Z}$ d'indices i tels que $q_{j(i)} = q_0$. On regarde les états traversés aux instants i_p dans la seconde lecture. Puisque Q est fini, il existe q (resp. q') $\in Q$ tel que q (resp. q') soit traversé dans la seconde lecture, à une infinité d'instants i_p où $p > 0$(resp. < 0). La seconde lecture étant réussie, on peut ne retenir que les passages par q (resp q') entre lesquels existe un passage par q_0. Si $q \neq q_0$ ou $q' \neq q_0$, w est dans X_2. Si $q = q' = q_0$, puisque les C-décompositions sont distinctes, w est dans X_1.

Le langage $D'' = X_1 \cup X_2$ est donc l'ensemble des mots bi-infinis dont les représentants ont plusieurs C-décompositions.

On démontre de la même façon que D' est l'ensemble des mots infinis qui ont plusieurs C-factorisations. •

Autre preuve dans le cas des mots infinis [42]: Si C est rationnel, la semi-congruence définie par:

$$u \simeq v \Leftrightarrow u^{-1}C = v^{-1}C$$

est d'index fini. Si l'on note par $[u]$ la classe d'un mot u, l'ensemble des mots infinis qui ont plusieurs C-factorisations peut s'écrire:

$$\bigcup_{[u]\subset C} C^*.[u].(C^\omega \cap ([u]^{-1}C - \{\varepsilon\})C^\omega) \ .$$

Cet ensemble est rationnel. •

Proposition 5.1.2 *: Soit C un langage rationnel inclus dans A^+. L'ensemble Y_1 des mots bi-infinis ayant plusieurs C-factorisations et l'ensemble Y_2 des mots bi-infinis dont les représentants ont plusieurs C-décompositions et une seule C-factorisation peuvent ne pas être rationnels, même si C est un code fini. Les ensembles Y_1 et Y_2 sont rationnels en même temps, et ils le sont si et seulement si Y_2 est fini.*

Preuve: Y_1 et Y_2 sont disjoints, leur union D'' est l'ensemble des mot bi-infinis dont les représentants ont plusieurs C-décompositions et D'' est rationnel. D'autre part, Y_2 est constitué de mots périodiques. Y_1 et Y_2 sont donc rationnels en même temps, et le sont si et seulement si Y_2 est fini. Dans l'exemple suivant Y_2 est infini. •

Exemple: Considérons le code bipréfixe fini $C = \{ab, ba, aa\}$. Tout représentant d'un mot de la forme ${}^\omega(a^n b)^\omega$ où n est pair a deux C-décompositions mais une seule C-factorisation. L'ensemble Y_2 est infini.

Corollaire 5.1.3 *Soit C un code pur rationnel. L'ensemble Y_1 des mots bi-infinis ayant plusieurs C-factorisations est rationnel.*

Preuve: D'après la proposition 3.2.3, l'ensemble Y_2 des mots bi-infinis dont les représentants ont plusieurs C-décompositions et une seule C-factorisation est vide. •

Rappelons que la réciproque est fausse: Y_2 peut être vide sans que le langage C soit un code pur.

Si on s'intéresse au nombre de C-factorisations d'un mot donné w, on sait déjà que ce nombre est fini dans le cas où C est un code et w un mot rationnel (théorème 1.3.1). Mais, même si C est un code, cela peut être faux quand le mot n'est pas rationnel: on peut même avoir une infinité non dénombrable de C-factorisations (proposition 1.3.5). Dans le cas rationnel, cela change et on a le théorème suivant:

Théorème 5.1.4 *: Soit C un langage rationnel inclus dans A^+. Les conditions suivantes sont équivalentes:*
1) C est un code,
2) tout mot infini a un nombre fini de C-factorisations,
3) il existe p tel que tout mot infini ait au plus p C-factorisations,
4) tout Z-mot a un nombre fini de C-décompositions,
5) il existe q tel que tout Z-mot ait au plus q C-décompositions,
6) tout mot bi-infini a un nombre fini de C-factorisations,
7) il existe q tel que tout mot bi-infini ait au plus q C-factorisations.

Preuve: $3 \Rightarrow 2,5 \Rightarrow 4,5 \Rightarrow 7,7 \Rightarrow 6$: évident.
$2 \Rightarrow 1,4 \Rightarrow 1,6 \Rightarrow 1$: immédiat d'après le théorème 1.3.1.
$1 \Rightarrow 3$: Soit C un code rationnel, $\Omega_0 = (Q_0, q_0, q_F, R_0)$, un automate *ad hoc* pour C et $\Omega = (Q, q_0, q_0, R)$ l'automate dérivé de Ω_0. L'automate Ω est non ambigu pour C^*. Considérons $w \in C^\omega$ et $t \geq 1$. Nous appellerons découpage de (w,t) toute suite $(n_1, ..., n_{p-1})$ telle qu'il existe n_p vérifiant:

(1) $\quad n_0 = 0 < n_1 < ... < n_{p-1} \leq t < n_p$, $p \geq 2$ et $w[n_{i-1}, n_i[\in C$

pour $i = 1, ..., p$.

Nous montrons d'abord que, pour tout t, (w,t) a au plus $Card(Q)$ découpages. Considérons deux suites $(n_1, ..., n_p)$ et $(n'_1, ..., n'_k)$ vérifiant (1). Soit q (resp. q') l'état atteint après avoir lu $w[0,t[$ dans l'unique lecture réussie (sur Ω) de $w[0, n_p[$ (resp. $w[0, n'_k[$). Si $q = q'$, $w[0, n_p[$ a une autre lecture réussie: celle qui suit le chemin relatif à la lecture de $w[0, n'_k[$ jusque t, puis le chemin relatif

à $w[0, n_p[$. Alors $p = k$ et $(n_1, ..., n_{p-1}) = (n'_1, ..., n'_{k-1})$ puisque Ω est non ambigu. Il s'ensuit que (w, t) a au plus $Card(Q)$ découpages, quel que soit t.

On en déduit que w a au plus $Card(Q)$ C-factorisations.

$1 \Rightarrow 5$: On reprend la démonstration précédente avec un mot w de A^Z. Nous appelons maintenant découpage de (w, t) toute suite $(n_1, ..., n_{p-1})$ telle qu'il existe n_0, n_p vérifiant:

$$(2) \qquad n_0 < -t \leq n_1 < ... < n_{p-1} \leq t < n_p \ , \ p \geq 2 \text{ et } w[n_{i-1}, n_i[\in C$$

$$\text{pour } i = 1, ..., p \ .$$

Pour montrer que w a au plus $Card(Q)^2$ C-décompositions, nous montrons que, pour tout t, (w, t) a au plus $(Card(Q))^2$ découpages. Considérons deux suites $(n_0, ..., n_p)$ et $(n'_0, ..., n'_k)$ vérifiant (2). Soient q et q_1 (resp. q' et q'_1) les états atteints après avoir lu $w[n_0, -t[$ et $w[n_0, t[$ (resp. $w[n'_0, -t[$ et $w[n'_0, t[$) dans l'unique lecture réussie (sur Ω) de $w[n_0, n_p[$ (resp. $w[n'_0, n'_k[$). Si $q = q'$ et $q_1 = q'_1$, comme précédemment, $w[n_0, n_p[$ aurait deux lectures réussies. En fait, ces deux lectures sont identiques et $(n_1, ..., n_{p-1}) = (n'_1, ..., n'_{k-1})$. •

5.2 ω-codes, biω-codes et Z-codes dans le cas rationnel

De la proposition 5.1.1 on peut déduire que certaines classes de codes rationnels sont confondues. Cela donne, dans le cas rationnel, des définitions des ω-codes, biω-codes et Z-codes en termes de mots finis. Nous avons déjà remarqué que ces résultats sont faux dans le cas non rationnel (voir sections 3 et 4).

Dans le cas des mots infinis, ce résultat est un résultat de L. Staiger [61]. Dans celui des Z-mots, ce résultat est un résultat de M. P. Beal [2].

Corollaire 5.2.1 *Soit C un langage rationnel inclus dans A^+. On a:*

- *C est un ω-code si et seulement si C est un code faiblement préfixe,*
- *C est un biω-code si et seulement si C est un code fortement précirculaire,*
- *C est un Z-code si et seulement si C est un code circulaire.*

Preuve: L' ensemble des mots infinis ayant plusieurs C-factorisations étant rationnel, il peut être vide et alors C est un ω-code, ou bien il n'est pas vide et il contient des mots ultimement périodiques, et dans ce cas C n'est pas faiblement préfixe.

De même pour les C-décompositions des Z-mots: ou C est un Z-code, ou l'ensemble des mots bi-infinis dont les représentants ont plusieurs C-décompositions contient un mot rationnel et C n'est pas un code circulaire.

Dans le cas des mots bi-infinis, considérons l'ensemble D'' des mots bi-infinis dont les représentants ont plusieurs C-décompositions (défini dans la preuve de la proposition 5.1.1). Soit Y_1 l'ensemble des mots ayant au moins deux C-factorisations distinctes et Y_2 l'ensemble des mots périodiques ayant une seule C-factorisation et dont la C-factorisation est chevauchante. Clairement Y_1 et Y_2 sont disjoints et $D'' = Y_1 \cup Y_2$. Puisque D'' est rationnel, c'est une union finie d'ensembles ${}^{\omega}A_iB_iC_i^{\omega}$. Si ${}^{\omega}A_iB_iC_i^{\omega}$ n'est pas réduit à un unique mot périodique, il existe un mot rationnel non périodique dans D'' et donc un mot rationnel dans Y_1. Par conséquent, si C est un code fortement précirculaire, tout mot rationnel ayant au plus une C-factorisation, D'' est de la forme $\cup {}^{\omega}u_i^{\omega}$ (union finie), $D'' = Y_2$ et Y_1 est vide. Donc C est un biω-code. •

De cette démonstration on déduit le corollaire:

Corollaire 5.2.2 *Soit C un biω-code rationnel, l'ensemble des mots bi-infinis dont les représentants ont plusieurs C-décompositions est un ensemble fini de mots périodiques.*

Nous avons déjà vu que l'ensemble Y_2 pouvait être infini, même si le langage C est un code fini.

Le résultat du corollaire précédent est faux aussi pour un biω-code non rationnel. Soit $C = \{ab^nab^n \mid n > 0\}$; l'ensemble infini $\{{}^{\omega}(ab^n)^{\omega} \mid n > 0\}$ est l'ensemble des mots bi-infinis dont les représentants ont plusieurs C-décompositions.

Une autre conséquence de la proposition 5.1.1 est le corollaire:

Corollaire 5.2.3 *Un langage rationnel $C \subset A^+$ est un ω-code (resp. Z-code, resp. biω-code) si et seulement si tous ses sous-ensembles finis sont des codes faiblement préfixes (resp. codes circulaires, resp. codes fortement précirculaires).*

Preuve:
- Si C n'est pas un code, il existe un sous-ensemble fini de C qui n'est pas un code.
- Si C est un code rationnel mais pas un ω-code, il existe un mot ultimement périodique w qui a au moins deux C-factorisations. Ces factorisations sont ultimement périodiques (théorème 1.2.5). Ainsi $w = uv^{\omega} = u'v'^{\omega}$ où $u, v, u', v' \in C^+$. L'ensemble (fini) des mots de C intervenant dans les C-factorisations de u, v, u' et v' n'est pas un ω-code.
- Si C est un code rationnel mais pas un biω-code, le raisonnement est identique.
- Si C est un biω-code mais pas un Z-code, il existe un mot périodique ${}^{\omega}u^{\omega}$ (où $u \in C^+$) ayant une C-décomposition chevauchante. L'ensemble (fini) des mots de C intervenant dans la C-factorisation de u n'est pas circulaire. •

Ce résultat n'est pas vrai pour les codes non rationnels. Par exemple, tout sous-ensemble fini de $C = \{ab\} \cup \{ab^n ab^{n+1} \mid n > 0\}$ est un ω-code circulaire ; mais l'ensemble des mots infinis ayant plusieurs C-factorisations n'est pas rationnel, c'est $C^* w_0$ où $w_0 = abab^2ab^3ab^4....$ Le code C n'est pas un ω-code.

5.3 Tableaux récapitulatifs

Les tableaux de cette sous-section sont relatifs au cardinal de l'ensemble des C-factorisations d'un mot donné, quand le code C est rationnel. Ils résultent des définitions et des résultats démontrés dans ce qui précède .

Théorème 5.3.1 *Les tableaux suivants donnent, en fonction de la nature du code rationnel C, le nombre maximal de C-factorisations (ou de C-décompositions dans le cas des Z-mots) d'un mot de la famille considérée. Un astérisque $*$ indique que la propriété caractérise les codes particuliers considérés.*

codes **rationnels**	mots infinis			
	u^ω $(u \in C^+)$	u^ω	uv^ω	quelconques
ω-code	1	1	1 $*$	1 $*$
faiblement préfixe	1	1	1 $*$	1 $*$
π-code	1	1 $*$	fini	fini
code	1 $*$	fini $*$	fini $*$	fini $*$

codes **rationnels**	mots bi-infinis		
	${}^\omega u^\omega$	${}^\omega u' v u^\omega$	quelconques
biω-code	1	1 $*$	1 $*$
fortement précirculaire	1	1 $*$	1 $*$
précirculaire	1 $*$	fini	fini
code	fini $*$	fini $*$	fini $*$

codes **rationnels**	Z-mots		
	périodiques	ultimement périodiques	quelconques
Z-code	1 $*$	1 $*$	1 $*$
circulaire	1 $*$	1 $*$	1 $*$
code	fini $*$	fini $*$	fini $*$

Le théorème suivant résume les inclusions entre les différentes classes de codes rationnels étudiées.

Théorème 5.3.2 *Le schéma suivant concerne les* **codes rationnels**, *les implications y sont strictes et il n'y a pas d'autres implications que celles indiquées :*

Pour les codes rationnels :

Z-code	$\Leftrightarrow$	*code circulaire*	$\Rightarrow$	*code pur*
$\Downarrow$		$\Downarrow$		
biω-code	$\Leftrightarrow$	*code fortement précirculaire*	$\Rightarrow$	*code précirculaire*
$\Downarrow$		$\Downarrow$		
ω-code	$\Leftrightarrow$	*code faiblement préfixe*	$\Rightarrow$	*π-code*
$\Uparrow$				$\Uparrow$
code préfixe	$\Leftarrow$	*code bipréfixe*	$\Rightarrow$	*code suffixe*

5.4 Décidabilité

Dans cette section on montre que l'appartenance d'un langage donné aux classes de codes étudiées est décidable.

Lemme 5.4.1 *: On peut décider si un langage rationnel de A^ω contient un mot périodique.*

Preuve : Considérons un automate $\Omega = (Q, q_0, F, R)$ reconnaissant (au sens de Büchi) un langage rationnel infinitaire L. Notons $A_{q,q'}$ l'ensemble des mots qui peuvent être lus sur Ω de q à q', et par $D_{q,q'}$ l'ensemble des mots qui peuvent être lus sur Ω de q à q', en passant par un état final.
La congruence de Büchi sur A^* définie par :

$$u \simeq v \Leftrightarrow \forall q, q' \in Q \ (u \in A_{q,q'} \Leftrightarrow v \in A_{q,q'}) \text{ et } (u \in D_{q,q'} \Leftrightarrow v \in D_{q,q'})$$

est d'index fini d. Clairement on a :

$$\forall u, v \in A^* \quad u \simeq v \Rightarrow (u^\omega \in L \Leftrightarrow v^\omega \in L).$$

Il est donc suffisant d'étudier un mot par $\simeq$-classe, donc un nombre fini de mots. •

Corollaire 5.4.2 *: On peut décider si un langage rationnel est un ω-code (i.e. un code faiblement préfixe). On peut décider si un langage rationnel est un π-code.*

Preuve: Pour décider si C est un ω-code (resp. π-code), il suffit de tester si l'ensemble rationnel des mots infinis qui ont plusieurs C-factorisations est vide (resp. ne contient pas de mot périodique). •

Corollaire 5.4.3 *On peut décider si un langage rationnel est un Z-code (i.e. un code circulaire).*

Preuve: Il suffit de tester si l' ensemble rationnel des mots bi-infinis dont les représentants ont plusieurs C-décompositions est vide.
Autre preuve: Soit $C \subset A^+$ un langage rationnel. D'après la proposition 3.2.1, il suffit de tester si le langage: $(({}^\omega C)^{-1}({}^\omega C) \cap A^+).((C^\omega)(C^\omega)^{-1} \cap A^+) \cap C$ est vide ou non. Or ce langage est rationnel quand C est rationnel; en effet, d'une part, C^ω et ${}^\omega C$ sont des langages rationnels, d'autre part, si U et V sont des langages rationnels de A^ω (resp. ${}^\omega A$), UV^{-1} (resp. $U^{-1}V$) est un langage (finitaire) rationnel. •

Notation: Etant donné un automate $\Omega_0 = (Q_0, q_0, q_F, R_0)$, on note (dans le lemme suivant et ses applications) $\boldsymbol{A_q}$ l'ensemble des mots qui sont lus de q_0 à q et $\boldsymbol{B_q}$ l'ensemble des mots qui sont lus de q à q_F.

Lemme 5.4.4 *Soit C un code rationnel et $\Omega_0 = (Q_0, q_0, q_F, R_0)$ un automate ad hoc pour C. Le code C est presque circulaire (resp. circulaire) si et seulement si pour tout état q de $Q_0 - \{q_0, q_F\}$, $B_qC^*A_q \cap C^*$ est vide ou monogène (resp. vide).*

Preuve: Posons $Q' = Q_0 - \{q_0, q_F\}$.
Supposons que C soit un code presque circulaire. On a: $\forall q \in Q'$

$$\forall u, u' \in A_q \quad \forall v, v' \in B_q \quad (vC^*u \cap C^* \text{ et } v'C^*u' \cap C^* \neq \emptyset \Rightarrow vC^*u' \cap C^* \neq \emptyset)$$

puisqu'alors uv' est dans C. Comme $u'v \in C$ on a $v = v'$ (lemme 2.4.2) et donc $u = u'$. Alors $B_qC^*A_q \cap C^* = vC^*u \cap C^*$ est monogène.

Inversement, pour tous $u, v \neq \varepsilon$ tels que $uv \in C$ et $vC^*u \cap C^* \neq \emptyset$, il existe $q \in Q'$ tel que $(u, v) \in A_q \times B_q$ et donc $vC^*u \cap C^*$, inclus dans un semigroupe monogène, est monogène (lemme 2.4.3). •

Proposition 5.4.5 *On peut décider si un langage rationnel est un code presque circulaire (resp. circulaire).*

Preuve: Soit C un langage rationnel. On peut d'abord décider si C est un code.

Sur un automate $\Omega_0 = (Q_0, q_0, q_F, R_0)$ *ad hoc* pour C, on peut tester pour tout q de $Q_0 - \{q_0, q_F\}$ si le langage rationnel $B_qC^*A_q \cap C^*$ est vide ou non (les notations sont celles du lemme précédent). Si ce n'est pas le cas, on peut

rechercher un mot u_q appartenant à $B_qC^*A_q \cap C^*$ et de longueur minimale, et tester si $B_qC^*A_q \cap C^* = u_q^+$. •

Proposition 5.4.6 *On peut décider si un langage rationnel est précirculaire.*

Preuve: Soit C un langage rationnel et un automate $\Omega_0 = (Q_0, q_0, q_F, R_0)$ *ad hoc* pour C. À l'aide de Ω_0, on peut tester si C est presque circulaire. Si c'est le cas, d'après la caractérisation 2.4.5 des codes précirculaires, il suffit de rechercher, pour tout q de $Q_0 - \{q_0, q_F\}$, la C-factorisation de u_q et sa C-interprétation cyclique dans $B_qC^*A_q \cap C^*$ puis de tester la propriété nécessaire. •

De même que pour les codes presque circulaires, nous allons donner, en utilisant un automate, une caractérisation des biω-codes dans le cas rationnel. Cette caractérisation s'appuie sur le théorème 3.3.3.

Notation: Soit $\Omega_0 = (Q_0, q_0, q_F, R_0)$, un automate *ad hoc* pour C et $\Omega = (Q, q_0, q_0, R)$ l'automate dérivé de Ω_0. On note $\boldsymbol{C_{q,q'}}$ l'ensemble des mots pouvant être lus sur Ω_0 de q à q', $\boldsymbol{M_q}$ l'ensemble ${}^\omega C \cap {}^\omega C.C_{q_0,q}$ et $\boldsymbol{M'_q}$, l'ensemble $C^\omega \cap C_{q,q_0}.C^\omega$.

Théorème 5.4.7 *Soit C un code rationnel précirculaire inclus dans A^+ et $\Omega_0 = (Q_0, q_0, q_F, R_0)$ un automate ad hoc pour C. Le code précirculaire C est un biω-code si et seulement si pour tout $q \in Q_0 - \{q_0, q_F\}$ tel que $M_q.M'_q \neq \emptyset$, M_q et M'_q sont monogènes.*

Preuve: D'après la caractérisation 3.3.3 la condition est suffisante.

Inversement, considérons $s_0, s \in C_{q_0,q}$, $t \in C_{q,q_F}$ tels que M_{s_0}, M_s et $M'_t \neq \emptyset$. D'après la caractérisation 3.3.1, $M_{s_0} = {}^\omega u$, $M'_t = u^\omega$ et donc $M_s = {}^\omega u$. Alors $M_q = {}^\omega u$ et $M'_q = u^\omega$. •

Corollaire 5.4.8 *: On peut décider si un langage rationnel est un biω-code (i.e. un code fortement précirculaire).*

Preuve: On sait décider si un langage rationnel est un code précirculaire. Il suffit alors de décider si certains langages rationnels infinitaires (en nombre fini : les M_q et M'_q) sont monogènes. Pour cela on considère un mot ultimement périodique uv^ω d'un tel langage, on regarde si uv^ω est périodique et on teste si le langage est réduit à uv^ω. •

5.5 Une autre caractérisation des biω-codes rationnels

Le prochain lemme permet d'obtenir, dans le cas rationnel, une autre caractérisation des biω-codes.

Lemme 5.5.1 *Un biω-code $C \subset A^+$ vérifie la propriété :*
(4) : $\forall s,t,s',t',s'',t'' \in A^+$ tels que $st, s't', st''$ et $s''t' \in C$, $C^* \cap tC^*s$, $C^* \cap t''C^*s''$ et $C^* \cap t'C^*s' \neq \emptyset$, on a $t'' = t, s'' = s'$ et il existe $p > 0$, un mot primitif u, un conjugué v de u $(v = u''u', u = u'u'')$ tels que

$$C^* \cap tC^*s = (u^p)^+ \ , \ C^* \cap t''C^*s'' \subset u^*u' \text{ et } C^* \cap t'C^*s' = (v^p)^+ \ .$$

Preuve : Un biω-code C est presque circulaire et il existe des mots u et v primitifs et des entiers p et n tels que $C^* \cap tC^*s = (u^p)^+$ et $C^* \cap t'C^*s' = (v^n)^+$. Considérons $w \in C^* \cap t''C^*s''$. Les représentants de ${}^\omega uwv^\omega$ ont deux C-décompositions distinctes. C étant un biω-code, ${}^\omega uwv^\omega$ est périodique et a une C-factorisation chevauchante, donc $v = u''u'$, $u = u'u''$ et $w \in u^*u'$. On a aussi $n = p$ car C est un code précirculaire. La C-factorisation de ${}^\omega u^\omega$ est unique. Posons $t^{-1}ut = c_1 ... c_p$ où $c_p = st$ et c_i appartient à C pour tout i. La C-factorisation de ${}^\omega u^\omega$ est ${}^\omega(c_1, ..., c_p)^\omega$ et appartient à ${}^\omega(c_1, ..., c_p), c_1, ..., c_{p-1}, (st'')C^*(s''t')C^\omega$, donc $st'' = c_p = st$ et $t = t''$. De même $s'' = s'$. •

Un biω-code doit factoriser de façon unique les mots bi-infinis périodiques, les mots infinis à droite, les mots infinis à gauche, mais aussi les mots bi-infinis ultimement périodiques. Dans le cas rationnel, cette dernière condition suffit, tandis que la condition (4) ne suffit pas à elle seule. Dans le théorème suivant toutes ces conditions interviennent.

Théorème 5.5.2 *: Un langage rationnel est un biω-code si et seulement si il satisfait aux quatre conditions :*
- c'est un code précirculaire
- c'est un ω-code
- c'est un $\tilde{\omega}$-code
- il vérifie la propriété :
(4') : $\forall s,t,s',t',s'',t'' \in A^+$ tels que $st, s't', st''$ et $s''t' \in C$, $C^* \cap tC^*s$, $C^* \cap t''C^*s''$ et $C^* \cap t'C^*s' \neq \emptyset$, il existe $p,n > 0$, un mot primitif u, un conjugué v de u $(v = u''u', u = u'u'')$ tels que

$$C^* \cap tC^*s = (u^p)^+ \ , \ C^* \cap t''C^*s'' \subset u^*u' \text{ et } C^* \cap t'C^*s' = (v^n)^+ \ .$$

Preuve : Nous avons déjà vu que les conditions sont nécessaires. Pour prouver la suffisance, nous montrons que tout mot rationnel a au plus une C-factorisation. Supposons qu'un représentant r du mot ${}^{\omega}uwv^{\omega}$ (u et v primitifs) ait deux C-décompositions d_1 et d_2 et considérons $\varphi : X^* \to C^*$ un morphisme de codage pour C. L'application τ_r associe aux deux décompositions deux C-factorisations f_1 et f_2. Puisque C est un code, il existe $x \in X^*$ et des mots primitifs $y, z \in X^+$ tels que $f_1 = {}^{\omega}yxz^{\omega}$. Puisque C est un code précirculaire, il existe $x' \in X^*$ tel que $f_2 = {}^{\omega}yx'z^{\omega}$ (lemme 2.2.1).

Observons les décompositions d_1 et d_2. Deux cas se présentent :
- les décompositions d_1 et d_2 coupent le mot r en (au moins) un même point ($d_1(Z) \cap d_2(Z) \neq \emptyset$), alors $d_1 = d_2$ puisque C est un $\tilde{\omega}$-code et un ω-code.
- les décompositions d_1 et d_2 ne coïncident nulle part. Puisque $f_1 = {}^{\omega}yxz^{\omega}$ et $f_2 = {}^{\omega}yx'z^{\omega}$, on peut trouver $s, t, s', t' \in A^+$ tels que $st \in C$ et $\varphi(y) \in C^* \cap tC^*s$, ($s, t \neq \varepsilon$ puisque d_1 et d_2 ne coïncident nulle part), $s't' \in C$ et $\varphi(z) \in C^* \cap t'C^*s'$. Or d_1 et d_2 décomposent le même mot, on peut donc trouver $s'', t'' \in A^+$ tels que $st'' \in C$, $t's' \in C$, $C^* \cap t''C^*s'' \neq \emptyset$ et le mot ${}^{\omega}uwv^{\omega}$ appartient à ${}^{\omega}(C^* \cap tC^*s)(C^* \cap t''C^*s'')(C^* \cap t'C^*s')^{\omega}$. D'après la condition (4) le mot ${}^{\omega}uwv^{\omega}$ est périodique, et, puisque C est précirculaire, ${}^{\omega}uwv^{\omega} (= {}^{\omega}u^{\omega})$ a une seule C-factorisation. •

Dans le théorème précédent, aucune condition ne peut être supprimée, même dans le cas fini. De plus, ce théorème est faux pour un langage non rationnel. En effet :
- $C = \{abab\} \cup \{ab^nab^{n+1} \mid n > 0\}$ est un ω-code suffixe et précirculaire, mais non rationnel, qui satisfait à (4). Pour tous $s, t, s', t', s'', t'' \in A^+$ tels que $st, s't', st'', s''t' \in C$, $C^* \cap tC^*s, C^* \cap t''C^*s''$ et $C^* \cap t'C^*s' \neq \emptyset$, on a : $s = s' = s'' = t = t' = t'' = ab$ et $C^* \cap tC^*s = (abab)^+$. Mais le mot ${}^{\omega}(ab)ab^2ab^3ab^4...$ a deux C-factorisations.
- $C = \{a^2, ab, b^2\}$ est un code fini précirculaire et bipréfixe, mais ${}^{\omega}ab^{\omega}$ a deux C-factorisations.
- $C = \{a, ab, b^2\}$ est un $\tilde{\omega}$-code fini et précirculaire qui satisfait à (4), mais ${}^{\omega}ab^{\omega}$ a deux C-factorisations.
- $C = \{ab, ba\}$ est un code fini bipréfixe qui satisfait à (4), mais ${}^{\omega}(ab)^{\omega}$ a deux C-factorisations.

D'autre part, dans le théorème précédent, la propriété (4') (ou la propriété (4)) ne peut pas être remplacée par la propriété (5) :
(5) : $\forall s, t, s', t', s'', t'' \in A^+$ tels que $st, s't', st''$ et $s''t' \in C$

$$C^* \cap tC^*s, C^* \cap t''C^*s'', C^* \cap t'C^*s' \neq \emptyset \Rightarrow t'' = t \text{ et } s'' = s'.$$

En effet, le langage $C = \{a, aba^2b, a^2bc, c^2\}$ est un ω-code précirculaire et suffixe qui ne vérifie pas la propriété (4') : ${}^{\omega}(a^2b)c^{\omega}$ a deux C-factorisations mais C

vérifie la propriété (5) puisque:
- $C^* \cap tC^*s \neq \emptyset$ et $st \in C \Rightarrow (s,t) \in \{(c,c),(ab,a^2b),(aba,ab)\}$,
- $st \in C$ et $s \in \{c,ab,aba\} \Rightarrow (s,t) \in \{(c,c),(ab,a^2b),(aba,ab)\}$.

Par contre, on peut modifier la condition (4) en considérant un automate $\Omega_0 = (Q_0, q_0, q_F, R_0)$ *ad hoc* pour C et $\Omega = (Q, q_0, q_0, R)$ l'automate dérivé de Ω_0. Notons alors $C_{q,q'}$ l'ensemble des mots pouvant être lus sur Ω_0 de q à q' et $T_{q,q'}$, l'ensemble $C^+ \cap C_{q,q_0}.C^*.C_{q_o,q'}$. Dans le théorème précédent, on peut remplacer la condition (4) par la suivante:
$(4'')$: $\forall q, q' \in Q_0 - \{q_0, q_F\}$ tels que $q \neq q'$, si $T_{q,q}, T_{q,q'}$ et $T_{q',q'}$ sont non vides, il existe $n,p > 0$, un mot primitif u, un conjugué v de u $(v = u''u', u = u'u'')$ tels que $T_{q,q} = (u^p)^+$, $T_{q,q'} \subset u^*u'$ et $T_{q',q'} = (v^n)^+$.

Chapitre 6

Codes à délai de déchiffrage fini

Le décodage des mots infinis, unique si le code choisi est un ω-code, est facilité si le code est à délai de déchiffrage fini.

Dans cette partie nous étudions les **codes à délai de déchiffrage fini**, et plus particulièrement ceux qui sont rationnels.

On peut définir diverses notions de langage à délai de déchiffrage fini. Nous ne retiendrons ici que celles où l'on compte le nombre de mots. Ces notions sont moins restrictives que celles où l'on compte le nombre de lettres, et cependant elles sont encore suffisamment intéressantes pour faciliter le décodage [13].

Dans un premier temps, nous énonçons quatre propriétés, notées P_ω, P_*, Q_ω, Q_* en relation avec une possibilité de décodage avec délai fini. Pour un code, il s'avère que ces propriétés sont équivalentes. Un code vérifiant ces propriétés est par définition un code à délai (de déchiffrage) fini. Un code de délai m est un langage C tel que, dès que l'on a reconnu, au début d'un mot fini ou infini à droite, $(m+1)$ mots de C consécutifs, on est sûr du premier de ces mots.

Nous démontrons que, pour un code C à délai fini, le langage C^ω est un langage limite qui n'est généralement pas une adhérence. Quand l'alphabet est fini, c'est une adhérence si et seulement si C est fini.

Les codes à délai borné sont a priori plus intéressants que les codes à délai fini. En fait ces notions coïncident pour les langages rationnels [61]. Nous présentons et comparons deux conditions nécessaires et suffisantes pour qu'un ω-code rationnel C soit à délai borné. L'une d'elles est de Capocelli [17] ; l'autre est nouvelle et requiert que $C^\omega \cap C^* Adh(C)$ soit vide.

En conséquence de ces résultats, on peut décider si un code rationnel C est à délai de déchiffrage borné et quand c'est le cas, C^ω est un langage rationnel déterministe.

6.1 Codes à délai fini

Définitions 6.1.1 *On définit de la façon suivante les propriétés P_ω, P_*, Q_ω, Q_* d'un langage G inclus dans A^+ :*
$P_\omega : \forall v \in G \ \exists m(v) \geq 0 \ \forall v' \in G \ (\ vG^{m(v)}A^\omega \cap v'G^\omega \neq \emptyset \ \Rightarrow \ v = v' \ )$,
$P_* : \forall v \in G \ \exists m(v) \geq 0 \ \forall v' \in G \ (\ vG^{m(v)}A^* \cap v'G^* \neq \emptyset \ \Rightarrow \ v = v' \ )$ *[17] [8],*
$Q_* : \forall v \in G \ \exists m(v) \geq 0 \ vG^{m(v)}A^* \cap G^* \subset vG^*$,
$Q_\omega : \forall v \in G \ \exists m(v) \geq 0 \ vG^{m(v)}A^\omega \cap G^\omega \subset vG^\omega$ *[61].*

Les nombres $m(v)$ qui interviennent dans les définitions précédentes seront appelés **délais pour** v (pour la propriété P_ω, P_*, Q_ω ou Q_* considérée). Pour chaque propriété, le plus petit $m(v)$ possible sera appelé **le délai de** v. Si un mot v n'admet pas de délai pour une certaine propriété (par exemple pour Q_* : si $\not\exists m \geq 0 \ vG^mA^* \cap G^* \subset vG^*$), on dira que son délai (ici pour Q_*) est **infini.**

On voit facilement que :
- P_ω et P_* sont équivalentes,
- P_* implique Q_*,
- Q_* implique Q_ω.

Cependant Q_ω et Q_* ne sont pas équivalentes. Par exemple, le langage $G = \{a, bb, ba, ab\}$ vérifie Q_ω : $m(v) = 0$ pour tout v, car $G^\omega = A^\omega$. Mais G ne vérifie pas Q_* ; en effet, pour savoir s'il existe une factorisation de ab^n qui commence par a, il faut connaître la parité de n.

Un langage C vérifiant P_* est intéressant dans la mesure où la factorisation du début d'un mot infini v en un nombre suffisant de mots de C nous assure que le premier mot de cette factorisation est le bon. Les propriétés Q_ω et Q_* n'en sont pas négligeables pour autant. En effet, même si le langage G n'est pas un ω-code, on est quand même assurés d'obtenir de la même façon une factorisation du mot considéré.

Rappelons qu'un langage qui vérifie P_* est un code [8]. Ce n'est pas toujours le cas pour un langage qui vérifie Q_ω ou Q_*, et donc les propriétés Q_* et P_* ne sont pas équivalentes. Par exemple le langage $G = \{a^2, a^3\}$ vérifie Q_* (avec $m(v) = 1$ pour tout v) et ce n'est pas un code (de plus, ici, G^* n'est pas libre).

On voit aussi clairement qu'un code qui vérifie Q_* vérifie P_ω. En fait, on a :

Proposition 6.1.1 *Pour un code, les propriétés P_ω, P_*, Q_ω et Q_* sont équivalentes.*

Preuve : Il s'agit de montrer qu'un code qui vérifie Q_ω vérifie aussi P_ω.

Soit C un code vérifiant Q_ω. Montrons que c'est un ω-code. Soient $v, v' \in C$ tels que $vC^\omega \cap v'C^\omega \neq \emptyset$. Pour $n \geq max(m(v), m(v'))$ et $w \in vC^\omega \cap v'C^\omega$, il

existe $u, u' \in C^n$ tels que vu et $v'u'$ soient facteurs gauches de w. Si vu est un facteur gauche de $v'u'$, $(v'u')^\omega \in vC^{m(v)}A^\omega \cap C^\omega$, donc $(v'u')^\omega \in vC^\omega$. Puisque $v'u'$ appartient au monoïde libre C^*, d'après le théorème 1.1.2, $v = v'$. Donc C est un ω-code.

Considérons maintenant v et $v' \in C$ tels que $vC^{m(v)}A^* \cap v'C^* \neq \emptyset$ et $w \in vC^{m(v)}A^* \cap v'C^*$. Le mot wv^ω appartient à $vC^{m(v)}A^\omega \cap C^\omega$ et donc appartient à vC^ω et à $v'C^\omega$, et donc $v = v'$ puisque C est un ω-code. Le code C vérifie donc la propriété P_ω. •

Définition 6.1.2 *Un langage C inclus dans A^+ est dit* **code à délai de déchiffrage fini** *ou plus simplement* **code à délai fini** *s'il vérifie P_ω ou P_* ou encore si c'est un code qui a les propriétés Q_ω ou Q_*.*

La proposition suivante utilise la notion d'adhérence. Rappelons qu'un mot infini w appartient à l'adhérence $\boldsymbol{Adh(C)}$ d'un langage C ($\subset A^+$), si et seulement si tout facteur gauche de w est facteur gauche d'un mot de C.

Proposition 6.1.2 *Tout code à délai fini C inclus dans A^+ est un ω-code et vérifie : $C^\omega \cap C^*Adh(C) = \emptyset$.*

Preuve : D'après la propriété P_ω, il est clair qu'un code à délai fini est un ω-code.

Soit C un langage et w appartenant à $C^\omega \cap C^*Adh(C)$ (supposé non vide). Sans perte de généralité, on peut supposer que $w = u_1u_2u_3.... = u'_1u'_2...u'_pw'$ où $u_i, u'_i \in C$ pour tout i , $w' \in Adh(C)$, et $u'_1 \neq u_1$ ou $p = 0$. Puisque $w' \in Adh(C)$, pour tout $d \geq 0$ il existe $v \in C$ tel que $u_1...u_{d+1} \leq u'_1u'_2...u'_pv$. Comme $u'_1 \neq u_1$ ou $p = 0$, le mot u_1 n'est pas de délai d, ceci pour tout d. Donc C ne vérifie pas la propriété P_*. Ce n'est pas un code à délai fini. •

La réciproque de la proposition 6.1.2 est fausse :

Exemple : Le langage $C = \{a\} \cup b^*c \cup \{ab^nc^nd \mid n > 0\}$ est un ω-code (non rationnel). Aucune puissance de b n'est dans C et $Adh(C) = b^\omega \cup ab^\omega$ donc $C^\omega \cap C^*Adh(C) = \emptyset$. Cependant, le délai de a est infini.

De la proposition 6.1.2 on peut déduire une nouvelle preuve d'un résultat de Staiger [61] :

Corollaire 6.1.3 *Si C est un code à délai fini, le langage C^ω est un G_δ-langage (et donc la limite d'un langage finitaire).*

Preuve : Puisque $Adh(C^*) = C^\omega \cup C^*Adh(C)$ [40], si $C^\omega \cap C^*Adh(C) = \emptyset$, le langage C^ω est la différence entre le fermé $Adh(C^*)$ et le F_σ-langage $C^*Adh(C)$, donc C^ω est un G_δ-langage. •

Cette propriété n'est pas vérifiée par les ω-codes. Staiger en donne l'exemple suivant : $C = a + ba + (ab + ac)^*aca$ [61].

Staiger a remarqué aussi que cette propriété ne signifie pas que $C^\omega = \cap_{i\geq 1} C^i A^\omega$. L'exemple qu'il donne est le suivant.

Exemple : Posons $w_1 = b$, $w_{i+1} = (w_i)^i a^i b$ pour tout $i \geq 1$ et $C = \{w_i \mid i \geq 1\}$. Le langage C est un code à délai fini (mais non borné) : pour tout i, le délai de w_i est $\frac{i(i-1)}{2} + 1$. L'unique mot β appartenant à $Lim(C)$ est dans $\cap_{i\geq 1} C^i A^\omega$ mais n'est pas dans C^ω.

Corollaire 6.1.4 *Si C est un code à délai fini, C^ω est une adhérence si et seulement si $Adh(C) = \emptyset$. Si de plus A est fini, C^ω est une adhérence si et seulement C est fini.*

Preuve : D'après [11], C^ω est une adhérence si et seulement si C^ω est l'adhérence de C^*. Or $Adh(C^*) = C^\omega \cup C^* Adh(C)$, et dans le cas présent $C^\omega \cap C^* Adh(C) = \emptyset$, donc C^ω est une adhérence si et seulement si $C^* Adh(C) = \emptyset$ ou encore $Adh(C) = \emptyset$. Si l'alphabet est fini, l'adhérence de C est vide si et seulement si C est fini. •

6.2 Codes à délai borné

Le dernier exemple cité montre qu'un code à délai fini n'est pas forcément à délai borné. Les codes à délai borné sont plus souvent utilisés que les codes à délai fini. Pour définir des notions de langage à délai borné il suffit simplement de borner la fonction délai, c'est-à-dire de remplacer dans les définitions « $\forall v \in G\ \exists m(v) \geq 0$ » par « $\exists m \geq 0\ \forall v \in G$ ».

Définitions 6.2.1 *On définit de la façon suivante les propriétés P'_ω, P'_*, Q'_ω, Q'_* d'un langage G inclus dans A^+ :*
P'_ω : $\exists m \geq 0\ \ \forall v \in G\ \ \forall v' \in G\ (\ vG^m A^\omega \cap v'G^\omega \neq \emptyset \ \Rightarrow\ v = v'\ )$,
P'_* : $\exists m \geq 0\ \ \forall v \in G\ \ \forall v' \in G\ (\ vG^m A^* \cap v'G^* \neq \emptyset \ \Rightarrow\ v = v'\ )$ *[17] [8]*,
Q'_* : $\exists m \geq 0\ \ \forall v \in G\ \ vG^m A^* \cap G^* \subset vG^*$,
Q'_ω : $\exists m \geq 0\ \ \forall v \in G\ \ vG^m A^\omega \cap G^\omega \subset vG^\omega$ *[61]*.

Il est clair qu'un langage à délai borné pour l'une des propriétés considérées est à délai fini : P'_* implique P_*, P'_ω implique P_ω, Q'_* implique Q_* et Q'_ω implique Q_ω.

Les réciproques sont clairement vraies dans le cas d'un langage fini, mais fausses en général. Un contre-exemple sera donné par la suite.

Les relations existant entre les propriétés P'_ω, P'_*, Q'_ω et Q'_* sont les mêmes que pour les propriétés P_ω, P_*, Q_ω et Q_* :
- P'_ω et P'_* sont équivalentes (avec le même m),
- P'_* implique Q'_*,
- Q'_* implique Q'_ω,
- Q'_ω et Q'_* ne sont pas équivalentes (on peut prendre l'exemple donné pour Q_ω et Q_*),
- un langage qui vérifie P'_* est un code,
- un langage qui vérifie Q'_ω ou Q'_* n'est pas toujours un code, et donc les propriétés Q'_* et P'_* ne sont pas équivalentes (on peut prendre l'exemple donné pour Q_*),
- un code qui vérifie Q'_ω vérifie P'_ω,
- pour un code, les propriétés P'_ω, P'_*, Q'_ω et Q'_* sont équivalentes.

On peut donc donner la définition suivante :

Définition 6.2.2 *Un langage C inclus dans A^+ est dit* **code à délai borné** *s'il vérifie P'_ω ou P'_* ou encore si c'est un code qui a les propriétés Q'_ω ou Q'_*.*

L'exemple de Staiger d'un code à délai fini et non borné (fin de sous-section 6.1.1) montre aussi qu'il n'y a pas équivalence entre les propriétés Q_* et Q'_* ni entre les propriétés Q_ω et Q'_ω, puisque, pour un code , les propriétés P'_* et Q'_ω sont équivalentes.

Nous avons vu que C^ω est un G_δ-langage quand C est un code à délai fini. Staiger a démontré que, pour un code C à délai borné, ce G_δ-langage C^ω vérifie de plus $C^\omega = \cap_{i \geq 1} C^i A^\omega$ [61].

6.3 Etude du cas rationnel

Une propriété très intéressante des codes rationnels à délai de déchiffrage borné est obtenue comme corollaire de la proposition 6.1.3 :

Théorème 6.3.1 *Si C est un code rationnel à délai de déchiffrage borné, C^ω est un langage rationnel déterministe.*

Il est intéressant d'avoir des conditions nécessaires et suffisantes pour qu'un langage soit un code à délai borné. C'est dans le cas rationnel qu'il y a des résultats.

Nous avons vu que la réciproque de la proposition 6.1.2 est fausse en général : un ω-code C vérifiant $C^\omega \cap C^* Adh(C) = \emptyset$ peut ne pas être à délai fini. Dans le cas rationnel, la condition $C^\omega \cap C^* Adh(C) = \emptyset$ ne suffit pas pour qu'un code C soit à délai borné : le code fini $\{a, ab, bb\}$ n'est pas un ω-code.

De même, un ω-code, même rationnel, peut ne pas être à délai fini: si $C = ab^*d + a + b$, le délai de a est infini.

On a cependant la caractérisation:

Proposition 6.3.2 *Un langage rationnel C inclus dans A^+ est un code à délai borné si et seulement si c'est un ω-code qui vérifie $C^\omega \cap C^*Adh(C) = \emptyset$.*

Preuve: Nous avons déjà montré qu'un code à délai borné C est un ω-code qui vérifie: $C^\omega \cap C^*Adh(C) = \emptyset$.

Etudions la réciproque. Soit C un langage rationnel et $\Omega = (Q, q_0, q_0, R)$ l'automate dérivé d'un automate *ad hoc* pour C. Soit d le nombre de ses états. Supposons que C ne soit pas un code de délai d. Il existe $n \geq 0$, $u_0, ..., u_d, u'_0, ..., u'_n \in C$, $z \in A^*$ tels que $u_0...u_dz = u'_0...u'_n$ et $u'_0 \neq u_0$.

On peut trouver un chemin de label $u'_0...u'_n$ de q_0 à q_0. Dans ce chemin, notons q_{i_j} l'état atteint après avoir lu $u_0u_1...u_j$. Il existe j et $j' > j$ tels que $q_{i_j} = q_{i_{j'}}$. Notons alors: $y = u_0...u_j = u'_0...u'_{m-1}x'$ où $x' < u'_m$, $x = u_{j+1}...u_{j'}$, $x'xx'' = u'_m...u'_{m+h}$, où x'' est un facteur droit de u'_{m+h}, $u_{j'+1}...u_dz = x''u'_{m+h+1}...u'_n$.

Si $h = 0$, $x'x^nx''$ appartient à C pour tout n et par conséquent yx^ω appartient à $C^*Adh(C) \cap C^\omega$. Cet ensemble n'est donc pas vide.

Si $h \geq 1$, yx^ω a deux C-factorisations distinctes: $u_0, ..., u_j, (u_{j+1}, ..., u_{j'})^\omega$ et $u'_0, ..., u'_{m-1}, (u'_m, ..., u'_{m+h-1}, v)^\omega$ où $v = (u'_{m+h}x''^{-1}).(x'^{-1}u'_m)$, C n'est donc pas un ω-code. •

Remarquons que l'on vient de démontrer que le délai du code est au plus le nombre d'états de l'automate Ω considéré.

De cette proposition on déduit le résultat suivant, résultat déjà démontré par Levenshtejn [39], Capocelli [17] et Blanchard [9].

Corollaire 6.3.3 *Tout ω-code fini est à délai borné.*

Preuve: $Adh(C)$ est vide quand C est fini. •

De la proposition 6.3.2, on déduit facilement le résultat de Cori:

Corollaire 6.3.4 *[18] On peut décider si un langage rationnel est un code à délai borné, et donner un majorant de ce délai.*

Preuve: On sait décider si un langage rationnel C est un ω-code et si le langage rationnel infinitaire $C^\omega \cap C^*Adh(C)$ est vide ou non. Si C est un code à délai borné, le nombre d'états de l'automate dérivé d'un automate pour C majore le délai de C. •

Le théorème suivant énonce des caractérisations des codes à délai borné dans le cas rationnel. L'équivalence (0) $\Leftrightarrow$ (1) est de Staiger, l'équivalence (0) $\Leftrightarrow$ (2) de Capocelli, et nous venons de démontrer l'équivalence: (0) $\Leftrightarrow$ (3).

Théorème 6.3.5 *Soit C un langage rationnel inclus dans A^+. Les conditions suivantes sont équivalentes:*
(0) - C est un code à délai borné,
(1) - C est un code à délai fini,
(2) - C est un ω-code vérifiant: $\exists p \geq 0 \ \forall u \in A^*$

$$(\ uC^pA^* \cap C \neq \emptyset \ \Rightarrow \ C^+u \cap C^+ = \emptyset \ ) \ ,$$

(3) - C est un ω-code vérifiant: $C^\omega \cap C^*Adh(C) = \emptyset$.

Dans ce théorème, puisqu'il s'agit de langages rationnels, on peut remplacer le terme « ω-code » par le terme « faiblement préfixe ».

Dans le cas d'un ω-code fini, toutes les conditions sont trivialement vérifiées.

Comparons les caractérisations (2) et (3) du théorème 6.3.5.
- l'expression de la caractérisation (3) est plus simple et sans quantificateur,
- l'implication: (2) $\Rightarrow$ (3) se démontre aisément par contraposition,
- l'implication: (3) $\Rightarrow$ (2) ne semble pas simple à démontrer. Le moyen le plus économique semble être: (3) $\Rightarrow$ (0) et (0) $\Rightarrow$ (2) ,
- l'implication: (3) $\Rightarrow$ (2) nécessite l'hypothèse de rationnalité (voir le premier exemple présenté ci-après),
- la condition (2) est vérifiée par un code à délai borné (il suffit de prendre $p = m$), mais elle peut ne pas l'être par un code à délai fini (voir le second exemple présenté ci-après),
- la condition (2) est donc plus astreignante que la condition (3).

Comme condition suffisante, la propriété (3) est donc plus intéressante que la propriété (2).

Exemple: Le langage $C = \{a\} \cup b^*c \cup \{ab^nc^nd \mid n > 0\}$ est un ω-code non rationnel, ne vérifiant pas (2) (on peut prendre $u = a$ pour tout p). Cependant, il vérifie (3) puisque $Adh(C) = b^\omega \cup ab^\omega$ et aucune puissance de b n'est dans C. Donc l'implication: (3) $\Rightarrow$ (2) nécessite l'hypothèse de rationnalité.

Exemple: Reprenons l'exemple de la sous-section 6.1 : $w_1 = b$, $w_{i+1} = (w_i)^i a^i b$ pour tout $i \geq 1$ et $C = \{w_i \mid i \geq 1\}$. Le langage C est un code à délai fini (non borné) qui ne vérifie pas la propriété (2). En effet, pour tout p, posons $u = w_{p+1}$, on a: $w_{p+2} \in uC^pA^* \cap C$ et $u \in C$.

Ces exemples montrent aussi que le théorème précédent n'est plus vrai quand on supprime l'hypothèse de rationnalité.

Cependant, on a le lemme suivant dont le résultat aurait pu servir à démontrer le théorème 6.3.5.

Lemme 6.3.6 *Soit G un langage rationnel inclus dans A^+. Les conditions suivantes sont équivalentes:*
(2') - G vérifie: $\exists p \geq 0 \ \forall u \in A^*$

$$(\ uG^pA^* \cap G \neq \emptyset \ \Rightarrow \ G^+u \cap G^+ = \emptyset \ ) \ ,$$

(3') - G vérifie: $G^\omega \cap G^*Adh(G) = \emptyset$.

Preuve : (2') $\Rightarrow$ (3') se démontre aisément par contraposition, même si G n'est pas rationnel.
(3') $\Rightarrow$ (2') se démontre aussi par contraposition:

Soit G un langage rationnel et $\Omega = (Q, q_0, q_0, R)$ l'automate dérivé d'un automate *ad hoc* pour G. Soit d le nombre de ses états. Si (2') n'est pas vérifiée, pour tout $p > 0$, il existe $r_p \in A^*$, $v_1, ..., v_p, u_p \in G$, $w_p \in G^+$ tels que $w_p r_p \in G^+$, $r_p v_1 ... v_p \in Fg(u_p)$; notons: $u_p = r_p v_1 ... v_p s_p$. Considérons une lecture réussie des mots $w_p, u_p, w_p r_p, v_1, ..., v_p$ sur Ω.

On peut toujours choisir les mots v_j de longueur inférieure ou égale à d^2. En effet, si ce n'était pas le cas, le mot v_j pourrait s'écrire $t_1 t_2 t_3$ où les états atteints après avoir lu $r_p v_1 ... v_{j-1} t_1$ et $r_p v_1 ... v_{j-1} t_1 t_2$ dans la lecture réussie du mot u_p sont égaux, et où les états atteints après avoir lu t_1 et $t_1 t_2$ dans la lecture réussie du mot v_j sont égaux; on pourrait alors remplacer v_j par $t_1 t_3$. De la même façon, on peut choisir les mots w_p et r_p de longueur inférieure ou égale à d^2.

Le nombre de r_p, $v_1, ..., v_p$, u_p, w_p possibles est donc fini. Il existe une infinité de p ayant le même couple (w_p, r_p). On choisit un tel couple (w_p, r_p) (que l'on note (w, r) dans la suite) et on désigne par I_0 l'ensemble des p correspondants. Il existe une infinité de $p \in I_0$ ayant le même v_1. On choisit un tel v_1 et on désigne par I_1 l'ensemble des p correspondants. Par récurrence, on obtient une suite de mots $v_j \in G$ et une suite d'ensembles infinis I_j. Clairement, le mot $wrv_1 v_2 ... v_j ...$ appartient à G^ω. D'autre part, pour tout j, pour tout p de I_j, on a $rv_1 v_2 ... v_j \in Fg(u_p)$ et $u_p \in G$, donc le mot $wrv_1 v_2 ... v_j ...$ appartient à $G^*Adh(G)$. Donc $G^\omega \cap G^*Adh(G) \neq \emptyset$. •

Chapitre 7

Codes à délai de synchronisation borné

Le décodage des mots bi-infinis, unique si le code choisi est un biω-code, est facilité si le code est à délai de synchronisation borné.

Considérons un mot bi-infini codé sur un langage G, ou un mot quelconque, codé, dont on ne possède qu'un facteur. Si l'on veut pouvoir décoder ce mot ou ce facteur à la lecture (à partir d'un certain point, vers la droite ou (et) vers la gauche), il faut que le langage G préserve une possibilité de synchronisation. On ne peut plus ici considérer de propriétés analogues à P_*, P_ω, Q_* ou Q_ω intéressantes en elles-mêmes : il faudra imposer que G soit un code.

Comme pour le délai de déchiffrage, on peut, pour la synchronisation, définir un délai fini. La propriété analogue à R_* pourrait être par exemple :

R'_* : $\forall u \in G\ \exists m(u) \geq 0\ \exists m'(u) \geq 0\ \forall f, f' \in A^*\ \forall v \in G^{m(u)}\ \forall v' \in G^{m'(u)}$

$$fvuv'f' \in G^* \Rightarrow fv, v'f' \in G^* \quad .$$

Nous ne présentons ici que le cas des codes à délai de synchronisation borné. La plupart des propositions qui suivent peuvent se démontrer pour les codes à délai de synchronisation fini. Les différences entre les deux notions sont analogues aux différences entre les notions de codes à délai de déchiffrage borné et fini. En particulier, nous le remarquerons à propos des propriétés (2) et (3) du théorème 7.3.2, non équivalentes dans le cas non rationnel : les codes à délai de synchronisation borné vérifient (2), les codes à délai de synchronisation fini vérifient (3) mais pas nécessairement (2).

Les codes que nous étudions sont donc les codes à délai de synchronisation borné usuels [8]. Un tel code est un Z-code.

Nous nous intéressons aux caractérisations des codes rationnels à délai de synchronisation borné. Nous rappelons la caractérisation connue : les codes circulaires rationnels vérifiant $F(C) \cap C^p = \emptyset$ pour un certain p sont les codes rationnels à délai de synchronisation borné. Nous démontrons une autre

caractérisation : un code circulaire rationnel C a un délai de synchronisation borné si et seulement si ${}^{\omega}C^{\omega} \cap K(C) = \emptyset$, où $K(C)$ représente le langage :

$$Ladh(C).C^*.Adh(C) \bigcup {}^{\omega}C.Adh(C) \bigcup Ladh(C).C^{\omega} \bigcup Biadh(C).$$

Cette dernière propriété, valable pour tout code à délai de synchronisation borné (même non rationnel), permet d'obtenir des propriétés de ${}^{\omega}C^{\omega}$ quand C a un délai de synchronisation borné. En particulier, ${}^{\omega}C^{\omega}$ est une bilimite mais n'est généralement pas une biadhérence. Quand l'alphabet est fini, c'est une biadhérence si et seulement si C est fini.

Comme conséquence des résultats énoncés, on montre qu'on peut décider si un code rationnel est à délai de synchronisation borné.

7.1 Définitions

Définissons les deux propriétés suivantes liées à une possibilité de synchronisation :

Définitions 7.1.1 *On définit de la façon suivante les propriétés R_ω et R_* d'un langage G inclus dans A^+ :*
R_ω : $\exists m > 0 \ \forall f \in {}^{\omega}A \ \forall f' \in A^{\omega} \ \forall v, v' \in G^m$

$$(fvv'f' \in {}^{\omega}G^{\omega} \Rightarrow fv \in {}^{\omega}G, \ v'f' \in G^{\omega})$$

R_* : $\exists m > 0 \ \forall f, f' \in A^* \ \forall v, v' \in G^m \ (fvv'f' \in G^* \Rightarrow fv, v'f' \in G^*)$.

On voit facilement que R_* implique R_ω. Cependant R_ω et R_* ne sont pas équivalentes. Par exemple, le langage $G = \{a, ab, ba, bb\}$ vérifie R_ω (pour $m = 1$, car ${}^{\omega}G = {}^{\omega}A$ et $G^{\omega} = A^{\omega}$) mais pas R_* (pour tout m, $b(b^2)^{2m}b \in G^+$, mais $(b^2)^m b \notin G^+$).

Un langage qui vérifie R_* ou R_ω peut ne pas être un code. Par exemple le langage $G = \{a^2, a^3\}$ vérifie R_* et R_ω (pour $m = 1$) mais n'est pas un code (et même G^* n'est pas libre).

Si l'on veut obtenir un décodage unique des mots de ${}^{\omega}A^{\omega}$, il faut au moins que G soit un biω-code. Même en ajoutant cette hypothèse, les propriétés R_ω et R_* ne deviennent pas équivalentes (on peut considérer par exemple $G = \{bb\}$).

Dans le cas des biω-codes, la différence entre les propriétés R_* et R_ω relève de la notion de pureté. En effet, le résultat suivant fait intervenir les Z-codes qui sont en fait les biω-codes purs.

Proposition 7.1.1 *[27] Un code vérifiant la propriété R_* est un Z-code.*

Preuve : Soit C un code vérifiant R_*. Considérons deux C-décompositions d et d' d'un mot w de A^Z. Montrons que, pour tout $i \in Z$, d'_i appartient à $d(Z)$. Il existe j, k, h tels que :

$$d_j \leq d'_{i-m}; \ d_h \leq d'_i < d_{h+1}; \ d'_{i+m} \leq d_k \ .$$

D'après R_*, $w[d_j, d'_i[$ et $w[d'_i, d_k[$ appartiennent à C^*. Or C est un code et tous les mots de la forme $w[d_n, d_{n+1}[$ sont dans C, donc d'_i est l'un des d_n. De même, tout d_n appartient à $d'(Z)$. Donc $d = d'$ et C est un Z-code. •

Cette proposition justifie la définition suivante:

Définition 7.1.2 *On appelle* **code à délai de synchronisation borné** *tout code vérifiant la propriété R_*.*

Nous verrons qu'un code à délai de synchronisation borné m est à délai de déchiffrage borné m, vers la droite et vers la gauche, et a donc les propriétés de ces codes pour le décodage des mots finis, infinis à droite et infinis à gauche. De plus, un code à délai de synchronisation borné m permet de décoder concrètement les facteurs de mots quelconques (finis, infinis ou bi-infinis), en ce sens que si l'on a su reconnaître dans un facteur d'un mot w un mot de C^{2m+h}: $v_1v_2...v_{2m+h}$, on est sûr que le facteur $v_{m+1}v_{m+2}...v_{m+h}$ fait bien partie de la décomposition unique de w. Ces codes sont donc les codes intéressants quand une partie du message codé n'est pas disponible.

Proposition 7.1.2 *Considérons un langage C inclus dans A^+ . Les conditions suivantes sont équivalentes:*
- C est un code à délai de synchronisation borné m,
- $\forall n \geq 0\ \forall p > 0\ \forall c_1, ..., c_m, c'_1, ..., c'_m, c''_1, ...c''_n, b_1, ..., b_p \in C\ \forall u, u' \in A^*$

$$uc_1...c_mc''_1...c''_nc'_1...c'_mu' = b_1...b_p \Rightarrow \exists i\ uc_1...c_m = b_1...b_i\ ,$$

$$c'_1...c'_mu' = b_{i+n+1}...b_p \text{ et } \forall h \in \{1, 2, ..., n\}\ c''_h = b_{i+h}\ .$$

La proposition qui suit donne une caractérisation des codes à délai de synchronisation borné ne faisant intervenir que des mots infinis et bi-infinis.

Proposition 7.1.3 *Un langage inclus dans A^+ est un code à délai de synchronisation borné si et seulement si c'est un Z-code vérifiant la propriété R_ω.*

Preuve: Soit C un Z-code vérifiant la propriété R_ω. Soient $f, f' \in A^*$, $v, v' \in C^m$ tels que $fvv'f' \in C^*$. Le mot ${}^\omega vfvv'f'v^\omega$ appartient à ${}^\omega C^\omega$. D'après R_ω, $v'f'v^\omega$ appartient à C^ω et ${}^\omega vfv$ à ${}^\omega C$. Comme ${}^\omega vfvv'f'v^\omega$ a une seule C-décomposition, il s'ensuit que fv et $v'f'$ sont dans C^*. Donc le code C vérifie la propriété R_*. •

Nous ne savons si l'on peut remplacer dans cette dernière proposition le terme « Z-code » par le terme « code pur ». Nous verrons que c'est possible dans le cas d'un code à deux éléments.

7.2 Propriétés des langages engendrés

Rappelons quelques définitions et notations. Soit G un langage inclus dans A^+, A^ω, ${}^\omega A$ ou ${}^\omega A^\omega$ (suivant le cas), on note:
- $F(G)$ l'ensemble des facteurs finis de G,
- $Fg(G)$ l'ensemble des facteurs gauches finis de G,
- $Fd(G)$ l'ensemble des facteurs droits finis de G.

Si G est inclus dans A^+, on note:
- $Adh(G) = \{w \in A^\omega \mid Fg(w) \subset Fg(G)\}$ l'adhérence droite de G,
- $Ladh(G) = \{w \in {}^\omega A \mid Fd(w) \subset Fd(G)\}$ l'adhérence gauche de G,
- $Biadh(G) = \{w \in {}^\omega A^\omega \mid F(w) \subset F(G)\}$ la biadhérence de G,
- $K(G) = Ladh(G).G^*.Adh(G) \bigcup {}^\omega G.Adh(G) \bigcup Ladh(G).G^\omega \bigcup Biadh(G)$.

Proposition 7.2.1 *Soit C un code à délai de synchronisation borné m. Le code C est à délai de déchiffrage borné m, vers la droite et vers la gauche, et l'ensemble $F(C) \cap C^{2m}$ est vide.*

Preuve: Démontrons la propriété P'_*. Soient $f \in A^*$, $u \in C^*$, $v, v_1, ..., v_m, v' \in C$ tels que: $vv_1...v_m f = v'u$. Alors $v^m v_1...v_m f = v^{m-1}v'u \in C$. Posons $u' = v_1...v_m f$, d'après R_*, $u' \in C^*$. Or C est un code et $vu' = v'u$, donc $v = v'$. Le code C est à délai de déchiffrage (à droite) borné m. De même à gauche.

Si $F(C) \cap C^{2m}$ n'est pas vide, il est clair qu'un langage qui vérifie R_* pour m ne peut être un code. Donc, pour un code C à délai de synchronisation borné m, l'ensemble $F(C) \cap C^{2m}$ est vide. •

Les conditions données dans la proposition précédente ne suffisent pour que le code considéré ait un délai de synchronisation borné. L'exemple suivant est de A. Restivo.

Exemple: [53] Le code $C = \{x^{2n-1}y^{2n} \mid n > 0\} \cup \{y^{2n}x^{2n+1} \mid n > 0\}$ vérifie: $\forall p > 1$ $F(C) \cap C^p = \emptyset$. C'est un Z-code bipréfixe non rationnel, et l'on a: $xy^2x^3y^4...y^{4q+2} \in C^*$, $y^2x^3y^4...x^{2q+1} \in C^q$, $y^{2q+2}x^{2q+3}...x^{4q+1} \in C^q$ et cependant $xy^2x^3y^4...x^{2q+1} \notin C^*$.

Le corollaire suivant nous permettra d'étudier la nature du langage ${}^\omega C^\omega$. Rappelons que l'on a [23]:
$Biadh(C^*) = {}^\omega C^\omega \cup K(C)$,
$K(C) = Biadh(C) \cup Ladh(C).Adh(C^*) \cup Ladh(C^*).Adh(C)$,
$Ladh(C^*).Adh(C) = Ladh(C).C^*.Adh(C) \cup {}^\omega C.Adh(C)$,
$Ladh(C).Adh(C^*) = Ladh(C).C^*.Adh(C) \cup Ladh(C).C^\omega$.

Corollaire 7.2.2 *Si C est un code à délai de synchronisation borné, on a :*

$$^{\omega}C^{\omega} \bigcap K(C) = \emptyset$$
$$^{\omega}C.Adh(C) \bigcap Ladh(C).C^{\omega} = \emptyset$$
$$Ladh(C).C^{\omega} \bigcap Biadh(C) = \emptyset$$
$$^{\omega}C.Adh(C) \bigcap Biadh(C) = \emptyset$$
$$Ladh(C).C^{*}.Adh(C) \bigcap {}^{\omega}C.Adh(C) = \emptyset$$
$$Ladh(C).C^{*}.Adh(C) \bigcap Ladh(C).C^{\omega} = \emptyset.$$

D'autre part, si $Ladh(C).C^{n}.Adh(C) \bigcap Biadh(C)$ est non vide, n est strictement inférieur à 2 fois le délai de synchronisation de C.

Preuve : Soit $w \in {}^{\omega}C^{\omega}$, $w = ...v_{-1}v_0v_1...$ où $v_i \in C$ pour tout i. Si w appartient à $K(C)$, d'après les définitions des adhérences, pour tout $m > 0$, il existe un facteur $v_{i+1}...v_{i+2m}$ de w qui est facteur d'un mot de C. Cela contredit le fait que $F(C) \cap C^{2m} = \emptyset$.

On raisonne de la même façon pour démontrer qu'il n'existe pas d'éléments dans les autres ensembles considérés. •

Cette dernière proposition permet d'énoncer les corollaires qui suivent, et de schématiser la biadhérence de C^*.

Corollaire 7.2.3 *Soit C un code à délai de synchronisation borné, $^{\omega}C^{\omega}$ est une biadhérence (c'est-à-dire un fermé de $^{\omega}A^{\omega}$) si et seulement si*

$$Adh(C) = Ladh(C) = Biadh(C) = \emptyset\ .$$

Si de plus A est fini, $^{\omega}C^{\omega}$ est une biadhérence si et seulement si C est fini.

Preuve : D'après [23], $^{\omega}C^{\omega}$ est une biadhérence si et seulement si $^{\omega}C^{\omega}$ est la biadhérence de C^*. Or $Biadh(C^*) = {}^{\omega}C^{\omega} \cup K(C)$ et $^{\omega}C^{\omega} \cap K(C) = \emptyset$, donc $^{\omega}C^{\omega}$ est une biadhérence si et seulement si $K(C)$ est vide. Si C est fini, cette condition est trivialement vérifiée. Si l'alphabet est fini, on démontre que K(C) n'est vide que si C est fini [23]. •

Remarquons que les conditions $Adh(C) = \emptyset$, $Ladh(C) = \emptyset$ et $Biadh(C) = \emptyset$ sont indépendantes quand l'alphabet est infini :

Exemples :
- Si $A = \{b_n \mid n > 0\}$, $w = b_1b_2b_3...$ et $C = Fg(w)$, on a :
$Adh(C) = \{w\}$, $Ladh(C) = Biadh(C) = \emptyset$.
- Si $A = \{a\} \cup \{b_n \mid n > 0\}$ et $C = \{b_na^n \mid n > 0\}$, on a :
$Adh(C) = \emptyset$, $Ladh(C) = {}^{\omega}a$, $Biadh(C) = {}^{\omega}a^{\omega}$.

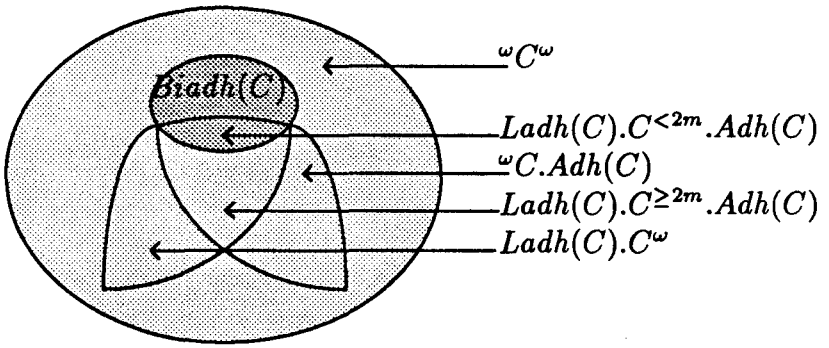


FIG. 7.1 - Représentation de $Biadh(C^*)$

Corollaire 7.2.4 *Si C est un code à délai de synchronisation borné, on a :*

$$^{\omega}C^{\omega} = Biadh(C^*) - \Big(Biadh(C) \bigcup Ladh(C).Adh(C^*) \bigcup Ladh(C^*).Adh(C)\Big)$$

et $\pi^{-1}(^{\omega}C^{\omega})$ est un G_δ-langage de A^Z.

Preuve : De la relation : $Biadh(C^*) = {}^{\omega}C^{\omega} \bigcup K(C)$, où la réunion est disjointe, on déduit la formule annoncée.

On en déduit aussi que, dans A^Z, on a : $Biadh(C^*) = \pi^{-1}(^{\omega}C^{\omega}) \bigcup K'(C)$, où la réunion est disjointe et où $K'(C) =$

$$\bigcup_{n\in Z} \sigma^n(Ladh(C)\#Adh(C^*)) \bigcup_{n\in Z} \sigma^n(Ladh(C^*)\#Adh(C)) \bigcup Biadh(C)\ .$$

$K'(C)$ est une réunion dénombrable de fermés. Le langage de A^Z : $\pi^{-1}(^{\omega}C^{\omega})$ est donc la différence entre le fermé $Biadh(C^*)$ et son F_σ-sous-ensemble $K'(C)$, c'est donc un G_δ-langage. •

La formule donnée ne permet pas d'affirmer que $^{\omega}C^{\omega}$ est un G_δ-langage de $^{\omega}A^{\omega}$. Il se peut même en fait que $^{\omega}C^{\omega}$ n'en soit pas un, et donc que $^{\omega}C^{\omega}$ diffère de $\bigcap {}^{\omega}AC^iA^{\omega}$.

Exemple : Le code $C = a^*b$ est à délai de synchronisation 1. Clairement $^{\omega}ab^{\omega}$ appartient à $\bigcap {}^{\omega}AC^iA^{\omega}$ mais pas à $^{\omega}C^{\omega}$.

Si $^{\omega}C^{\omega}$ était un G_δ-langage, $K(C) = {}^{\omega}abA^{\omega} + {}^{\omega}Aba^{\omega} + {}^{\omega}a^{\omega}$ serait un F_σ-langage : $K(C) = \cup F_i$. Le mot $^{\omega}a(ba)^{\omega}$ appartiendrait à un fermé F_i. Le mot $^{\omega}(ba)^{\omega}$ appartiendrait aussi à F_i, donc à $K(C)$, ce qui n'est pas le cas.

Dans cet exemple, ${}^{\omega}C^{\omega}$ est une bilimite : ${}^{\omega}C^{\omega} = Bilim(b(a^*b)^*)$. La question suivante vient alors naturellement : si C est un code à délai de synchronisation borné, le langage ${}^{\omega}C^{\omega}$ est-il une bilimite? Le résultat suivant renforce l'analogie avec les codes à délai de déchiffrage borné.

Théorème 7.2.5 *Si C est un code à délai de synchronisation borné, ${}^{\omega}C^{\omega}$ est une bilimite. Plus précisément, si m est un délai de synchronisation de C, pour tout $i \geq m$, posons $P_i = C^i - C^iA^+$, $S_i = C^i - A^+C^i$, $G_i = S_iP_i$ et $G = \cup_{i \geq m} G_i$, on a : ${}^{\omega}C^{\omega} = Bilim(G)$.*

Preuve : Montrons d'abord l'inclusion : ${}^{\omega}C^{\omega} \subset Bilim(G)$. Soit $(c_n)_{n \in Z}$ une suite de mots de C, et considérons le Z-mot w défini par $w_+ = c_0c_1c_2...$ et $w_- = ...c_{-2}c_{-1}$. Pour tout $i \geq m$, $c_0c_1...c_{i-1}$ est de la forme u_ix où $u_i \in P_i$, $x \in A^*$ et $|u_i| \geq i$. De même, $c_i...c_{-2}c_{-1} = yv_i$ où $v_i \in S_i$ $y \in A^*$ et $|v_i| \geq i$. Clairement w appartient à la bilimite de G.

Pour montrer l'inclusion : $Bilim(G) \subset {}^{\omega}C^{\omega}$, nous remarquons d'abord le fait suivant :
Fait : Si un élément de C^i est facteur gauche strict d'un élément de P_n, alors $i < n$.
Soit w un Z-mot bilimite d'une suite strictement croissante $(r_n)_{n \in N}$ d'éléments de G. Pour tout n, il existe i_n et une factorisation de r_n tels que $r_n = v_nu_n$, $v_n \in S_{i_n}$ et $u_n \in P_{i_n}$; il existe aussi une suite strictement croissante $(p_n)_{n \in N}$ et une suite strictement décroissante $(q_n)_{n \in N}$ telles que $w[q_n, p_n[= r_n$ pour tout n. Considérons trois indices successifs $n, n+1, n+2$, et les découpages induits sur w par r_n, r_{n+1}, r_{n+2} ; posons en particulier : $w[q_{n+1}, q_n[= x$ et $w[p_n, p_{n+1}[= y$. On a soit $xv_n < v_{n+1}$, soit $v_{n+1} \leq xv_n$. Pour des raisons de symétrie, nous n'étudierons que ce dernier cas. Puisque le délai de synchronisation de C est m et d'après la proposition 7.1.2, les découpages sur C de $w[q_{n+1}, p_{n+1}[$ et de $w[q_n, p_n[$ coïncident au moins pour les $2(i_n - m)$ facteurs centraux : $c_{-i_n+m}, ..., c_{-1}, c_0, ..., c_{i_n-m-1}$ de v_nu_n. De même, les découpages sur C de $w[q_{n+2}, p_{n+2}[$ et de $w[q_{n+1}, p_{n+1}[$ coïncident au moins pour les $2(i_{n+1} - m)$ facteurs centraux : $c'_{-i_{n+1}+m}, ..., c'_{-1}, c'_0, ..., c'_{i_{n+1}-m-1}$ de $v_{n+1}u_{n+1}$. Il existe h tel que : $c_{-i_n+m} = c'_h, ..., c_{-1} = c'_{h+i_n-m-1}, c_0 = c'_{h+i_n-m}, ..., c_{i_n-m-1} = c'_{h+2(i_n-m)-1}$. Nous allons montrer que $i_{n+1} > i_n$ et que h vérifie $-i_{n+1}+m < h$ et $h+2(i_n-m)-1 < i_{n+1}-m-1$. Ceci signifie que, pour les facteurs reconnus, il y a un gain de 1 facteur au moins à droite et à gauche, quand on passe de l'indice n à l'indice $n+1$.
Dans le cas considéré, il est clair que :
- soit $h > 0$ et, d'après le fait remarqué, $i_{n+1} > h+(i_n-m)+i_n$, donc $i_{n+1} > i_n$ et h vérifie la propriété annoncée ;
- soit $-i_n + m \leq h \leq 0$ et, d'après le fait remarqué, $i_{n+1} > i_n + (i_n - m) + h$, donc $i_{n+1} > i_n$ et donc h vérifie la propriété annoncée.

Il nous reste à montrer que w représente un élément de ${}^{\omega}C^{\omega}$. Sans perte de généralité, on peut supposer que u_0 est facteur gauche de w_+ et v_0 facteur droit de w_-. En considérant l'indice 1 et puisque le code est à délai de synchronisation borné, on voit que, si w représente un élément de ${}^{\omega}C^{\omega}$, alors $w_+ \in C^{\omega}$ et $w_- \in {}^{\omega}C$.

D'après la démonstration précédente, à chaque pas on obtient un facteur gauche de w_+ : $d_n \in C^n$, et $d_{n+1} \in d_nC$; il s'ensuit que $w_+ = Lim(d_n)$ appartient à C^{ω}. De même, on voit que $w_- \in {}^{\omega}C$. Donc w représente un élément de ${}^{\omega}C^{\omega}$. •

7.3 Etude du cas rationnel

Nous avons vu qu'un code C à délai de synchronisation borné m est un Z-code qui vérifie $F(C) \cap C^{2m} = \emptyset$. Dans le cas rationnel, Lassez, Restivo et Do Long Van ont démontré la réciproque [53] [34] [27]. Nous rappelons cette caractérisation :

Proposition 7.3.1 *Soit C un langage rationnel inclus dans A^+. Les conditions suivantes sont équivalentes :*
(1) - C est un code à délai de synchronisation borné
(2) - C est un code circulaire vérifiant : $\exists p > 0 \;\; F(C) \cap C^p = \emptyset$.

L'objet de la proposition suivante est de donner une caractérisation en termes de mots infinis.

Nous avons vu qu'un code à délai de synchronisation borné vérifie la condition : ${}^{\omega}C^{\omega} \cap K(C) = \emptyset$. Cette condition ne suffit pas pour qu'un code rationnel C soit à délai de synchronisation borné : le code fini $\{a, ab, bb\}$ n'est pas un Z-code.

Un code à délai de synchronisation borné est un Z-code. Mais un Z-code, même rationnel, peut ne pas être à délai de synchronisation borné. Si $C = ab^*d + a + b$, le délai (de déchiffrage) de a est infini.

Dans le cas non rationnel, un Z-code vérifiant la condition : ${}^{\omega}C^{\omega} \cap K(C) = \emptyset$ peut ne pas être à délai de synchronisation borné (ni même fini) :

Exemple: Le langage $C = \{a\} \cup c^*b \cup \{a(c^nb)^nd \mid n > 0\}$ est un Z-code, non rationnel, vérifiant la condition : ${}^{\omega}C^{\omega} \cap K(C) = \emptyset$. En effet, $Ladh(C) = {}^{\omega}cb \cup {}^{\omega}cbd$, $Adh(C) = c^{\omega} \cup ac^{\omega}$, $Biadh(C) = {}^{\omega}c^{\omega}$ et aucune puissance de c n'est dans C. Cependant, le délai de a est infini.

Cependant, on a :

Proposition 7.3.2 *Soit C un langage rationnel inclus dans A^+. Les conditions suivantes sont équivalentes :*
(1) - C est un code à délai de synchronisation borné
(3) - C est un code circulaire vérifiant : ${}^\omega C^\omega \cap K(C) = \emptyset$.

Preuve :
- (1) $\Rightarrow$ (3) : Si C est un code (rationnel ou non) à délai de synchronisation m, c'est un Z-code donc un code circulaire (propositions 7.1.1 et 4.6.2). De plus, on vient d'établir à la proposition précédente que ${}^\omega C^\omega \cap K(C) = \emptyset$.
- (3) $\Rightarrow$ (1) Considérons un code circulaire rationnel C, un automate Ω_0 *ad hoc* pour C et son automate dérivé $\Omega = (Q, q_0, q_0, R)$. L'automate Ω reconnaît ${}^\omega C^\omega$. Soit m son nombre d'états.

Supposons que C ne soit pas à délai de synchronisation m. Nous allons montrer que ${}^\omega C^\omega \cap K(C) \neq \emptyset$.

Il existe $f, f' \in A^*$, $v_1, v_2, ..., v_{2m} \in C$ tels que $fv_1...v_{2m}f' \in C^+$, $fv_1...v_m \notin C^+$ ou $v_{m+1}...v_{2m}f' \notin C^+$. Sur l'automate il existe une lecture de q_0 à q_0 du mot $w = fv_1...v_{2m}f'$. Dans cette lecture, notons q_{i_0} (resp. q_{i_j}) l'état atteint après la lecture de f (resp. $fv...v_j$). Il existe des entiers j, j', k, k' tels que $0 \leq j < j' \leq m \leq k < k' \leq 2m$ et $q_{i_j} = q_{i_{j'}}$, $q_{i_k} = q_{i_{k'}}$. Posons $x = v_{j+1}...v_{j'}$, $z = v_{j'+1}...v_k$ et $y = v_{k+1}...v_{k'}$. Le mot w de C^+ a une unique C-factorisation : $w = u_0...u_n$ où u_i appartient à C pour tout i. Dans cette factorisation, on note $u_p...u_q$ (resp. $u_{p'}...u_{q'}$) le plus court mot qui recouvre x (resp. y). Selon les positions respectives de p, q, p', q', on a des schémas différents et le mot ${}^\omega xzy^\omega$ qui appartient à ${}^\omega C^\omega$ appartiendra aussi à $K(C)$. En effet :
- si $p = q = p' = q'$, ${}^\omega xzy^\omega$ appartient à $Biadh(C)$,
- si $p = q \leq p' < q'$, ${}^\omega xzy^\omega$ appartient à $Ladh(C).C^\omega$,
- si $p = q < p' = q'$, ${}^\omega xzy^\omega$ appartient à $Ladh(C).C^*Adh(C)$,
- si $p < q \leq p' = q'$, ${}^\omega xzy^\omega$ appartient à ${}^\omega C.Adh(C)$,
- le cas $p < q \leq p' < q'$ ne se produit pas. En effet, si $p < q \leq p' < q'$, ${}^\omega xzy^\omega$ a deux C-décompositions; C étant circulaire, ces décompositions sont confondues et $q_{i_j} = q_{i_k} = q_0$. On en déduit que $fv_1...v_j$ appartient à C^+ et donc $fv_1...v_m$ aussi. De même, $v_{m+1}...v_{2m}f'$ appartient à C^+. Cela contredit les hypothèses. •

Remarquons que l'on vient de démontrer que le délai du code est au plus le nombre d'états de l'automate Ω considéré.

Ici encore, comme dans le cas des codes à délai borné, nous avons des corollaires immédiats.

Corollaire 7.3.3 *Tout code circulaire fini est à délai de synchronisation borné.*

Preuve : $K(C)$ est vide quand C est fini. •

Corollaire 7.3.4 *On peut décider si un langage rationnel est un code à délai de synchronisation borné, et donner un majorant de ce délai.*

Preuve : On sait décider si un langage rationnel C est un code circulaire et si le langage rationnel bi-infinitaire ${}^{\omega}C^{\omega} \cap K(C)$ est vide ou non.

Si C est un code à délai de synchronisation borné, le nombre d'états de l'automate dérivé d'un automate pour C majore le délai de synchronisation de C. •

A propos des caractérisations 7.3.1 et 7.3.2, on peut se demander si la seconde ne se déduit pas facilement de la première, d'autant plus que les preuves des conditions nécessaires sont simples, et celles des conditions suffisantes nettement plus compliquées. Nous verrons qu'en fait, cela peut se faire, mais ce n'est pas immédiat.

Remarquons cependant que la seconde caractérisation, d'une autre nature que la première (puisqu'elle fait intervenir les mots bi-infinis) est, en un certain sens, plus intéressante que celle-ci. En effet :
- ces caractérisations diffèrent par le nombre de quantificateurs,
- l'implication : (2) $\Rightarrow$ (3) se démontre aisément par contraposition,
- l'implication : (3) $\Rightarrow$ (2) ne semble pas simple à démontrer. Le moyen le plus économique semble être : (3) $\Rightarrow$ (1) et (1) $\Rightarrow$ (2) ,
- l'implication : (3) $\Rightarrow$ (2) nécessite l'hypothèse de rationnalité (voir l'exemple ci-après),
- la condition (2) est vérifiée par un code à délai de synchronisation borné (il suffit de prendre $p = 2m$), mais elle peut ne pas l'être par un code à délai de synchronisation fini (voir l'exemple ci-après).

Comme condition suffisante, la propriété (3) est donc plus intéressante que la propriété (2).

Exemple: Le langage $C = c^*b \cup \{a(c^nb)^nd \mid n > 0\}$ est un Z-code, donc un code circulaire, non rationnel, ne vérifiant pas (2). Cependant, il vérifie (3) puisque $Ladh(C) = {}^{\omega}cb \cup {}^{\omega}cbd$, $Adh(C) = c^{\omega} \cup ac^{\omega}$, $Biadh(C) = {}^{\omega}c^{\omega}$ et aucune puissance de c n'est dans C. Donc l'implication : (3) $\Rightarrow$ (2) nécessite l'hypothèse de rationnalité. D'autre part, C est à délai de synchronisation fini (au sens R'_* : $m(a(c^nb)^nd) = p(a(c^nb)^nd) = 0$, $m(c^nb) = 1$, $m'(c^nb) = n - 1$).

Cependant, on a le lemme suivant dont le résultat aurait pu servir à démontrer la proposition 7.3.2.

Lemme 7.3.5 *Soit G un langage rationnel inclus dans A^+. Les conditions suivantes sont équivalentes :*
(2') - G vérifie : $\exists p > 0 \;\; F(G) \cap G^p = \emptyset$
(3') - G vérifie : ${}^{\omega}G^{\omega} \cap K(G) = \emptyset$,
(4') - G vérifie : ${}^{\omega}G^{\omega} \cap Biadh(G) = \emptyset$.

Preuve : (3') $\Rightarrow$ (4') est évident.
(2') $\Rightarrow$ (3') se démontre par contraposition, exactement comme le corollaire 7.2.2 (même si G n'est pas rationnel).
(4') $\Rightarrow$ (2') se démontre aussi par contraposition :

Soit G un langage rationnel et $\Omega = (Q, q_0, q_0, R)$ l'automate dérivé d'un automate *ad hoc* pour G. Soit d le nombre de ses états. Si (2') n'est pas vérifiée, pour tout $p > 0$, il existe $v_{-p}, ..., v_{-1}, v_1, ..., v_p, u_p \in G$ tels que $w_p = v_{-p}...v_{-1}v_1...v_p \in F(u_p)$; notons : $u_p = r_p w_p s_p$. Considérons une lecture réussie des mots $v_{-p}, ..., v_{-1}, v_1, ..., v_p$ et u_p sur Ω.

On peut toujours choisir les mots v_j de longueur plus petite que d^2. En effet, si ce n'était pas le cas, le mot v_j pourrait s'écrire $t_1t_2t_3$ où les états atteints après avoir lu $r_p v_{-p}...v_{j-1}t_1$ et $r_p v_{-p}...v_{j-1}t_1t_2$ dans la lecture considérée du mot u_p sont égaux, et où les états atteints après avoir lu t_1 et t_1t_2 dans la lecture considérée du mot v_j sont égaux ; on pourrait alors remplacer v_j par t_1t_3.

Le nombre de v_1 et de v_{-1} possibles est donc fini. Il existe une infinité de p ayant le même couple (v_{-1}, v_1). On choisit un tel couple (v_{-1}, v_1) et on note J_1 l'ensemble infini des p qui lui correspondent. Par récurrence, on peut choisir un couple (v_{-n}, v_n) tel que l'ensemble J_n des $p \in J_{n-1}$ qui correspondent à ce couple soit infini. On construit ainsi un mot bi-infini $w = ...v_{-2}v_{-1}v_1v_2...$ de ${}^{\omega}G^{\omega}$, dont tous les facteurs $v_{-n}...v_{-1}v_1...v_n$ appartiennent à $F(G)$. Ce mot w est dans $Biadh(G)$. On a donc : ${}^{\omega}G^{\omega} \cap Biadh(G) \neq \emptyset$. •

On obtient donc le théorème suivant :

Théorème 7.3.6 *Soit C un langage rationnel inclus dans A^+. Les conditions suivantes sont équivalentes :*
- C est un code à délai de synchronisation borné
- C est un code circulaire vérifiant : $\exists p > 0 \;\; F(C) \cap C^p = \emptyset$
- C est un code circulaire vérifiant : ${}^{\omega}C^{\omega} \cap Biadh(C) = \emptyset$.

Chapitre 8

Etude du cas fini

Nous avons vu, au chapitre 4, que les familles de codes étudiées sont distinctes. Dans le chapitre 5, nous avons étudié les familles de codes rationnels correspondantes. Quand c'était possible, nous avons donné des exemples de codes finis pour les distinguer. Cela est possible sauf dans deux cas ; en effet :
- les ω-codes finis sont à délai borné,
- les Z-codes finis sont à délai de synchronisation borné.

Par contre, la famille des biω-codes finis se distingue bien des autres familles. Cela est sans doute en relation avec les faits (déplorables) suivants :
- l'ensemble des mots bi-infinis qui ont plusieurs factorisations sur un langage fini donné n'est pas toujours rationnel,
- l'ensemble des mots bi-infinis qui ont plusieurs décompositions mais une seule factorisation sur un langage fini donné n'est pas toujours fini.

Dans la plupart des cas, pour deux familles de codes finis distinctes, nous donnons un exemple de code à trois éléments pouvant les distinguer. Dans l'un des cas, nous n'avons pas d'exemple ; ceci nous mène à émettre la conjecture suivante : tout code pur à trois éléments est un π-code.

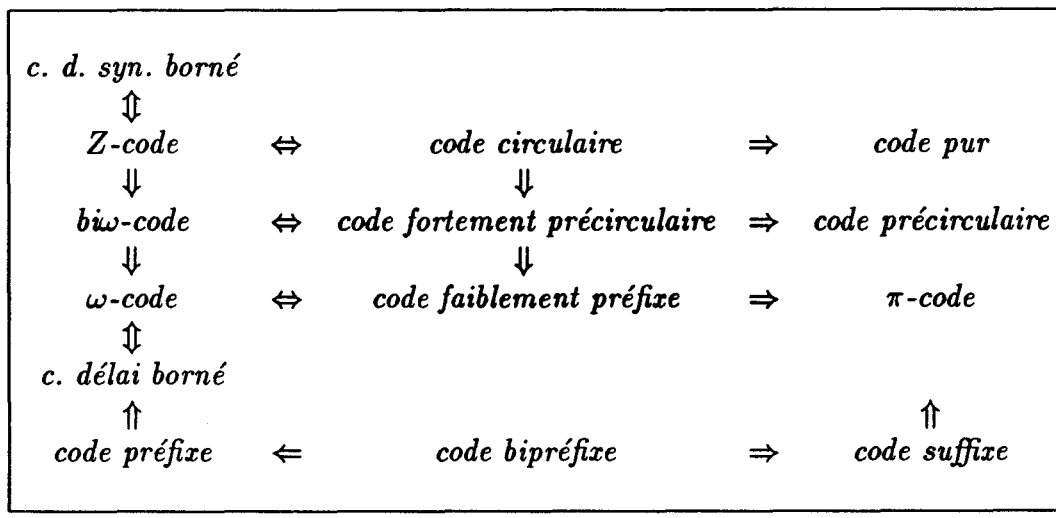
Nous étudions ensuite les codes à deux éléments. Sur la base de résultats de A. Lentin, M. P. Schutzenberger [38], de E. Barbin Le Rest et M. Le Rest [1], nous caractérisons les diverses familles de codes à deux éléments et nous dénombrons les décompositions des mots bi-infinis sur les codes à deux éléments.

8.1 Codes finis et codes à trois éléments

Nous rappelons les inclusions existant entre les différentes classes de codes finis étudiées.

Théorème 8.1.1 *Le schéma suivant concerne les* **codes finis**, *les implications y sont strictes et il n'y a pas d'autres implications que celles indiquées :*

Pour les codes finis :



Les exemples qui ont permis d'énoncer ces résultats ont été choisis de cardinal inférieur ou égal à 3, sauf pour l'un d'entre eux. Nous les rappelons ici.

Exemples :
- le code préfixe $C = \{a, ba\}$ n'est pas suffixe,
- le code bipréfixe et pur $\{ab, ba\}$ n'est pas précirculaire,
- le biω-code bipréfixe $\{a, bab\}$ n'est pas pur,
- le code précirculaire, de délai gauche 1 $\{a, a^2ba, aba^2ba\}$ n'est ni un π-code , ni pur,
- le code suffixe $C = \{ab, aba, ba^2\}$ n'est pas faiblement préfixe,
- le code précirculaire et bipréfixe $C = \{a^2, ab, b^2\}$ n'est pas un biω-code,
- le ω-code précirculaire $C = \{a, ba, b^2\}$ n'est pas un biω-code,
- le code suffixe précirculaire $C = \{c, abab, cab\}$ n'est ni pur, ni fortement précirculaire,
- le code pur $\{a, ab, cab, bca\}$, de délai gauche 1, n'est ni un π-code, ni précirculaire,

Rappelons que l'ensemble des mots bi-infinis qui ont plusieurs factorisations sur un langage fini donné n'est pas toujours rationnel, même dans le cas d'un code à trois éléments (par exemple pour le code $C = \{ab, ba, aa\}$. Du point de vue qui nous intéresse, il semble donc que les codes à trois éléments ne se différencient des codes finis que par la propriété suivante (démontrée par J. Karhumäki) et éventuellement par la conjecture que nous proposons :

Proposition 8.1.2 *[32] Tout code à trois éléments est à délai de déchiffrage borné vers la droite ou vers la gauche.*

Conjecture 1 *Tout code pur à trois éléments est un π-code.*

8.2 Codes à deux éléments

8.2.1 Codes et ω-codes

Rappelons quelques caractérisations de certains codes à deux éléments.

Proposition 8.2.1 *[45] Soit $C \subset A^+$ un langage à deux éléments u et v, les conditions suivantes sont équivalentes:*
- C n'est pas un code,
- $uv = vu$,
- Il existe une puissance de u qui est une puissance de v,
- les racines primitives de u et de v sont égales.

Proposition 8.2.2 *Tout code à deux éléments est un code à délai borné.*

Preuve: Il suffit de démontrer qu'un code à deux éléments est un ω-code.

Considérons $C = \{u, v\}$ où u et v sont primitifs et distincts. C est un code. Si C n'est pas un ω-code, on peut supposer que $|u| > |v|$. Soit $n > 0$ le plus grand entier tel que $v^n \leq u$. Posons $u = v^n u'$. Il existe un mot w de C^ω ayant une C-factorisation commençant par u et une autre commençant par v. Si $w = v^\omega$, u' est clairement un facteur gauche de v. Sinon, $w = u... = v^p u...$ où $p > 0$, donc $w = v^n u'... = v^{p+n} u'...$ donc u' est un facteur gauche de v, et $n > 0$. On a alors $w = v^n u' v... = v^{p+n} u'...$, v étant primitif, ce n'est possible que si $u' = \varepsilon$. D'où $u = v^n$, il y a contradiction.

Considérons $C' = \{u', v'\}$ où u', ou v', n'est pas primitif. Soient u et v les racines primitives de u' et v'. Si C' est un code, on a : $u \neq v$, $C = \{u, v\}$ est un code à deux éléments, et donc C est un ω-code. Si C' n'était pas un ω-code, $C = \{u, v\}$ ne pourrait être un ω-code que si $u = v$. Donc C' est un ω-code. •

Remarquons que le délai d'un code à deux éléments peut être quelconque.

Exemple: Le code $C = \{ab, (ab)^n ba\}$ est de délai n.

8.2.2 Biω-codes

Les travaux de A. Lentin et M. P. Schutzenberger, et ceux de E. Barbin Le Rest et de M. Le Rest vont nous permettre d'étudier les biω-codes à deux éléments. Voici une partie de leurs résultats:

Proposition 8.2.3 *[38] Soit $C \subset A^+$ un code à deux éléments u et v, C est un code pur si et seulement si $u^*v \cup v^*u$ ne possède pas d'élément imprimitif.*

Proposition 8.2.4 *[38] Soit $C \subset A^+$ un code à deux éléments u et v primitifs. Dans $u^+v \cup v^+u$ il y a au plus un élément imprimitif. Il n'y en a pas si u et v sont conjugués.*

Proposition 8.2.5 *[1] Soit $C = \{u, v\}$, où u et v sont deux mots primitifs non conjugués tels que $|u| \geq |v|$. S'il existe un mot α de C^+, de longueur supérieure ou égale à $2|u| + 2|v|$, qui a une C-interprétation non adjacente à sa C-factorisation t alors :*
- (u, u, u) n'apparaît pas dans t,
- si (u, u) apparaît dans t, $\alpha \in (\varepsilon + u + u^2)v(u^2v)^*(\varepsilon + u + u^2)$ et u^2v est imprimitif,
- si (u, u) n'apparaît pas dans t et si $(u, v, ..., v, u)$ y apparaît (avec j fois v), uv^j est imprimitif et $\alpha \in v^*(uv^j)^*uv^*$,
- si $\alpha = v^n uv^p$, il existe j tel que uv^j soit imprimitif, $(v^n)^{-1}\alpha \in Fg((uv^j)^+)$ et $\alpha(v^p)^{-1} \in Fd((v^ju)^+)$.

Lemme 8.2.6 *Soient u et v deux mots primitifs non conjugués. Le langage $C = \{u, v\}$ est un biω-code. Au plus un mot bi-infini a des représentants ayant plusieurs C-décompositions.*

Preuve : Soit w un Z-mot ayant plusieurs C-décompositions. Le langage C est un code, donc un ω-code et un $\tilde{\omega}$-code, donc ces décompositions ne sont pas adjacentes (*i. e.* elles n'ont pas de point commun). Regardons l'une de ces décompositions et la factorisation associée. Supposons que $|u| \geq |v|$. D'après la proposition précédente, (u, u, u) n'y apparaît pas.

Si (u, u) y apparaît, u^2v est imprimitif, et tout facteur de w, appartenant à C^* et apparaissant dans la décomposition considérée est de la forme $(\varepsilon + u + u^2)v(u^2v)^*(\varepsilon + u + u^2)$. La factorisation considérée ne peut être que ${}^\omega(u, u, v)^\omega$ et alors w est un représentant de ${}^\omega(u^2v)^\omega$.

Si (u, u) n'apparaît pas, et que $(u, v, ..., v, u)$ apparaisse (avec j fois v), uv^j est imprimitif, et tout facteur de w, appartenant à C^* et apparaissant dans la décomposition considérée est de la forme $v^*(uv^j)^*uv^*$. La factorisation considérée ne peut être que ${}^\omega(u, (v)^j)^\omega$ et alors w est un représentant de ${}^\omega(uv^j)^\omega$.

Si la factorisation considérée était ${}^\omega(v), u, (v)^\omega$, il existerait j tel que uv^j soit imprimitif, et tout facteur de w, assez long, appartenant à C^* et apparaissant dans la décomposition considérée ferait apparaître deux u. Ce cas ne se présente donc pas.

Comme il ne peut y avoir qu'un élément imprimitif dans $u^+v \cup uv^+$, les cas considérés sont exclusifs, c'est-à-dire que toutes les décompositions de w relèvent du même cas. Il s'ensuit que le mot bi-infini de représentant w est

unique, et qu'il n'a qu'une seule C-factorisation. •

Corollaire 8.2.7 *Soient u et v deux mots primitifs non conjugués. Le langage $C = \{u^n, v^p\}$ est un biω-code. Au plus trois mots bi-infinis ont des représentants ayant plusieurs C-décompositions.*

Preuve : D'après le lemme précédent, $C_0 = \{u, v\}$ est un biω-code. Le langage C est composé de ce code par le biω-code sur l'alphabet $\{a, b\}$: $\{a^n, b^p\}$. C est donc un biω-code et tout mot a au plus une C-factorisation.

Supposons que $|u| \geq |v|$. Soit w un Z-mot ayant plusieurs C-décompositions. Le langage C est un ω-code et un $\tilde{\omega}$-code, donc ces décompositions ne sont pas adjacentes. Si il existe un u^n et un v^p dans la C-factorisation de w, d'après la proposition 8.2.5 et la démonstration du lemme précédent, ce n'est possible que dans les cas (exclusifs) suivants :
- $n = 2, p = 1, u^2v$ est imprimitif, la C-factorisation est ${}^{\omega}(u, u, v)^{\omega}$ et w est un représentant de ${}^{\omega}(u^2v)^{\omega}$,
- $n = 1, uv^p$ est imprimitif, la C-factorisation est ${}^{\omega}(u, v, ..., v)^{\omega}$ (où v apparaît p fois) et w est un représentant de ${}^{\omega}(uv^p)^{\omega}$.

Si u^n n'apparaît pas dans la C-factorisation de w, w est un représentant de ${}^{\omega}u^{\omega}$ qui diffère du mot précédent, et w a n C-décompositions associées à son unique C-factorisation ${}^{\omega}(u^n)^{\omega}$.

De même si v^p n'apparaît pas dans la C-factorisation de w, w est un représentant de ${}^{\omega}v^{\omega}$ qui diffère des mots précédents et w a p C-décompositions. •

Théorème 8.2.8 *Soit $C \subset A^+$ un langage à deux éléments u et v, les conditions suivantes sont équivalentes :*
1 - C est un code précirculaire,
2 - C est un biω-code,
3 - les racines primitives de u et de v ne sont pas conjuguées,
4 - pour tout i, pour tout j, u^i et v^j ne sont pas conjugués,
5 - pour tout conjugué u' de u, l'ensemble $\{u', v\}$ est un code à deux éléments,
6 - pour tout conjugué u' de u, l'ensemble $\{u', v\}$ est un biω-code à deux éléments.

Preuve : $6 \Rightarrow 5$ et $2 \Rightarrow 1$ sont immédiats.
L'équivalence : $3 \Leftrightarrow 4$ se déduit aisément de la proposition 8.2.3.
$5 \Rightarrow 3$: Si $\{u', v\}$ est un code, les racines de u' et v sont différentes.
$1 \Rightarrow 5$: Si $\{u', v\}$ n'est pas un code, il existe i et $j \neq 0$ tels que $u'^i = v^j$. Alors u^i et v^j sont des factorisations différentes de mots conjugués, donc $\{u, v\}$ n'est pas un code précirculaire.
$3 \Rightarrow 2$: C'est le lemme précédent.
$5 \Rightarrow 6$ parce que $5 \Rightarrow 3$ et $5 \Rightarrow 2$. •

Corollaire 8.2.9 *Un code pur à deux éléments qui vérifie la propriété R_ω est à délai de synchronisation borné.*

Preuve: Il suffit de démontrer qu'un code pur: $C = \{u, v\}$ où u et v sont conjugués ne peut vérifier une condition R_ω. Posons $u = xy$ et $v = yx$. Pour tout $m > 0$, le mot ${}^\omega uv^{2m+1}u^\omega = {}^\omega uyu^{2m}xu^\omega$ appartient à ${}^\omega C^\omega$ mais le mot ${}^\omega uyu^m$ n'appartient pas à ${}^\omega C$, donc m n'est pas un délai de synchronisation pour C. •

8.2.3 Z-codes

De l'étude des biω-code, on peut déduire une caractérisation des codes circulaires à deux éléments.

Proposition 8.2.10 *Soit $C \subset A^+$ un langage à deux éléments u et v, C est un code circulaire (ou un Z-code, ou un code à délai de synchronisation borné) si et seulement si les deux propriétés suivantes sont vérifiées:*
- $u^*v \cup v^*u$ ne possède pas d'élément imprimitif,
- u et v ne sont pas conjugués.

Preuve: Un code circulaire, c'est un code précirculaire pur. Il suffit d'utiliser les résultats des propositions 8.2.1, 8.2.3 et 8.2.8. •

Do Long Van, Nguyen Huong Lam et Phan Trung Huy ont montré ce résultat sous la forme plus précise. Cet énoncé peut aussi être obtenu à partir de la proposition 8.2.5.

Théorème 8.2.11 *[27] Soit $C \subset A^+$ un langage à deux éléments u et v où $|u| \leq |v|$, C est un Z-code si et seulement si aucune des propriétés suivantes n'est vérifiée:*
- u ou v est imprimitif,
- v^2u est un carré,
- vu^i est imprimitif pour un $i \leq \frac{|v|}{|u|} + 1$,
- u et v sont conjugués.

8.2.4 Dénombrement

On a dénombré les mots bi-infinis ayant plusieurs décompositions sur un biω-code à deux éléments, on peut aussi dénombrer les mots bi-infinis ayant plusieurs factorisations sur un langage à deux éléments.

Proposition 8.2.12 *[1] Soit $C = \{u, v\}$, où u et v sont deux mots primitifs conjugués. Le mot uv n'a qu'une seule C-interprétation, celle qui se déduit de sa C-factorisation.*

Proposition 8.2.13 *Considérons le code $C = \{u^n, v^p\}$ où u et v sont deux éléments primitifs conjugués distincts. Seul le mot ${}^\omega u^\omega$ a des représentants ayant plusieurs C-décompositions ; ce mot a exactement deux C-factorisations et ses représentants ont $(n+p)$ C-décompositions.*

Preuve : Le mot ${}^\omega u^\omega$ coïncide avec ${}^\omega v^\omega$ et a donc au moins deux C-factorisations et (n+p) C-décompositions.

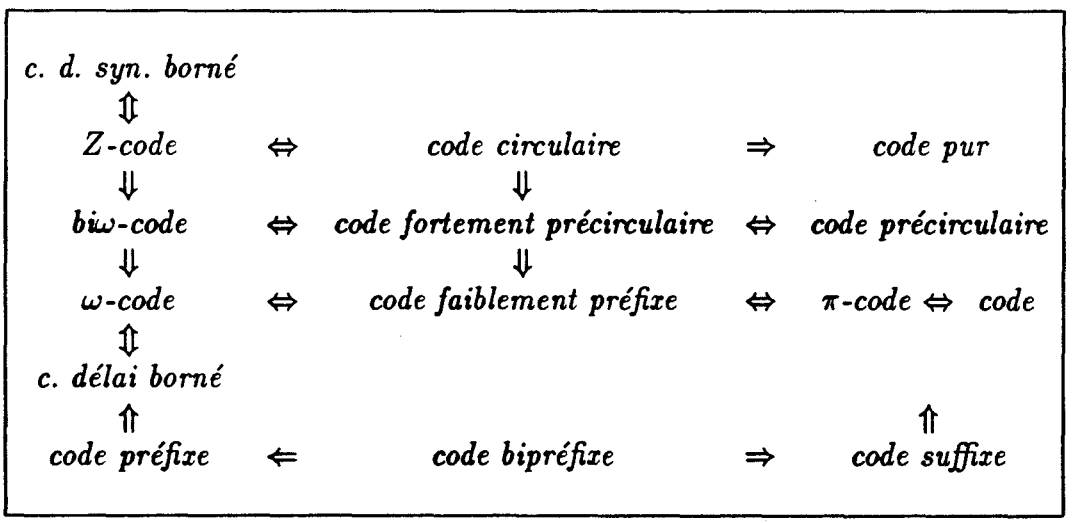
Considérons un Z-mot ayant une ou plusieurs C-décompositions. Si u^n et v^p apparaissent dans l'une des C-factorisations associées, (u^n, v^p) ou (v^p, u^n) y sont des facteurs. D'après la proposition précédente, et puisque $|u| = |v|$, les C-décompositions coïncident. Donc seul les représentants du mot ${}^\omega u^\omega$ peuvent avoir plusieurs C-décompositions, et ses seules C-factorisations sont ${}^\omega(u)^\omega$ et ${}^\omega(v)^\omega$. A la première sont associées exactement n C-décompositions et à la seconde p C-décompositions. •

8.2.5 Résultats

Les tableaux suivants résument les résultats.

Proposition 8.2.14 *Le schéma suivant concerne les* **codes à deux éléments**, *les implications y sont strictes et il n'y a pas d'autres implications que celles indiquées :*

Pour les codes à deux éléments :



Théorème 8.2.15 *Le tableau suivant concerne les* **codes à deux éléments**, *$C = \{u^n, v^p\}$ où u et v sont primitifs. La première colonne donne les mots*

ayant plusieurs C-factorisations, la deuxième le nombre de leurs C-factorisations, la troisième les mots dont les représentants ont plusieurs C-décompositions et la quatrième le nombre de ces C-décompositions. Si $u^+v \cup uv^+$ a un élément imprimitif, celui-ci est noté u^iv^j, et q est tel que $u^iv^j = \sqrt{u^iv^j}^q$. Si $u^+v \cup uv^+$ n'a que des éléments primitifs, la ligne correspondante du tableau disparaît.

Z-code				
biω-code non Z-code			$^\omega u^\omega$ $^\omega v^\omega$ $^\omega(u^iv^j)^\omega$	n p q
code non biω-code	$^\omega u^\omega$	2	$^\omega u^\omega$	$n+p$
non code $(n \neq p)$	$^\omega u^\omega$	∞	$^\omega u^\omega$	∞

Chapitre 9

Composition ; Maximalité

Ce chapitre est un catalogue des propriétés que l'on peut obtenir, pour les codes que nous étudions, dans le domaine de la composition et de la maximalité, à partir de démonstrations exemplaires. Il n'y a donc ici pas beaucoup d'innovation.

En ce qui concerne la composition des codes, nous avons repéré les familles de codes pour lesquelles la composition de codes composables est une opération interne. Curieusement, une seule famille étudiée n'a pas cette propriété : celle des π-codes.

Soit P une propriété et C un code vérifiant la propriété P. Peu de résultats sont acquis en ce qui concerne la détermination d'un code maximal parmi les codes vérifiant P et contenant C. L'utilisation du lemme de Zorn ne donne qu'un théorème d'existence, et ne marche pas pour toutes les propriétés. Le problème est cependant résolu pour certaines familles de codes (pour les codes préfixes et les codes à délai borné [14]). Pour d'autres familles, le problème est résolu, au moins dans le cas des codes minces, par la construction de Ehrenfeucht et Rozenberg d'un code complet contenant un code donné. Ceci n'est pas nouveau, et a déjà été remarqué, pour les codes rationnels par exemple, les codes circulaires [8], précirculaires [21], les ω-codes (communication privée de L. Staiger) . Nous avons ici rapporté les résultats qui s'obtiennent de cette façon.

Parmi les familles de codes étudiées, si on se limite aux codes rationnels, il n'y a que deux familles (celle des codes bipréfixes et celle des codes à délai de synchronisation borné) pour lesquelles il n'y a pas actuellement de réponse satisfaisante. Pour cette dernière famille, par analogie avec celle des codes à délai borné, on peut espérer qu'on pourra (même si le problème paraît difficile pour l'instant) construire un code, maximal parmi les codes de la famille, contenant un code de la famille (même non rationnel) donné.

Pour simplifier les notations, nous donnons la définition suivante:

Définition 9.0.1 *Soit P une propriété qui peut être vérifiée par certains langages de mots finis. Nous dirons qu'un code C est un* **P-code** *si C vérifie la propriété P.*

9.1 Composition des codes

Une méthode intéressante de construction de codes est la composition. En fait, la composition permet d'obtenir des codes de plus en plus compliqués. Le travail inverse de décomposition des codes est aussi important, mais nous ne l'étudierons pas ici.

Définition 9.1.1 *Soit C un langage inclus dans A^+ et $\phi : A^* \to B^*$ un morphisme de codage pour un langage D inclus dans B^+. Le langage $\phi(C)$ est appelé composé de C par ϕ et noté $C \circ_\phi D$.*

Notons A' l'alphabet de C. La condition d'existence d'un langage de la forme $C \circ_\phi D$ est qu'il existe une injection ϕ de A' dans D. Seule la partie de D constituée des mots de $\phi(A')$ est utile dans la construction de $\phi(C)$. On pourra donc supposer (si besoin) que le morphisme de codage de D est défini sur l'alphabet de C.

Proposition 9.1.1 *Soit C un langage inclus dans A^+ et $\phi : A^* \to B^*$ un morphisme de codage pour un langage D. Pour les propriétés P suivantes:*

- *être un code,*
- *être un code préfixe,*
- *être un code suffixe,*
- *être un code bipréfixe,*
- *être un biω-code,*
- *être un ω-code,*
- *être un code précirculaire,*
- *être un code pur,*
- *être un code circulaire,*
- *être un Z-code,*
- *être un code faiblement préfixe,*
- *être un code fortement précirculaire,*
- *être un code à délai de déchiffrage borné,*
- *être un code à délai de synchronisation borné,*

si C et D sont des P-codes, $\phi(C)$ est un P-code.

Preuve: Considérons un morphisme de codage $\psi : X^* \to A^*$ pour C. L'application composée $\phi \circ \psi$ est un morphisme de codage pour $\phi(C)$. La propriété P

se traduit dans chaque cas par une propriété Q des morphismes de codage. Ces propriétés Q ont été établies lors des chapitres précédents, ou sont immédiates.

Le résultat découle alors de la stabilité des propriétés Q par composition. Cette stabilité est assez claire dans la plupart des cas (par exemple, elle est immédiate quand Q est une propriété d'injectivité). Nous ne la démontrerons ici que dans les cas plus délicats des codes à délai de déchiffrage borné et des codes à délai de synchronisation borné. Pour ceux-ci, si c est un délai pour C et d un délai pour D, $(c+d)$ est un délai pour $\phi(C)$. •

Lemme 9.1.2 *Soit $\phi : A^* \to B^*$ un morphisme de codage pour un langage D. D est un code à délai borné d si et seulement si*

$$\forall h \geq 0 \ \forall a, a_1, ..., a_d, a', a'_1, ..., a'_h \in A$$

$$\phi(aa_1...a_d) \leq \phi(a'a'_1...a'_h) \quad \Rightarrow \quad a = a' \ .$$

Composition des codes à délai de déchiffrage borné: Considérons un code C à délai de déchiffrage c, $\psi : X^* \to A^*$ un morphisme de codage pour C, $\phi : A^* \to B^*$ un morphisme de codage pour un code D à délai de déchiffrage d. Montrons que $\phi(C)$ a pour délai $c+d$.

Soient $x, x_1, ..., x_{d+c}, x', x'_1, ..., x'_h \in X$ tels que:

$$\phi\psi(xx_1...x_{d+c}) \leq \phi\psi(x'x'_1...x'_h) \ .$$

Posons: $\psi(xx_1...x_{d+c}) = a_1...a_t$ et
$\psi(x'x'_1...x'_h) = a'_1...a'_s$, où les a_j, a'_j appartiennent à A.
Comme D a pour délai d, on a $a_j = a'_j$ pour $j \in \{1, 2, ..., t-d+1\}$. Donc $\psi(xx_1...x_c) \leq \psi(x'x'_1...x'_h)$, et C ayant pour délai c, $x = x'$. Donc $\phi(C)$ a pour délai $c+d$. •

Lemme 9.1.3 *Soit $\phi : A^* \to B^*$ un morphisme de codage pour un code D. Les conditions suivantes sont équivalentes:*
- D est un code à délai de synchronisation borné d,
- $\forall p > 1 \ \forall a, a_1, ..., a_d, a', a'_1, ..., a'_d, b_1, ..., b_p \in A \ \forall u, u' \in B^*$

$$u\phi(a_1...a_d a'_1...a'_d)u' = \phi(b_1...b_p) \ \Rightarrow$$

$$\exists i \ u\phi(a_1...a_d) = \phi(b_1...b_i), \ \phi(a'_1...a'_d)u' = \phi(b_{i+1}...b_p) \ ,$$

- $\forall n \geq 0 \ \forall p > 1 \ \forall a, a_1, ..., a_d, a', a'_1, ..., a'_d, a''_1, ...a''_n, b_1, ..., b_p \in A \ \forall u, u' \in B^*$

$$u\phi(a_1...a_d a''_1...a''_n a'_1...a'_d)u' = \phi(b_1...b_p) \ \Rightarrow \ \exists i \ u\phi(a_1...a_d) = \phi(b_1...b_i) \ ,$$

$$\phi(a'_1...a'_d)u' = \phi(b_{i+n+1}...b_p) \ \textit{et} \ \forall h \in \{1, 2, ..., n\} \ a''_h = b_{i+h} \ .$$

Composition des codes à délai de synchronisation borné: Considérons un code C à délai de synchronisation c, $\psi : X^* \to A^*$ un morphisme de codage pour C, $\phi : A^* \to B^*$ un morphisme de codage pour un code D à délai de synchronisation d. Montrons que $\phi(C)$ a pour délai $c+d$.

Soient $u, u' \in B^*$, $q > 0$, $x_1, ..., x_{d+c}, x'_1, ..., x'_{d+c}, y_1, ..., y_q \in X$ tels que:

$$u.\phi\psi(x_1...x_{d+c}x'_1...x'_{d+c}).u' = \phi\psi(y_1...y_q) ,$$

et posons:
$\psi(x_1...x_d) = a_1...a_d a''_1...a''_t$,
$\psi(x_{d+1}...x_{d+c}) = a''_{t+1}...a''_s$,
$\psi(x'_1...x'_c) = a''_{s+1}...a''_r$,
$\psi(x'_{c+1}...x'_{c+d}) = a''_{r+1}...a''_n a'_1...a'_d$,
$\psi(y_1...y_q) = b_1...b_p$, où les a_j, a'_j, a''_j, b_j appartiennent à A.

Le code D ayant pour délai d, on a:

$$\exists i \; u\phi(a_1...a_d) = \phi(b_1...b_i), \; \phi(a'_1...a'_d)u' = \phi(b_{i+n+1}...b_p)$$

et pour tout h de $\{1, 2, ..., n\}$ $a''_h = b_{i+h}$.
Donc $\psi(y_1...y_q) = b_1...b_i a''_1...a''_n b_{i+n+1}...b_p$. Comme $\psi(x_{d+1}...x_{d+c}x'_1...x'_c) = a''_{t+1}...a''_r$, c étant un délai pour C, on a $b_1...b_i a''_1...a''_t \psi(x_{d+1}...x_{d+c}) \in C^*$ et $\psi(x'_1...x'_c)a''_{r+1}...a''_n b_{i+n+1}...b_p \in C^*$. Par conséquent, $b_1...b_i a''_1...a''_s \in C^*$ et $a''_{s+1}...a''_n b_{i+n+1}...b_p \in C^*$. Or $u.\phi\psi(x_1...x_{d+c}) = u.\phi(a_1...a_d)\phi(a''_1...a''_s) = \phi(b_1...b_i)\phi(a''_1...a''_s)$, et donc $u.\phi\psi(x_1...x_{d+c})$ appartient à $\phi(C)^*$. De même, on démontre que $\phi\psi(x'_1...x'_{d+c}).u'$ appartient à $\phi(C)^*$, et donc $\phi(C)$ a pour délai $d+c$. •

Rappelons aussi que si C est un code rationnel (resp. fini) et D un code, $\phi(C)$ est un code rationnel (resp. fini).

Curieusement, le code composé de deux π-codes composables peut ne pas être un π-code:

Exemple: Considérons le π-code $C = \{c, ca, aba, baa\}$ et le morphisme ϕ défini par: $\phi(a) = ac$, $\phi(b) = b$, $\phi(c) = c$. Le code $D = \{ac, b, c\}$ est préfixe, donc c'est un π-code. Mais $\phi(C) = \{c, cac, acbac, bacac\}$ n'est pas un π-code puisque le mot $(cacba)^\omega$ a deux C-factorisations. Remarquons que C n'est pas faiblement préfixe puisque le mot $c(aba)^\omega$ a deux C-factorisations.

Cependant, on a:

Proposition 9.1.4 *Soient $C \subset A^+$ un code faiblement préfixe et $\phi : A^* \to B^*$ un morphisme de codage pour un π-code $D \subset A^+$. Le langage $\phi(C)$ est un π-code.*

Preuve : Considérons un morphisme de codage $\psi : X^* \to A^*$ pour C, et deux mots z et z' de X^ω tels que $\phi\psi(z)$ égale $\phi\psi(z')$ et soit périodique. Puisque D est un π-code, $\psi(z) = \psi(z')$. Puisque D est un code, d'après la proposition 1.2.5, $\psi(z)$ est ultimement périodique. Le code C étant faiblement préfixe, on a alors $z = z'$. •

Proposition 9.1.5 *Soit C un langage inclus dans A^+ et $\phi : A^* \to B^*$ un morphisme de codage pour un langage D.*

Pour les propriétés P suivantes :

- *être un code,*
- *être un code préfixe,*
- *être un code suffixe,*
- *être un code bipréfixe,*
- *être un ω-code,*
- *être un code faiblement préfixe,*
- *être un π-code,*
- *être un code fortement précirculaire,*
- *être un code précirculaire,*
- *être un code pur,*
- *être un code circulaire,*
- *être un Z-code,*
- *être un $bi\omega$-code,*
- *être un code de délai de déchiffrage d,*
- *être un code de délai de synchronisation d,*

si $\phi(C)$ un P-code, C est un P-code.

Preuve : Considérons un morphisme de codage $\psi : X^* \to A^*$ pour C. Pour chacune des propriétés considérées, le résultat s'énonce aussi : si $\phi \circ \psi$ vérifie certaine propriété Q alors ψ vérifie Q. Et dans chaque cas, ce fait se vérifie (presque) facilement. •

Pour toutes les propriétés considérées, on peut faire la remarque suivante : même si D n'est pas un code, (ou un P-code), $\phi(C)$ peut être un P-code.

Exemple : Le langage $D = \{a, ba, ab\}$ n'est pas un code. Considérons $C = \{aaba, c\}$ et posons $\phi(a) = a$, $\phi(b) = ba$ et $\phi(c) = ab$. Le langage $\phi(C) = \{aabaa, ab\}$ est un code bipréfixe, c'est aussi un Z-code fini donc un code à délai de synchronisation borné.

9.2 Maximalité

9.2.1 Plongement dans un code P-maximal

Définitions 9.2.1 *Nous dirons qu'un P-code C inclus dans A^+ est un code* **P-maximal** *si C est maximal (pour la relation d'inclusion) dans l'ensemble des P-codes inclus dans A^+. Un code P-maximal est dit* **maximal** *quand P est la condition vide.*

De façon claire, tout P-code maximal est P-maximal. De même, si la propriété P' implique la propriété P, tout P'-code P-maximal est P'-maximal. Les réciproques semblent *a priori* fausses, mais la question reste ouverte dans la plupart des cas. Par exemple, on ne connaît pas de code circulaire-maximal (ou précirculaire-maximal) non maximal [54]. Voici cependant un exemple montrant que la réciproque est fausse pour certaines propriétés P et P'.

Exemple [8]: Posons $A = \{a, b\}, C = \{wba^{|w|} \mid w \in A^*\}$, $C_0 = C - CA^+$. Le code bipréfixe C_0 est préfixe-maximal, donc aussi bipréfixe-maximal, mais est inclus strictement dans le code suffixe C, donc C_0 n'est ni maximal, ni même suffixe-maximal.

La proposition suivante regroupe des faits résultant de l'utilisation du lemme de Zorn.

Proposition 9.2.1 *L'ensemble $P(A)$ des P-codes inclus dans A^+ a des maximaux pour l'inclusion et tout élément de $P(A)$ est inclus dans un P-code P-maximal dès que P est l'une des propriétés suivantes:*

- *être un code,*
- *être un code préfixe,*
- *être un code bipréfixe,*
- *être un code suffixe,*
- *être un code circulaire,*
- *être un code faiblement préfixe,*
- *être un π-code,*
- *être un code fortement précirculaire,*
- *être un code précirculaire,*
- *être un code pur,*
- *être un code de délai de déchiffrage d,*
- *être un code de délai de synchronisation d.*

Preuve: La preuve est basée sur le fait suivant: pour chaque propriété P considérée, on peut exprimer qu'un langage C est un P-code par une formule ne faisant intervenir qu'un nombre fini de mots de C.

Par exemple, les mots de C intervenant dans la formule qui exprime que C est un code: $\forall n > 0\ \forall p > 0\ \forall u_1, u_2, ..., u_n, v_1, v_2, ..., v_p \in C$

$$u_1 u_2 ... u_n = v_1 v_2 ... v_p \ \Rightarrow\ u_1 = v_1 \ ,$$

sont les mots $u_1, u_2, ..., u_n, v_1, v_2, ..., v_p$.

Plus précisément, pour un langage C, le fait d'être un P-code s'exprime par une conjonction de formules du type suivant:
$\forall n_1, n_2, ..., n_p > 0\ \ \forall u_1, v_2, ..., u_{n_1} \in A^*\ \ \forall v_1, v_2, ..., v_{n_2} \in C$

$$Q(n_1, ...n_p, u_1, ..., u_{n_1}, v_1, ..., v_{n_2}) \ \Rightarrow\ Q'(n_1, ...n_p, u_1, ..., u_{n_1}, v_1, ..., v_{n_2}) \ ,$$

où Q est une conjonction d'égalités et Q' une propriété pouvant faire intervenir des quantificateurs, des égalités et l'appartenance à C ou à C^*.

Cette façon d'exprimer la propriété P est claire quand P est l'une des propriétés suivantes:
- être un code,
- être un code préfixe,
- être un code bipréfixe,
- être un code suffixe,
- être un code précirculaire,
- être un code circulaire.

C'est moins clair quand la définition des P-codes utilise des mots de C^*, mais c'est encore vrai puisqu'un mot de C^* se factorise en un nombre fini de mots de C. Par exemple, si l'on exprime que C est un code pur: C est un code (expression déjà vue) qui vérifie:

$$\forall u \in A^+\ \forall n > 0\ \forall p > 0\ \forall u_1, u_2, ..., u_p \in C\ (u^n = u_1 u_2 ... u_p \ \Rightarrow\ u \in C^+).$$

Les mots de C intervenant dans cette dernière formule sont les mots $u_1, u_2, ..., u_p$.

Cela donne le résultat quand P est l'une des propriétés suivantes:
- être un code pur,
- être un code faiblement préfixe,
- être un π-code,
- être un code fortement précirculaire,
- être un code de délai de déchiffrage d,
- être un code de délai de synchronisation d.

Venons-en à la démonstration proprement dite. Considérons une chaîne $(E_i)_{i \in I}$ croissante d'éléments de $P(A)$. Nous allons montrer que la réunion E des éléments de la chaîne est un P-code. Pour chacune des formules constituant la définition des P-codes, considérons les mots de E intervenant dans cette formule: $v_1, v_2, ..., v_d$. Pour tout $j \in \{1, 2, ..., d\}$, il existe i_j tel que v_j

appartienne à E_{i_j}, et donc $\{v_1, v_2, ..., v_d\} \subset E_{sup\{i_j | j=1,2,...,d\}}$. Si $(v_1, v_2, ..., v_d)$ vérifie la propriété Q, comme $E_{sup\{i_j | j=1,2,...,d\}}$ est un P-code, $(v_1, v_2, ..., v_d)$ vérifie la propriété Q'. On en déduit que E est un P-code.

On conclut en utilisant le lemme de Zorn. •

Les propriétés des codes qui n'ont pu être étudiées de cette façon sont les suivantes :

- être un Z-code,
- être un biω-code,
- être un ω-code,
- être un code à délai de déchiffrage borné,
- être un code à délai de synchronisation borné,
- être un code fini,
- être un code rationnel,
- être un P'-code rationnel (pour une propriété P' de la liste des propriétés étudiées).

Dans chacun de ces cas, la réunion E des éléments d'une chaîne $(E_i)_{i \in I}$ croissante d'éléments de $P(A)$ peut ne pas être un P-code. Les exemples suivants permettent de s'en convaincre.

Exemples :

- Posons $C_p = \{ab\} \cup \{ab^n ab^{n+1} \mid 0 < n < p\}$ pour tout $p > 0$. La suite $(C_p)_{p>0}$ forme une chaîne croissante. Pour tout p, C_p est un code circulaire fini, donc un code à délai de synchronisation borné. Mais le code $C = \bigcup C_p$ n'est ni rationnel, ni un ω-code : le mot $abab^2ab^3ab^4...$ a deux C-factorisations.
- Posons $C_p = \{a^n b^n \mid 0 < n \leq p\}$ pour tout $p > 0$. La suite $(C_p)_{p>0}$ forme une chaîne croissante. Pour tout p, C_p est un code bipréfixe fini, mais le code $C = \bigcup C_p$ n'est pas rationnel.

La proposition 9.2.1 de ce paragraphe n'est qu'un théorème d'existence. De plus, elle n'est pas valable pour toutes les propriétés étudiées. Il serait plus intéressant de savoir **déterminer** un P-code P-maximal contenant un P-code donné. Ce problème n'est en général pas résolu. Cela est fait cependant dans certains cas :

Proposition 9.2.2 *[8], [14], [16]. Pour tout P-code, où P signifie : préfixe, resp. suffixe, resp. à délai de déchiffrage d, on sait déterminer explicitement un P-code P-maximal qui le contient.*

9.2.2 Plongement dans un code complet

Définition 9.2.2 *Un code C inclus dans A^+ est dit* **complet** *si $A^* = F(C^*)$.*

Savoir déterminer un P-code P-maximal contenant un P-code donné n'est en général pas résolu. La recherche d'un code complet contenant un P-code donné permet d'avancer dans ce problème. Cependant, si tout code maximal est complet (voir la proposition qui suit) [58] , la réciproque est fausse, et même un P-code complet peut ne pas être P-maximal. Il suffit de considérer, pour des propriétés P et P' telles que P' implique P, un P'-code P'-maximal, non P-maximal. Nous en avons vu un exemple au début de la sous-section précédente. Voici un autre exemple:

Exemple: Considérons le code D'_1 appelé semi-Dyck sur l'alphabet $A = \{a, b\}$. Le code $C = D'_1 \cup \{a\}$ est circulaire; le code $C = D'_1 \cup \{a^2\}$ est précirculaire; donc D'_1, qui est complet, n'est ni maximal, ni précirculaire-maximal, ni circulaire-maximal.

Les résultats qui peuvent se déduire de la construction de Ehrenfeucht et Rozenberg d'un code complet contenant un code donné sont regroupés dans l'énoncé suivant. Remarquons que De Luca et Restivo ont démontré qu'un code circulaire-maximal est complet [46] et que Capocelli a prouvé qu'un code faiblement-préfixe-maximal est complet [17].

Proposition 9.2.3 *Tout code P-maximal est complet et pour tout P-code on sait déterminer explicitement un P-code complet qui le contient, dès que P est une propriété de la liste suivante:*

- *être un code [30],*
- *être un code rationnel [30],*
- *être un ω-code,*
- *être un code faiblement préfixe,*
- *être un π-code,*
- *être un Z-code,*
- *être un code circulaire,*
- *être un biω-code,*
- *être un code fortement précirculaire,*
- *être un code précirculaire,*
- *être un code pur,*
- *être un P'-code rationnel (pour une propriété P' de la liste des propriétés précédentes),*

Preuve: Dans le cas d'un alphabet à une lettre, les résultats sont immédiats: si $A = \{a\}$, tout code est de la forme $\{a^p\}$ et est maximal et complet.

Dans le cas où A a au moins deux lettres, la preuve découle de celle de A. Ehrenfeucht et G. Rozenberg, pour toutes les propriétés P considérées.

En effet, si A a au moins deux lettres et qu'un P-code C n'est pas complet, il existe un mot $w \notin F(C^*)$. On peut choisir w sans bord, c'est-à-dire tel que: $Fg(w) \cap Fd(w) = \{w, \varepsilon\}$.

Posons $U = A^* - C^* - A^*wA^*$ et $C' = C \cup w(Uw)^*$ et montrons que le langage $C' = C \cup w(Uw)^*$ est un P-code complet. On sait déjà que c'est un code complet [30]. Il reste à démontrer que C' est un P-code.

Pour la propriété « être rationnel », c'est immédiat.

Pour toutes les autres propriétés sauf « être pur », on s'intéresse à l'unicité de la C'-factorisation de certains mots (par exemple des mots infinis périodiques quand on étudie les π-codes).

Remarquons que l'on a:
$\forall u, u' \in A^* \ \forall \alpha \in w(Uw)^* \ \forall v_1, v_2, ..., v_n \in C'$

$$u\alpha u' = v_1 v_2 ... v_n \ \Rightarrow \ \exists i \ \alpha \in F(v_i), \ v_1 ... v_{i-1} \in Fg(u), \ v_{i+1} ... v_n \in Fd(u') \ .$$

Donc dans toutes les C'-factorisations d'un mot donné (fini, infini ou bi-infini) et dans toutes les C'-décompositions d'un Z-mot donné, les mots de $w(Uw)^*$ sont fixés. Il ne reste plus alors qu'à factoriser sur C certains mots finis, infinis ou bi-infinis, mots pour lesquels l'unicité de la C-factorisation est vérifiée car C est un P-code.

Il nous reste la propriété « être pur ». Considérons un élément u^n de C'^+ et supposons $n > 1$; si $u^n \in C^+$, $u \in C$ car le code C est pur. Si $u^n \notin C^+$, les mots de $w(Uw)^*$ qui interviennent dans sa C'-factorisation sont fixés, et se retrouvent dans cette C'-factorisation à intervalles réguliers (après translations de longueur $|u|$), et ne peuvent donc être à cheval sur plusieurs u consécutifs. On peut donc décomposer u^n sous la forme: $u^n = u_1 \alpha u_2 \alpha' u_3$ où $u_1 \in C^*$, $u_2 \in C'^*$, α et $\alpha' \in w(Uw)^*$, et $u_3 \in C^*$. Les positions de α et α' sont fixées, donc u s'écrit $u_1 \alpha u'_2 \alpha' u_3$ ou $u_1 \alpha u_3$, où $u'_2 \in C'^*$. Donc u est dans C'^+. •

Pour les autres propriétés, la construction précédente ne marche pas. C'est évident pour les codes préfixes, suffixes ou bipréfixes, et V. Bruyère l'a démontré pour les codes à délai de déchiffrage borné [14]. Cependant, l'énoncé de la proposition 9.2.3 reste vrai dans le cas des codes préfixes, suffixes et des codes à délai de déchiffrage d. cela résulte des démonstrations de la proposition 9.2.2.

Pour les codes bipréfixes, seul un résultat partiel a été démontré: tout code fini bipréfixe est inclus dans un code rationnel, maximal et bipréfixe [51]. Mais le cas général n'est pas résolu [15].

Dans le cas des propriétés P suivantes:
- être un code à délai de synchronisation borné,
- être un code de délai de synchronisation d,

les questions étudiées à la proposition précédente restent ouvertes. On peut remarquer que la construction d'Ehrenfeucht et Rozenberg ne marche pas en général. Ce fait est une conséquence du travail de V. Bruyère sur les codes à délai de déchiffrage borné. Si le code C' construit par Ehrenfeucht et Rozenberg était à délai de synchronisation borné, il vérifierait la propriété:

$\exists p > 0\ F(C') \cap C'^p = \emptyset$. Pour démontrer le contraire, il suffit donc d'exhiber pour tout p un élément u de U tel que $F(u) \cap C'^p \neq \emptyset$. Ceci est fait dans [14] pour la plupart des codes finis.

La propriété: « être un code fini » n'a pas été étudiée ici. Clairement, un code fini-maximal est un code maximal, donc un code complet. Mais le second résultat de la proposition précédente est faux pour la propriété « être un code fini ». L'exemple suivant est de A. Restivo [54].

Exemple: Posons $C = \{a^5, b, ab, ba^2\}$. Tout code maximal contenant le code C est infini, tout code complet contenant C est infini. On ne sait pas s'il existe des codes à trois éléments vérifiant la même propriété.

Rappelons que la décidabilité de l'existence d'un code fini et maximal contenant un code fini donné reste une question ouverte [15].

9.2.3 Plongement dans un code maximal

Définition 9.2.3 *Un code C inclus dans A^+ est dit* **mince** *si $F(C) \neq A^*$.*

Le plongement d'un P-code dans un P-code maximal est résolu par la proposition 9.2.3 dans le cas des codes rationnels et des codes minces. En effet, nous avons:

Proposition 9.2.4 *Tout code rationnel est mince.*
Pour un code mince, les propriétés « être un code maximal » et « être complet » sont équivalentes [58].

On déduit de la proposition 9.2.3 les deux résultats suivants.

Proposition 9.2.5 *Pour un P-code mince C, quand P est l'une des propriétés étudiées à la proposition 9.2.3, les propriétés suivantes sont équivalentes:*
- C est un code maximal,
- C est P-maximal,
- C est complet.

Théorème 9.2.6 *Pour les propriétés P étudiées à la proposition 9.2.3, tout P-code mince (resp. rationnel) est (effectivement) inclus dans un P-code mince (resp. rationnel), maximal en tant que code.*

Quand les codes considérés ne sont pas minces, un P-code P-maximal (qui est donc complet) peut *a priori* ne pas être maximal. Nous en avons vu un exemple dans le cas des codes préfixes et bipréfixes. Mais la question reste ouverte pour d'autres propriétés (dans le cas des codes circulaires et précirculaires, par exemple).

Chapitre 10

Générateurs de langages bi-infinitaires

L'objet de ce chapitre est d'ébaucher une étude de la famille des ensembles générateurs d'un langage de mots bi-infinis donné.

Il s'agit d'abord de savoir ce que l'on entend par générateur d'un langage de mots bi-infinis L. Plusieurs façons de voir les choses coexistent.
1) un générateur de L peut être un langage G de mots finis, infinis ou bi-infinis tel que $L = {}^{\infty}G^{\infty} \cap {}^{\omega}A^{\omega}$, où ${}^{\infty}G^{\infty}$ désigne l'ensemble des mots obtenus par concaténation (finie ou non) de mots de G;
2) un générateur de L peut être un langage G de mots finis, infinis ou bi-infinis tel que $L = G^* \cap {}^{\omega}A^{\omega}$, où G^* désigne l'ensemble des mots obtenus par concaténation finie de mots de G;
3) un générateur de L peut être un langage G de mots finis tel que L soit la fermeture de ${}^{\omega}G^{\omega}$;
4) un générateur de L peut être un langage G de mots finis tel que $L = {}^{\omega}G^{\omega}$.

Les deux premières notions ne semblent pas avoir été étudiées. La première est cependant liée à la notion de « code bi-infinitaire strict » et la seconde à celle de « code bi-infinitaire », notions définies et étudiées par Do Long Van et coll. [26], [25], [28]. Un « code bi-infinitaire strict » (resp. « code bi-infinitaire ») est en effet un ensemble C de mots finis, infinis ou bi-infinis tel que tout mot de ${}^{\infty}A^{\infty}$ a au plus une C-factorisation (resp. C-factorisation finie).

Les notions de générateurs qui s'apparentent aux deux premières notions, et qui ont été étudiées sont celles de générateurs, au sens des monoïdes, de langages L inclus dans ${}^{\infty}A^{\infty}$ ou A^{∞}, c'est-à-dire de langages G tels que $L = G^*$. On peut citer à ce sujet les travaux de I. Litovsky [43] et Do Long Van et coll. [28].

La troisième notion relève de la dynamique symbolique, et ne s'intéresse qu'à la génération de langages fermés.

L'objet de la dynamique symbolique est l'étude de systèmes abstraits, appelés systèmes dynamiques symboliques, qui sont en fait les sous-ensembles S de A^Z fermés et stables par décalage, c'est-à-dire tels que $\sigma(S) = S$. Ce sont les langages que nous avons appelé jusqu'à présent biadhérences de A^Z. Ils sont caractérisés par l'ensemble de leurs facteurs finis: $S = Biadh(F(S))$. Les systèmes sofiques sont les systèmes dynamiques symboliques reconnus par un automate fini. Ce sont les biadhérences rationnelles de A^Z. Les systèmes de type fini sont des systèmes sofiques S particuliers: $F(S)$ est de la forme $A^* - A^*HA^*$ où H est un ensemble fini.

Etudier les générateurs du troisième type d'un langage de mots bi-infinis L fermé revient à étudier les langages G de mots finis tels que $F(G^*) = F(L)$, c'est-à-dire tels que $F(G^*) = F(S)$ où S est le système dynamique symbolique $\pi^{-1}(L)$. Si L est rationnel, S est un système sofique et $F(S)$ est rationnel.

Un langage U de la forme $F(G^*)$ est **factoriel** ($u \in F(U) \Rightarrow u \in U$) et **transitif** ($u, v \in U \Rightarrow \exists x \in A^* \; uxv \in U$). Si U est un langage factoriel et transitif, la biadhérence dans A^Z de U est un système dynamique symbolique S tel que $F(S) = F(U)$. Si U est rationnel, S est un système sofique.

Etudier les générateurs du troisième type d'une biadhérence rationnelle revient donc à étudier les langages $G \subset A^*$ vérifiant $F(G^*) = U$, où $U \subset A^*$ est rationnel, factoriel et transitif. F. Blanchard et G. Hansel [10] ont démontré l'existence, pour tout U rationnel, factoriel et transitif, d'un code préfixe G vérifiant $F(G^*) = U$. Ce résultat a été renforcé par M. P. Béal et D. Perrin [3] qui ont prouvé qu'on pouvait trouver un code préfixe et circulaire vérifiant la même propriété. D. Beauquier [7] a démontré l'existence, pour tout U rationnel, factoriel et transitif, d'un automate (Q, I, T, R) déterministe, fortement connexe tel que $I = T = Q$ reconnaissant U et minimal en un certain sens. Cet automate se révèle un outil très intéressant pour étudier le problème posé. Finalement, A. Restivo [55], [57], a montré que l'existence, pour U rationnel, factoriel et transitif donné, d'un langage fini G vérifiant $F(G^*) = U$ est décidable.

Les troisième et quatrième notions de générateur ne coïncident pas:

Exemple : Soit $G = a^*b$, ${}^\omega G^\omega = {}^\omega(a^*b)^\omega$, $F({}^\omega G^\omega) = A^*$ et la fermeture de ${}^\omega G^\omega$ est ${}^\omega A^\omega$. G est un générateur du troisième type, mais pas du quatrième type pour ${}^\omega A^\omega$.

Cependant, pour un langage L fermé, un générateur G du quatrième type est du troisième type: ${}^\omega G^\omega = L = \overline{L} = \overline{{}^\omega G^\omega}$ ($\overline{U}$ désigne la fermeture d'un langage $U \subset {}^\omega A^\omega$). D'autre part, un générateur G du troisième type, tel que ${}^\omega G^\omega$ est fermé, est du quatrième type. C'est vrai en particulier pour les générateurs finis.

Pour notre part, nous nous intéresserons à la quatrième notion de générateur.

Dans le cadre des mots infinis, cette notion de générateur a été essentiellement étudiée par I. Litovsky [42], [41]. Nous avons examiné dans le cas bi-infinitaire les résultats valables dans le cas infinitaire. Le bilan n'est pas très encourageant. Certaines méthodes s'adaptent, mais la plupart ne s'adaptent pas. Des résultats simples ne se conservent pas. Comme pour l'étude des codes pour mots infinis, il semble que les résultats puissent être très différents, et que d'autres techniques de démonstration doivent être envisagées.

Dans le cas d'un langage infinitaire L, le langage $\{u \in A^* \mid uL \subset L\}$, appelé stabilisateur de L, joue un grand rôle dans l'étude des générateurs de L. Dans le cas bi-infinitaire, nous n'avons pas trouvé d'ensembles qui jouent le même rôle, (sauf peut-être les stabilisateurs des états de certains automates reconnaissant L, dans le cas où L est une biadhérence). C'est ce qui explique le peu de résultats.

Dans un premier temps, nous rappelons les résultats concernant l'existence de générateurs et celle de générateurs maximaux pour les langages bi-infinitaires rationnels. Puis nous nous intéressons aux générateurs minimaux. Nous ne savons pas grand'chose de leur existence. Mais clairement, un langage finiment engendré a au moins un générateur (fini) minimal.

Nos résultats sur les générateurs minimaux sont relatifs aux codes. L'unicité de la factorisation des mots bi-infinis périodiques sur un code précirculaire montre que ceux-ci sont des générateurs minimaux. Nous montrons aussi qu'un code C très mince, est un générateur minimal de ${}^\omega C^\omega$. Nous ne savons pas si cela reste vrai pour un code quelconque.

Nous avons essayé d'approfondir la liaison existant entre le cas infinitaire et le cas bi-infinitaire, en définissant une notion de « bissection générative » d'un langage bi-infinitaire. Une bissection générative de L est un couple de la forme $({}^\omega G, G^\omega)$ où G vérifie : ${}^\omega G^\omega = L$. Cette notion aurait pu être intéressante si le nombre de bissections génératives rationnelles d'un langage bi-infinitaire rationnel donné était fini. Hélas, ceci est faux, même dans le cas de langages bi-infinitaires finiment engendrés.

Cela fait le deuil de l'idée de simplifier l'étude des générateurs d'un langage bi-infinitaire rationnel à l'aide de celle des générateurs des langages infinitaires rationnels.

Cette idée nous semblait pourtant assez adaptée à la recherche de générateurs finis : on sait en effet décider si des langages G^ω ou ${}^\omega G$ sont finiment engendrés [44] [35], et donc décider si une bissection générative rationnelle est finiment engendrée. Mais cela est insuffisant pour pouvoir conclure.

Pour étudier l'existence de générateurs finis, nous sommes donc amenés à étudier directement le langage bi-infinitaire donné. Nous montrons finalement que l'existence de générateurs finis est décidable quand le langage est rationnel. Ce résultat est équivalent à celui de A. Restivo [55].

10.1 Générateurs d'un langage bi-infinitaire

La question que l'on étudie est la suivante : étant donné un langage rationnel de mots bi-infinis L, existe-t-il des générateurs de L du quatrième type, et si oui, que dire de la famille de ces générateurs?

Définition 10.1.1 *Soit L un langage de mots bi-infinis. Un* **générateur** *de L est un langage G de mots finis tel que $L = {}^{\omega}G^{\omega}$.*

Notations : Nous notons $[L]$ l'ensemble des générateurs d'un langage de mots bi-infinis L donné :

$$[L] = \{G \subset A^* \mid {}^{\omega}G^{\omega} = L\} ,$$

ainsi que, pour $L' \subset {}^{\omega}A$ et $L'' \subset A^{\omega}$:

$$_{\omega}[L'] = \{G \subset A^* \mid {}^{\omega}G = L'\} ,$$

$$[L'']_{\omega} = \{G \subset A^* \mid G^{\omega} = L''\} .$$

Nous notons $\overline{L}$ la fermeture d'un langage $L \subset {}^{\omega}A^{\omega}$).

Comme dans le cas infinitaire, on peut tout de suite remarquer les faits suivants :
- si $G \in [L]$ alors pour tout $i > 0$, $G^i \in [L]$ et $G^* \in [L]$, donc s'il existe des éléments maximaux dans $[L]$, ce seront des monoïdes,
- si $G, G' \in [L]$ et si $G \subset R \subset G'$, alors $R \in [L]$,
- si $G \in [L]$ alors $Prem(G) = G - GG^+ \in [L]$,
- en général, $[L]$ n'est stable ni par union ni par produit : par exemple, $G = \{ab\}$ et $G' = \{ba\}$ engendrent $L = {}^{\omega}(ab)^{\omega}$ et ni $G \cup G'$, ni $G \cap G'$, ni GG' n'engendrent L.
- $[L]$ n'est jamais stable par intersection :

$$G \in [L] \Rightarrow Prem(G) \in [L], (Prem(G))^2 \in [L], Prem(G) \cap (Prem(G))^2 = \emptyset .$$

Nous nous intéressons plus particulièrement aux langages $L \subset {}^{\omega}A^{\omega}$ rationnels. Remarquons qu'un langage rationnel peut avoir des générateurs non rationnels.

Exemple : Le langage $G = \cup_{p \text{ premier}}(ab + ba)^p$ engendre le langage rationnel ${}^{\omega}(ab + ba)^{\omega}$, cependant, G n'est pas rationnel.

Etant donné un automate de Büchi (Q, I, T, R) (non nécessairement fini), on peut considérer la **congruence de Büchi associée**. Notons :

$$R_I(q,u) = \{q' \in Q \mid \exists i \in I \ \exists v,w \in A^* \ u = vw, \ i \in R(q,v), \ q' \in R(i,w)\}$$

$$R_T(q,u) = \{q' \in Q \mid \exists t \in T \ \exists v,w \in A^* \ u = vw, \ t \in R(q,v), \ q' \in R(t,w)\}$$

La congruence de Büchi associée à l'automate est définie pour u et $v \in A^*$ par : $u \cong v \Leftrightarrow$

$$\forall q \in Q \ R(q,u) = R(q,v) \ \ R_T(q,u) = R_T(q,v) \ \ R_I(q,u) = R_I(q,v) \ .$$

Cette congruence est d'index fini quand l'automate est fini, c'est-à-dire pour un langage L rationnel.

Rappelons que, pour une relation d'équivalence donnée, on appelle **saturé** d'un langage $G \subset A^*$ l'ensemble des mots de A^* congrus à un mot de G.

Lemme 10.1.1 *Considérons un automate de Büchi (non nécessairement fini) et le langage de mots bi-infinis reconnu L. Si $[L]$ n'est pas vide, pour tout G de $[L]$, le saturé de G pour la congruence de Büchi associée à l'automate : $Sat(G)$ appartient à $[L]$.*

Preuve : Considérons un mot bi-infini de ${}^{\omega}Sat(G)^{\omega}$: $w = ...w_{-1}w_0w_1w_2...$ où $w_i \in Sat(G)$ pour tout i, et des mots u_i de G congrus aux w_i : $u_i \cong w_i$ pour tout i. Le mot de L : $...u_{-1}u_0u_1u_2...$ a une lecture réussie sur l'automate, et donc le mot w aussi. Le mot w appartient à L. •

Proposition 10.1.2 *[42] Si $L \subset {}^{\omega}A^{\omega}$ est rationnel, on peut décider si $[L]$ est vide ou non. Si $[L]$ n'est pas vide, $[L]$ a un nombre fini, calculable d'éléments maximaux pour l'inclusion. Ces générateurs maximaux sont rationnels et constructibles, et tout générateur est inclus dans un générateur maximal.*

Preuve : Considérons un automate de Büchi fini (Q, I, T, R) reconnaissant L et la congruence de Büchi associée. D'après le lemme précédent, l'étude des langages saturés (en nombre fini) permet de décider si $[L]$ est vide. Si $[L]$ n'est pas vide, les générateurs maximaux existent, et sont à rechercher parmi les langages saturés. Il suffit donc de tester si les langages saturés sont des générateurs, puis de déterminer ceux qui sont maximaux. •

Examinons l'existence d'un plus grand générateur. Comme dans le cas infinitaire, on a:

Corollaire 10.1.3 *Soit $L \subset {}^{\omega}A^{\omega}$ un langage rationnel tel que $[L]$ n'est pas vide. Les conditions suivantes sont équivalentes:*
- L a un plus grand générateur,
- $[L]$ est stable par union,
- l'ensemble des éléments de $[L]$ qui contiennent ϵ est stable par produit.

Preuve : S'il existe au moins deux générateurs maximaux M_1 et M_2, on a: $M_1 \subset M_1 \cup M_2 \subset M_1M_2$ (car ce sont des monoïdes) et $M_1 \neq M_1 \cup M_2$. Donc $[L]$ ne peut être stable ni par union, ni par produit.

Si L a un plus grand générateur M, tout générateur est inclus dans M et donc toute union de générateurs contient un générateur et est incluse dans le générateur M, et est donc un générateur. Donc $[L]$ est stable par union.

Tout générateur est inclus dans M et M est un monoïde, donc tout produit fini de générateurs est inclus dans M. Si un générateur G contient ϵ, tout produit GG' ou $G'G$ contient G', et est donc générateur. •

Dans le cas infinitaire,
- si L a un plus grand générateur, $[L]$ est stable par produit;
- si L est une adhérence rationnelle, L a un seul générateur maximal.
Dans le cas bi-infinitaire cela change:

Proposition 10.1.4 *La famille des générateurs d'un langage ayant un plus grand générateur peut ne pas être stable par produit. Une biadhérence rationnelle, même finiment engendrée, peut avoir un ou plusieurs générateurs maximaux.*

Exemple : Le langage $L = {}^{\omega}(a^*b)^{\omega}$ a un plus grand générateur A^*bA^*. Les langages a^*b et ba^* sont des générateurs, mais ni leurs produits a^*bba^* et ba^*b, ni même la réunion de leurs produits, n'engendrent L.

Exemple : Le langage $L = {}^{\omega}(ab)^{\omega}$ a pour générateurs maximaux les langages $(ab)^*$ et $(ba)^*$.

Exemple : Le monoïde $(ab+ba)^*$ est le plus grand générateur de ${}^{\omega}(ab+ba)^{\omega}$.

Le premier des exemples précédents montre que, comme dans le cas infinitaire, un langage rationnel bi-infinitaire non fermé peut avoir un plus grand générateur.

10.2 Générateurs minimaux; Cas particulier des codes

Si $L \subset {}^{\omega}A^{\omega}$ a des générateurs, certains d'entre eux peuvent être plus intéressants que d'autres.

Bien entendu, ceux qui sont rationnels le sont, puisqu'il suffit d'un automate fini pour les mémoriser. L'existence de générateurs rationnels implique que L soit aussi rationnel.

Les générateurs minimaux sont aussi appréciables, dans la mesure où ils ne possèdent que l'information strictement nécessaire pour pouvoir engendrer L. Hélas, comme dans le cas des langages L infinitaires, nous savons peu de choses sur les générateurs minimaux. Nous ne savons pas s'il en existe, ni si leur existence est décidable, ni *a fortiori* si tout générateur contient un générateur minimal.

Dans le cas infinitaire, on sait décider si un générateur rationnel donné est minimal. Dans le cas bi-infinitaire, nous ne savons pas si on peut le décider. Dans les cas concrets étudiés, il suffit de trouver pour chaque mot u du générateur un mot bi-infini dont toutes les C-factorisations utilisent u. A titre d'exemple, nous démontrons la propriété suivante:

Proposition 10.2.1 *Tout code C à deux éléments est un générateur minimal de ${}^{\omega}C^{\omega}$.*

Preuve : Considérons u et v primitifs distincts et le code $C = \{u^n, v^p\}$. Si u et v ne sont pas conjugués, le code est précirculaire. Si u et v sont conjugués, seul le mot ${}^{\omega}u^{\omega}$ a deux C-factorisations, le mot ${}^{\omega}uv^pu^{\omega}$ n'en a qu'une seule et donc u^n et v^p ne sont pas superflus. •

Certains générateurs: les biω-codes et en particulier les Z-codes, codes circulaires, codes à délai de synchronisation borné, sont intéressants parce qu'ils permettent un décodage unique des mots de L. Plus ils sont particuliers et plus ils seront intéressants, les codes à délai de synchronisation borné permettant quant à eux un décodage concret. La proposition suivante montre que ces générateurs sont minimaux.

Proposition 10.2.2 *Tout code précirculaire C est un générateur minimal de ${}^{\omega}C^{\omega}$.*

Preuve : Soit C un code précirculaire. Pour tout u de C, le mot périodique ${}^{\omega}u^{\omega}$ a une C-factorisation unique: $(u)_{n\in Z}$. Donc ${}^{\omega}u^{\omega}$ n'appartient pas à ${}^{\omega}(C-u)^{\omega}$, et donc $(C-u)$ n'engendre pas ${}^{\omega}C^{\omega}$. •

Cette propriété correspond à la propriété suivante vérifiée dans le cas infinitaire: les codes sont des générateurs minimaux pour les ω-langages.

Nous ne savons pas si tout code est générateur minimal. Cela est dû au fait que nous n'avons pas de caractérisation des codes en termes d'unicité du décodage de certains mots bi-infinis. Cependant, nous avons quelques résultats intéressants:

Proposition 10.2.3 *Tout code C mince et maximal est un générateur minimal de ${}^{\omega}C^{\omega}$.*

Preuve : Soit C un code mince et maximal et posons $L = {}^{\omega}C^{\omega}$. C est complet donc $F(C^*) = A^*$. Pour tout $u \in C$, le code mince $B_u = C - \{u\}$ n'est pas maximal, donc n'est pas complet. Il existe un mot v de A^*, n'appartenant pas à $F(B_u{}^*)$. C étant complet, on peut prolonger v en un mot de L. Ce mot de L ne se factorise pas sur B_u, donc B_u n'engendre pas L. •

Remarquons qu'un code maximal, non mince, peut aussi être un générateur minimal. Prenons comme exemple le code de Dyck D_1 sur $A = \{a, b\}$, base du monoïde des mots ayant autant de a que de b. Ce code n'est pas mince, car tout mot v est facteur du mot $a^{|v|_a}vb^{|v|_b}$ qui appartient à D_1.
Pour tout mot $v \notin D_1$ tel que $|v|_a > |v|_b$, le mot $vb^{(|v|_a - |v|_b)}v$ a plusieurs $(D_1 + v)$-factorisations. Le code D_1 est donc maximal.
Tout mot u de D_1 s'écrit $au'b$ ou $bu'a$; en considérant le mot ${}^{\omega}(ab)u(aub)(ab)^{\omega}$ de ${}^{\omega}D_1{}^{\omega}$, on voit que u ne peut être supprimé. Donc D_1 est un générateur minimal de ${}^{\omega}D_1{}^{\omega}$.

Définition 10.2.1 *Soit C un code et deux mots x et y de C^*. Nous dirons que (x, y) est une* **paire synchronisante** *pour C si et seulement si*

$$\forall\, u, v \in A^* \ \ (\ uxyv \in C^* \ \Leftrightarrow \ ux, yv \in C^*\ )\ .$$

Proposition 10.2.4 *Tout code C possédant une paire synchronisante est un générateur minimal de ${}^{\omega}C^{\omega}$.*

Preuve : Considérons un code C possédant une paire synchronisante (x, y). Pour tout $v \in C$, on peut considérer le mot $w_v = {}^{\omega}vx(yv)^{\omega}$ de ${}^{\omega}C^{\omega}$. D'après l'hypothèse, toute C-décomposition d'un représentant de w_v coupe ce mot entre x et y. Or le mot yv appartient à C^* et puisque C est un code, $(yv)^{\omega}$ a une seule C-factorisation. Cette C-factorisation utilise le mot v, donc $C - \{v\}$ n'engendre pas ${}^{\omega}C^{\omega}$.

Le code C est donc un générateur minimal. •

Nous avons montré que les codes
- précirculaires,
- minces et maximaux,
- ayant une paire synchronisante,
sont des générateurs minimaux.

La classe $\mathcal{P}$ des codes précirculaires, la classe $\mathcal{M}$ des codes minces et maximaux, la classe $\mathcal{S}$ des codes ayant une paire synchronisante sont incomparables :
- le code semi-Dyck sur quatre lettres D'_2 est un Z-code mince, non maximal, mais il n'a pas de paire synchronisante ;
- le code $\{ab, ba\}$ est mince, mais il n'est ni précirculaire, ni maximal ; il possède une paire synchronisante (ab, ba) ;
- le code A^2 pour $A = \{a, b\}$ est un code mince, maximal ; il n'est pas précirculaire et n'a pas de paire synchronisante.

Les codes minces et maximaux et les codes possédant une paire synchronisante ont en commun la propriété d'être très minces. Cela nous conduit à rechercher si les codes très minces sont des générateurs minimaux.

Définition 10.2.2 *Un code C est* **très mince** *si C^* n'est pas inclus dans $F(C)$.*

Lemme 10.2.5 *Un code très mince est mince. Un code rationnel est très mince. Un code mince et maximal est très mince. Un code possédant une paire synchronisante est très mince.*

Preuve : Le premier fait est évident. Les deux résultats suivants sont connus [8]. Le dernier fait se démontre par contraposition :

Si un code C a une paire synchronisante, il a une paire synchronisante constituée de mots de C^+. Si un code C n'est pas très mince, pour tout couple (x, y) de mots de C^+, xy appartient à $F(C)$, et le couple (x, y) ne peut être une paire synchronisante. •

Lemme 10.2.6 *Soient B et C deux codes tels que $B \subset C$ et $F(B^*) = F(C^*)$. Si C est très mince, B est très mince.*

Preuve : Si B n'est pas très mince, B^* est inclus dans $F(B)$. On a alors : $C^* \subset F(C^*) = F(B^*) \subset F(B) \subset F(C)$, et donc C n'est pas très mince. •

La proposition suivante nous assure qu'un code très mince B est maximal parmi les codes C' vérifiant $F(C'^*) \subset F(B^*)$.

Proposition 10.2.7 *Soient B et C deux codes tels que $B \subset C$. Si B ou C est très mince et si $F(B^*) = F(C^*)$, alors $B = C$.*

Preuve : D'après le lemme précédent, le code B est très mince.

Si le code B est mince et complet, il est maximal et donc $B = C$.

Nous supposons maintenant que B n'est pas complet. A partir d'un automate (non nécessairement fini) émondé non ambigu reconnaissant B, on peut construire un automate émondé, non ambigu, de la forme $(Q, 1, 1, R)$ reconnaissant B^*, et considérer le morphisme de représentation φ associé: pour $a \in A$, $\varphi(a)$ est la matrice m de $\{0,1\}^{Q\times Q}$ telle que $m_{p,q} = 1$ si et seulement si $q \in R(p, a)$. Le stabilisateur $Stab(q)$ d'un état q est l'image par φ de l'ensemble des mots finis qui admettent une lecture de q à q. Puisque B est un code très mince non complet, $M = \varphi(A^*)$ est un monoïde transitif, non ambigu, de rang minimal $r(M)$ fini, contenant la matrice nulle [8]. Il existe dans $Stab(1)$ un idempotent e de rang $r(M)$ et $e = \varphi(x)$ où $x \in B^*$. Si $B \neq C$, il existe $y \in C - B$. Le mot xyx appartient à $F(C^*) = F(B^*)$ donc $\varphi(xyx)$ n'est pas nul. $\varphi(xyx) = e\varphi(y)e$ est dans le groupe des unités de eMe. Ce groupe est fini ([8]), donc il existe $n > 0$ tel que $(e\varphi(y)e)^n = e$. Donc $\varphi((xyx)^n) = e$ et le mot $(xyx)^n$ est dans B^*. Donc $B \cup \{y\}$, et, *a fortiori*, C ne sont pas des codes. •

Théorème 10.2.8 *Tout code C très mince est un générateur minimal de ${}^\omega C^\omega$.*

Preuve : Soit C un code très mince. Posons $L = {}^\omega C^\omega$. On a $F(L) = F(C^*)$.

Pour tout $u \in C$, le code $B_u = C - \{u\} \subset C$ ne peut vérifier $F(B_u{}^*) = F(C^*)$ puisque C est très mince, et $B_u \neq C$. Donc il existe un mot v de $F(L)$ n'appartenant pas à $F(B_u{}^*)$. On peut prolonger v en un mot de L. Ce mot de L ne se factorise pas sur B_u, donc B_u n'engendre pas L. Le code C est donc un générateur minimal de L. •

On obtient en corollaire du théorème précédent le résultat très important suivant :

Théorème 10.2.9 *Tout code C rationnel est un générateur minimal de ${}^\omega C^\omega$.*

Nous avons montré que les codes
- précirculaires,
- très minces

sont des générateurs minimaux.

Cela ne recouvre cependant pas tous les cas. L'exemple déjà cité du code de Dyck D_1 sur $A = \{a, b\}$ montre qu'un code peut être un générateur minimal, alors même qu'aucune de ces conditions n'est réalisée (Le code D_1 n'est ni mince, ni précirculaire).

Désignons par $\mathcal{T}$ la classe des codes très minces. Il n'y a pas d'inclusion entre les différentes classes $\mathcal{P}$, $\mathcal{T}$, $\mathcal{M}$ et $\mathcal{S}$ sauf celles déjà indiquées. De plus, les éléments de la partition définie à partir des classes $\mathcal{P}$ et $\mathcal{T}$ sont non vides,

et ceux de la partition de $\mathcal{T}$ définie à partir des classes $\mathcal{M}$, $\mathcal{S}$ et $\mathcal{T} \bigcap \mathcal{P}$ le sont aussi. En effet :
- en ajoutant une lettre à l'alphabet, un code de $\mathcal{P}$, $\mathcal{T}$, $\mathcal{S}$, (ou de leurs complémentaires) ne change pas de classe ; par contre, un code maximal cesse de l'être ;
- $\{a\}$ est précirculaire, très mince, a une paire synchronisante, et est maximal quand l'alphabet est $\{a\}$;
- $\{a^2\}$ est précirculaire, très mince, n'a pas de paire synchronisante, et est maximal quand l'alphabet est $\{a\}$;
- $\{a^2, b^2, ab, ba, c, ac, bc\}$ n'est pas précirculaire, est très mince, a une paire synchronisante (c, c), et est maximal quand l'alphabet est $\{a, b, c\}$;
- le code $D_1 + c$, où D_1 est le code de Dyck sur $\{a, b\}$, n'est pas précirculaire, ni extrêmement mince, ni maximal, mais a une paire synchronisante (c, c).
- si D'_2 désigne le code semi-Dyck sur $\{a, a', b, b'\}$ où les couples de parenthèses sont (a, a') et (b, b'), le code $D'_2 + ab'$ est un Z-code qui n'est ni très mince, ni maximal mais qui possède une paire synchronisante (ε, ab').

Nous avons vu que l'étude réalisée ne recouvre cependant pas tous les cas. Cela nous mène à la conjecture suivante :

Conjecture 2 *Tout code C est générateur minimal de ${}^{\omega}C^{\omega}$.*

Remarquons qu'il peut exister des générateurs minimaux qui ne sont pas des codes, même quand il existe un code générateur.

Exemple : Posons $A = \{a, b\}$. A est un code fini générateur de ${}^{\omega}A^{\omega}$. Le langage $G = \{a, aba, bab, b^2, b^3\}$ n'est pas un code. C'est pourtant un générateur de ${}^{\omega}A^{\omega}$. En effet, dans un mot de ${}^{\omega}A^{\omega}$, seuls les b isolés pourraient poser problème, et il n'en est rien, car : aba, $a.bab.a$, $aba.bab.a$,... appartiennent à G^*.
Le langage G est un générateur minimal de ${}^{\omega}A^{\omega}$:
- a ne peut être ôté, car ${}^{\omega}a^{\omega}$ appartient à ${}^{\omega}A^{\omega}$;
- aba ne peut être ôté, car ${}^{\omega}aba^{\omega}$ appartient à ${}^{\omega}A^{\omega}$;
- bab ne peut être ôté, car ${}^{\omega}ababa^{\omega}$ appartient à ${}^{\omega}A^{\omega}$;
- b^2 ne peut être ôté, car ${}^{\omega}ab^2a^{\omega}$ appartient à ${}^{\omega}A^{\omega}$;
- b^3 ne peut être ôté, car ${}^{\omega}ab^3a^{\omega}$ appartient à ${}^{\omega}A^{\omega}$.

10.3 Bissections d'un langage bi-infinitaire

Dans le cas infinitaire, s'il existe des générateurs finis, certains problèmes sont plus simples à résoudre. En particulier, on sait trouver des générateurs minimaux finis. Cela nous mène à rechercher s'il existe des générateurs finis.

Dans cette section, nous verrons qu'il n'est pas facile d'appliquer les résultats connus dans le cas infinitaire.

Comme les exemples suivants le montrent, l'étude des générateurs dans le cas bi-infinitaire ne se ramène pas immédiatement à celle des générateurs dans le cas infinitaire. On va cependant essayer d'établir des liens entre ces deux études.

Bien sûr, si $R^\omega = R'^\omega$ et ${}^\omega R = {}^\omega R'$, on a ${}^\omega R^\omega = {}^\omega R'^\omega$. Cependant, la réciproque est fausse. Par exemple, si $R = ab$ et $R' = ba$, on a ${}^\omega R^\omega = {}^\omega R'^\omega$ mais $R^\omega \neq R'^\omega$ et ${}^\omega R \neq {}^\omega R'$.

De plus, l' hypothèse « $R^\omega = R'^\omega$ » ne suffit pas pour que ${}^\omega R^\omega = {}^\omega R'^\omega$. Par exemple, si $R = \{a, ab, ba\}$ et $R' = a + (ab)^* ba$. On a $R^\omega = R'^\omega$ mais ${}^\omega(ab)(ba)^\omega$ appartient à ${}^\omega R^\omega$ et non à ${}^\omega R'^\omega$.

10.3.1 Bissections

Définitions 10.3.1 *Un couple (L', L'') de langages $L' \subset {}^\omega A$ et $L'' \subset A^\omega$ est appelé* **bissection** *d'un langage $L \in^\omega A^\omega$ si on a : $L = L'.L''$. Une bissection (L', L'') est dite* **rationnelle** *si les langages L' et L'' sont rationnels. Une bissection (L', L'') est dite* **fermée** *si L' est une adhérence gauche et L une adhérence droite. Une bissection (L', L'') est dite* **générative** *s'il existe $G \subset A^*$ tel que $L' = {}^\omega G$ et $L'' = G^\omega$, G est alors appelé générateur de (L', L'').*

Le langage $L =^\omega ab^\omega +^\omega ba^\omega$ n'a pas de bissection.

Le langage $L = {}^\omega a^\omega$ a une seule bissection : $({}^\omega a, a^\omega)$. Cette bissection est générative, fermée, rationnelle.

L'ensemble des bissections d'un langage L sera noté $\mathcal{B}(L)$. On le munit de l'ordre partiel défini par l'inclusion des composantes :

$$(L', L'') \subset (L'_1, L''_1) \Leftrightarrow L' \subset L'_1 \text{ et } L'' \subset L''_1 .$$

L'ensemble des bissections génératives de L sera noté $[[L]]$.

Nous allons étudier les maximaux de $\mathcal{B}(L)$ quand L est rationnel.

Etant donné un automate de Büchi $\Omega = (Q, I, T, R)$ (non nécessairement fini), pour $q \in Q$, nous notons $\Omega_{q,T}$ (resp. $\Omega_{q,-I}$) l'ensemble des mots infinis à droite (resp. à gauche) qui ont une lecture sur Ω, partant de (resp. arrivant en) q et passant une infinité de fois par un état de T (resp. I).

On peut considérer les relations d'équivalence suivantes :
- pour u et $v \in A^\omega$: $u \sim_d v \Leftrightarrow \forall q \in Q\ (u \in \Omega_{q,T} \Leftrightarrow v \in \Omega_{q,T})$,
- pour u et $v \in {}^\omega A$: $u \sim_g v \Leftrightarrow \forall q \in Q\ (u \in \Omega_{q,-I} \Leftrightarrow v \in \Omega_{q,-I})$.

Quand l'automate est fini, ces relations d'équivalence ont un nombre fini de classes (indexé par 2^Q), et les classes sont des langages rationnels.

Soit $L' \subset {}^{\omega}A$ et $L'' \subset A^{\omega}$, nous noterons $\boldsymbol{Sat_g(L')}$ le saturé de L' pour $\sim_g$ et $\boldsymbol{Sat_d(L'')}$ le saturé de L'' pour $\sim_d$.

Lemme 10.3.1 *Considérons un automate de Büchi (non nécessairement fini) et le langage de mots bi-infinis reconnu L. Si $\mathcal{B}(L)$ n'est pas vide, pour tout (L', L'') de $\mathcal{B}(L)$, le couple défini précédemment $(Sat_g(L'), Sat_d(L''))$ appartient à $\mathcal{B}(L)$.*

Preuve : Il est clair que $L = L'.L'' \subset Sat_g(L').Sat_d(L'')$.

Réciproquement, si $u' \in L'$ et $u'' \in L''$, on a une lecture réussie de $u'u''$ sur l'automate Ω. Si $u' \sim_g v'$ et $u'' \sim_d v''$, on en déduit une lecture réussie de $v'v''$ sur l'automate Ω, et $v'v''$ appartient à L. •

Proposition 10.3.2 *Si $L \subset {}^{\omega}A^{\omega}$ est rationnel, on peut décider si $\mathcal{B}(L)$ est vide ou non. Si $\mathcal{B}(L)$ n'est pas vide, $\mathcal{B}(L)$ a un nombre fini, calculable d'éléments maximaux. Ces maximaux sont des bissections rationnelles constructibles. Toute bissection de L est incluse dans une bissection maximale.*

Preuve : Considérons un automate de Büchi (Q, I, T, R) reconnaissant L et les relations $\sim_g$ et $\sim_d$. D'après le lemme précédent, si $\mathcal{B}(L)$ n'est pas vide, ses éléments maximaux existent, et sont à rechercher parmi les couples de la forme $(\bigcup C'_i, \bigcup C''_j)$ où les C'_i sont des classes pour $\sim_g$ et les C''_j des classes pour $\sim_d$. Il suffit donc de tester si ces couples (en nombre fini) sont des bissections de L, puis de déterminer ceux qui sont maximaux. •

Si $\mathcal{B}(L)$ n'est pas vide, il peut y avoir une ou plusieurs bissections maximales. Par exemple, le langage $L = {}^{\omega}(ab)^{\omega}$ a deux bissections : $({}^{\omega}(ab), (ab)^{\omega})$ et
$({}^{\omega}(ba), (ba)^{\omega})$. Ces bissections sont maximales, rationnelles et génératives.

La proposition suivante montre que la situation n'est pas simple.

Proposition 10.3.3 *Il existe des langages bi-infinitaires rationnels ayant une infinité de bissections rationnelles emboîtées et des bissections non rationnelles.*

Exemple : En considérant l'automate *ad hoc* construit à partir de l'automate minimal reconnaissant $\{ab, ba\}$, on obtient trois bissections maximales (rationnelles) du langage $L = {}^{\omega}(ab + ba)^{\omega}$:
$({}^{\omega}(ab+ba), (ab+ba)^{\omega})$, $({}^{\omega}(ab+ba)a, b(ab+ba)^{\omega})$ et $({}^{\omega}(ab+ba)b, a(ab+ba)^{\omega})$.

Posons $L'_0 = {}^{\omega}(ab + ba)$ et $L''_0 = (ab + ba)^{\omega}$, on obtient une infinité de bissections rationnelles emboîtées (L'_0, L''_i) en posant $L''_i = L''_{i-1} - \{w_i\}$

où $w_i = ab(ab(ba)^i)^\omega$. En effet, tout mot bi-infini de la forme uw_i s'écrit $uab.(ab(ba)^i)^\omega$ où $uab \in L'_0$ et $(ab(ba)^i)^\omega \in L''_j$, pour tout j.

Les langages L' et L'' suivants ne sont pas rationnels :
$L' = L'_0 - \{...ab(ba)^3ab(ba)^2ab(ba)^1ab\}$, $L'' = L''_0 - \{ab(ba)^1ab(ba)^2ab(ba)^3...\}$, mais le couple (L', L'') est une bissection de L. •

10.3.2 Bissections génératives

La proposition suivante nous amène à étudier les bissections génératives.

Proposition 10.3.4 *Soit L un langage de mots bi-infinis, on a :*

$$[L] = \bigcup_{(L',L'')\in[[L]]} ({}_\omega[L'] \cap [L'']_\omega) \, .$$

Preuve : Si G appartient à $[L]$, $({}^\omega G, G^\omega)$ est une bissection de L, et G appartient à ${}_\omega[{}^\omega G] \cap [G^\omega]_\omega$. Inversement, si G appartient à ${}_\omega[L'] \cap [L'']_\omega$, et si $L'.L'' = L$, on a ${}^\omega G^\omega = L$. •

L'ensemble $[[L]]$ peut être vide, même si $\mathcal{B}(L)$ n'est pas vide. Par exemple, le langage $L = {}^\omega ab^\omega$ a une infinité de bissections : $({}^\omega ab^n, b^\omega)$, mais aucune bissection générative.

Lemme 10.3.5 *On peut tester si une bissection rationnelle d'un langage (rationnel) donné est une bissection générative. Si c'est le cas, elle a un nombre fini, calculable de générateurs maximaux. Ces maximaux sont rationnels et constructibles et tout générateur de cette bissection est inclus dans un générateur maximal. On peut aussi tester si elle admet un générateur fini.*

Preuve : Soit (L', L'') une bissection rationnelle d'un langage donné L. Si G est un générateur de (L', L''), G appartient à ${}_\omega[L'] \cap [L'']_\omega$; il existe un générateur maximal M' de L' et un générateur maximal M'' de L'' tels que $G \subset M' \cap M''$. $M' \cap M''$ est donc un générateur commun à L' et L''. Les générateurs maximaux de (L', L'') sont donc de la forme $M' \cap M''$, et sont rationnels et constructibles. Il suffit donc de tester si ${}_\omega[L']$ et $[L'']_\omega$ sont non vides, et si les $M' \cap M''$ sont des générateurs de (L', L'').

Pour savoir si une bissection rationnelle générative (L', L'') est finiment engendrée, il suffit de tester si l'un de ses générateurs maximaux M (en nombre fini) contient un générateur fini. Posons $M_u = \{v \in M \mid |v| < |u|\}$. D'après [42], M contient un générateur fini de M^ω si et seulement si :
$M_- = \{u \in M \mid uM^\omega \not\subset M_u M^\omega\}$ est fini. De façon symétrique, posons :
${}_-M = \{u \in M \mid {}^\omega Mu \not\subset {}^\omega M M_u\}$. On obtient finalement : M contient un générateur fini de (L', L'') si et seulement si ${}_-M \cup M_-$ (qui est un générateur de (L', L'')) est fini. •

Le problème est alors d'étudier toutes les bissections rationnelles génératives d'un langage donné. Remarquons qu'une bissection rationnelle, même maximale, n'est pas toujours générative.

Exemple : La bissection maximale (rationnelle) $({}^{\omega}(ab+ba)a, b(ab+ba)^{\omega})$ du langage $L = {}^{\omega}(ab+ba)^{\omega}$ n'est pas générative. En effet, si nous supposons que le langage $b(ab+ba)^{\omega}$ est de la forme G^{ω} :
- pour tout n, $(b(ba)^n b)^{\omega} \notin G^{\omega}$ et $b(ba)^{\omega} \in G^{\omega}$, donc il existe n tel que $b(ba)^n \in G$,
- pour tout m, $b(ab)^m \notin G$ puisque $b(ab)^m b(ba)^{\omega} \notin G^{\omega}$,
- or $b(ab)^{\omega} \in G^{\omega}$, donc il existe m tel que $b(ab)^m a \in G$,
- il s'ensuit que $b(ba)^n b(ab)^m a(b(ba)^n)^{\omega} \in G^{\omega}$, ce qui est faux.

Le langage $b(ab+ba)^{\omega}$ ne peut être de la forme G^{ω}. •

Nous dirons qu'une bissection générative est **générative-maximale** si elle est maximale en tant que bissection générative.

Proposition 10.3.6 *Si L est un langage rationnel de mots bi-infinis tel que $[[L]]$ n'est pas vide, toute bissection générative est incluse dans une bissection générative-maximale. Ces bissections génératives-maximales sont rationnelles, constructibles et en nombre fini.*

Preuve : Considérons un automate de Büchi (Q, I, T, R) reconnaissant L et la congruence de Büchi associée. Si (L', L'') est une bissection générative de générateur G, on a vu que $Sat(G)$ appartient à $[L]$. Or $Sat(G)$ engendre une bissection générative rationnelle, contenant (L', L''). Les bissections génératives maximales existent donc et sont à rechercher parmi les bissections engendrées par les unions de classes (en nombre fini) pour la congruence de Büchi. •

Il peut exister une ou plusieurs bissections génératives-maximales. Nous en avons vu des exemples.

La proposition suivante montre que la situation ne se simplifie pas en ne considérant que les bissections génératives.

Proposition 10.3.7 *Il existe des langages rationnels de mots bi-infinis ayant une infinité de bissections génératives rationnelles emboîtées et des bissections génératives non rationnelles.*

Exemple : Posons $G_0 = a^+b$, $M = (a^+b)^+a^*$ et, pour tout $n > 0$, $G_n = G_{n-1} + a^+ba^n$. On a : ${}^{\omega}G_0{}^{\omega} = {}^{\omega}M^{\omega} = {}^{\omega} G_n{}^{\omega}$ et $G_0{}^{\omega} = G_n{}^{\omega} = M^{\omega}$, pour tout n. On a aussi : ${}^{\omega}G_{n-1} \subset {}^{\omega}G_n \subset {}^{\omega}M$ et ${}^{\omega}G_{n-1} \neq {}^{\omega}G_n \neq {}^{\omega}M$.

Posons $G = \bigcup_{p \text{ premier}} a^+ba^p \bigcup G_0$. On a: ${}^\omega G^\omega = {}^\omega M^\omega$.
Le langage $U = \bigcup_{p \text{ premier}} \{a^p\} \bigcup \{\varepsilon\}$ n'est pas rationnel.
Si ${}^\omega G$ était rationnel, $({}^\omega G_0)^{-1}({}^\omega G) = G_0{}^*U$ le serait aussi, ainsi que U car $U = G_0{}^*U \cap a^*$. On en déduit que $({}^\omega G, G^\omega)$ est une bissection non rationnelle générative de ${}^\omega G^\omega$.

10.3.3 Bissections des biadhérences

Pour essayer d'obtenir des résultats moins négatifs, nous pouvons tenter de restreindre notre étude. Si notre but est d'étudier les générateurs finis, nous pouvons nous limiter à l'étude des bissections génératives fermées des biadhérences. Nous allons voir que la situation ne s'améliore pas, même dans le cas d'une biadhérence finiment engendrée.

Remarquons tout de suite qu'il existe des langages rationnels, qui ne sont pas des biadhérences, qui ont cependant des bissections fermées.

Exemple : Soit $U = a^*bc^*$. On a: $Adh(U) = a^*bc^\omega + a^\omega$ et $Ladh(U) = {}^\omega c + {}^\omega abc^*$. Le langage rationnel $L = Ladh(U).Adh(U)$ a une bissection fermée (rationnelle). Le mot ${}^\omega a^\omega$ n'appartient pas à L, mais il appartient à sa fermeture, donc L n'est pas fermé.

Il existe aussi des langages, qui ne sont pas des biadhérences, ayant cependant des bissections génératives fermées.

Exemple : Posons $G = \{a^n b^n a^n \mid n > 0\} \cup \{a\}$. Le langage ${}^\omega G^\omega$ n'est pas une biadhérence car il ne contient pas le mot ${}^\omega b^\omega$. Par contre, G^ω est une adhérence car $Adh(G) = a^\omega \subset G^\omega$. De même, ${}^\omega G$ est une adhérence gauche. Donc le langage ${}^\omega G^\omega$ a une bissection générative fermée.

Il existe aussi des biadhérences (finiment engendrées) possédant des bissections génératives non fermées.

Exemple : [42] Le langage $G = a + aba + b^+a^*b^+$ est un générateur de ${}^\omega(a+b)^\omega$, mais G^ω n'est pas une adhérence, ni ${}^\omega G$ une adhérence gauche.

On a cependant les propriétés suivantes:

Théorème 10.3.8 *Soit G inclus dans A^* tel que G^ω soit une adhérence et ${}^\omega G$ une adhérence gauche. On a:*
- ${}^\omega G^\omega$ est une biadhérence si et seulement si $Biadh(G)$ est inclus dans ${}^\omega G^\omega$,
- Si G est rationnel, ${}^\omega G^\omega$ est une biadhérence.

Preuve : On a:

$$\overline{{}^{\omega}G^{\omega}} = Biadh(G^*) = {}^{\omega}G^{\omega} \bigcup \overline{{}^{\omega}G}.\overline{G^{\omega}} \bigcup Biadh(G) .$$

Si G^{ω} et ${}^{\omega}G$ sont fermés, on a: $\overline{{}^{\omega}G^{\omega}} = {}^{\omega}G^{\omega} \bigcup Biadh(G)$, d'où l'équivalence annoncée.

Etudions maintenant le cas rationnel. Considérons l'automate déterministe minimal $\Omega = (Q, q_0, T, R)$ d'un langage G rationnel. Soit w un Z-mot de $Biadh(G)$. Pour tout n, il existe des mots u_n et v_n tels que $u_n w[-n, n]v_n \in G$. Ces mots ont une lecture réussie (unique) sur l'automate. Dans ces lectures, juste avant de lire w_0, on se trouve dans un état q_n. Comme Q est fini, il existe un état q tel que $q = q_n$ pour une infinité de n (disons pour $n \in I$). Soit α un mot, le plus court possible, que l'on peut lire de q_0 à q. Le mot αw_+ a une infinité de facteurs gauches dans G et donc appartient à $Adh(G)$, donc à G^{ω} qui est fermé. Il existe donc $\beta < w_+$ tel que $\alpha\beta \in G^+$ et $(\beta)^{-1}w_+$ soit dans G^{ω}. Si β est le plus court possible, $\alpha\beta$ appartient à G, et β est lu sur l'automate de q à un état final q_t.

Pour prouver que w est dans ${}^{\omega}G^{\omega}$, il suffit de prouver que $w_-.\beta$ appartient à ${}^{\omega}G$. Comme cet ensemble est fermé, il suffit d'avoir $w_-.\beta \in Ladh(G)$. Or les mots $u_n w[-n, 0[.\beta$, pour $n \in I$, sont lus de q_0 à q_t, donc on a bien $w_-.\beta \in Ladh(G)$. •

Proposition 10.3.9 *Soit L une biadhérence. Toute bissection générative est incluse dans une bissection fermée. Les bissections maximales et génératives sont fermées.*

Preuve : Soit G un générateur d'une biadhérence L. On a: $L = {}^{\omega}G^{\omega} = Biadh(G^*)$ et aussi:

$${}^{\omega}G^{\omega} \subset \overline{{}^{\omega}G}.\overline{G^{\omega}} = ({}^{\omega}G \cup Ladh(G)G^*).(G^{\omega} \cup G^*Adh(G)) \subset Biadh(G^*) .$$

Donc $(\overline{{}^{\omega}G}, \overline{G^{\omega}})$ est une bissection (fermée) de L, contenant la bissection $({}^{\omega}G, G^{\omega})$. Une bissection générative $({}^{\omega}G, G^{\omega})$ ne peut être maximale que si elle est égale à $(\overline{{}^{\omega}G}, \overline{G^{\omega}})$, c'est-à-dire si elle est fermée. •

Remarquons qu'une bissection maximale fermée n'est pas toujours générative. Par exemple, la bissection maximale fermée $({}^{\omega}(ab + ba)a, b(ab + ba)^{\omega})$ du langage $L = {}^{\omega} (ab + ba)^{\omega}$ n'est pas générative.

Les fermetures des bissections génératives d'une biadhérence ne sont pas toujours génératives:

Exemple : Considérons le code infini $C = a + b^2 + ab(b^2)^*a + ab(b^2)^*ab$. Le langage C engendre en fait ${}^\omega A^\omega$ où $A = \{a, b\}$. En effet, dans un mot de ${}^\omega A^\omega$, seuls les b^{2n+1} pourraient poser problème, et il n'en est rien, car : $ab^{2n+1}a.b^{2p}.a$ et $ab^{2n+1}ab.b^{2p}.a$ appartiennent à C^*.

Donc ${}^\omega C^\omega$ est une biadhérence. Puisque $Adh(C) = ab^\omega$, C^ω est une adhérence. Par contre, $\overline{{}^\omega C} = {}^\omega C + {}^\omega babC^*$. Le couple $(\overline{{}^\omega C}, C^\omega)$ est une bissection fermée de ${}^\omega A^\omega$. Nous allons montrer qu'elle n'est pas générative.

Supposons qu'un langage G engendre cette bissection. Puisque ${}^\omega bab \in {}^\omega G$, un des mots suivants appartient à G : b, ab ou b^nab pour un certain n. Nous allons exclure tous ces cas.

Il existe dans G une puissance de a. Les faits suivants :
- $ba^\omega \notin C^\omega$, donc $b \notin G$,
- $b^naba^\omega \notin C^\omega$, donc $b^nab \notin G$,
- ${}^\omega ab \notin \overline{{}^\omega C}$, donc $ab \notin G$,

montrent la contradiction.

La proposition suivante montre que la situation est vraiment catastrophique.

Proposition 10.3.10 *Il existe des langages de mots bi-infinis finiment engendrés, et ayant une infinité de bissections génératives finiment engendrées emboîtées.*

Exemple : Considérons $A = \{a, b\}$ et la biadhérence finiment engendrée ${}^\omega A^\omega$. Posons $G_0 = \{a, ab, bb\}$ et, pour tout $n > 0$, $G_n = G_{n-1} + ba^nb$.

Montrons que le langage G_0 est un générateur de ${}^\omega A^\omega$. Clairement, on a : ${}^\omega G_0 = {}^\omega A$ et $a^+b^+ \subset C^*$. Tout mot de ${}^\omega A^\omega$ est dans ${}^\omega Aa^\omega$, ou ${}^\omega Ab^\omega$, ou ${}^\omega A(a^+b^+)^\omega$, donc appartient à ${}^\omega C^\omega$.

Les langages G_0 et A^* étant générateurs de ${}^\omega A^\omega$, tous les G_n le sont aussi, et ils engendrent des bissections génératives finiment engendrées emboîtées. Ces bissections sont toutes distinctes car, pour tout $n > 0$, ba^nb^ω appartient à G_n^ω mais pas à G_{n-1}^ω.

Ce dernier exemple montre aussi qu'une bissection générative fermée n'est pas toujours maximale.

10.4 Existence de générateurs finis

Pour étudier l'existence de générateurs finis, il ne semble pas qu'on puisse se ramener aisément au cas infinitaire. Nous allons donc étudier le cas bi-infinitaire directement.

Pour décider s'il existe un générateur fini, il suffit de savoir décider si l'un des générateurs maximaux contient un générateur fini. On peut réduire le problème de la façon suivante:

Proposition 10.4.1 *[42] Soit R un langage rationnel de A^*. ${}^\omega R^\omega$ a un générateur fini si et seulement si les propriétés suivantes sont vérifiées:*
- ${}^\omega R^\omega$ est une biadhérence,
- il existe un ensemble fini G tel que $F(G^*) = F(R^*)$.

Preuve : Si G est un générateur fini de ${}^\omega R^\omega$, ${}^\omega R^\omega = {}^\omega G^\omega$ est une biadhérence, et $F(G^*) = F({}^\omega G^\omega) = F({}^\omega R^\omega) = F(R^*)$.

Réciproquement, si ${}^\omega R^\omega$ est une biadhérence, ${}^\omega R^\omega = Biadh(F(R^*))$. Si de plus $F(G^*) = F(R^*)$, ${}^\omega R^\omega = Biadh(F(G^*)) = {}^\omega G^\omega$ quand G est fini. •

On peut montrer qu'aucune des deux conditions indiquées n'est superflue:

Exemples: Pour $G = a + b$ et $R = ba^*$, on a: $F(G^*) = F(R^*)$ et cependant ${}^\omega R^\omega$ n'est pas une biadhérence.

Si l'on considère $G = a + b^2 + c + ab(b^2)^*c$, ${}^\omega G^\omega$ est une biadhérence non finiment engendrée [23].

Notation: Pour $m > 0$ et $R \subset A^*$, notons $R_m = \{u \in R \mid |u| \leq m\}$.

Pour démontrer le résultat essentiel de cette section, nous utilisons le résultat suivant de A. Restivo:

Proposition 10.4.2 *[55] Soit R un langage rationnel de A^*. On peut décider s'il existe un sous-ensemble fini G de R tel que $F(G^*) = F(R^*)$.*

Preuve : (La preuve est de A. Restivo; nous en présentons ici une version simplifiée.)

Montrons que, si il existe un sous-ensemble fini G de R tel que $F(G^*) = F(R^*)$, on peut trouver $m > 0$ tel que $F((R_m)^*) = F(R^*)$. Cela suffit pour obtenir la proposition annoncée.

Considérons la congruence syntaxique associée à R^*: pour u et v dans A^*,

$$u \sim v \Leftrightarrow \forall x, y \in A^*(\, xuy \in R^* \Leftrightarrow xvy \in R^* \,) .$$

Cette congruence est d'index fini k quand R est rationnel. Posons $d = k^2 + 2$. D'après [55], tout mot z de longueur d peut s'écrire $z = xvy$ où, pour tout n, $x \sim xv^n$ et $y \sim v^n y$.

Soit u un élément de $F(R^*)$ et w un mot de R^* prolongeant u et suffisamment long: $w = u'uu''$ où $|u'|$ et $|u''| > 2d$. Découpons w en facteurs de

longueur d: $w = z_0 z_1 ... z_t z'$ où $|z_i| = d$, $|z'| \leq d$. Pour tout i, il existe un découpage $z_i = x_i v_i y_i$ tel que $x_i \sim x_i v_i^n$ et $y_i \sim v_i^n y_i$ pour tout n.

Soit n la longueur maximale des éléments de G. Posons $z'_i = x_i (v_i)^n y_i$ et $w' = z'_0 z'_1 ... z'_t z'$. Le mot w', congru à w est dans R^*, donc dans $F(G^*)$ et a donc une G-interprétation. Une telle G-interprétation a au moins un indice de découpage à l'intérieur de chacun des $(v_i)^n$. Cela donne des mots de G^*, donc de R^*, de la forme $v'_i (v_i)^k y_i x_{i+1} (v_{i+1})^h v''_{i+1}$ (où v'_i est un suffixe de v_i et v''_{i+1} un préfixe de v_{i+1}). Pour tout i, $v'_i (v_i)^k y_i x_{i+1} (v_{i+1})^h v''_{i+1}$ est congru à $v'_i y_i x_{i+1} v''_{i+1}$, donc ces derniers mots appartiennent à R^*. Leur longueur est plus petite que $m = 2d$, donc ils appartiennent à $R_m{}^*$. Finalement u est un facteur de $R_m{}^*$. •

Théorème 10.4.3 *Soit $L \subset {}^\omega A^\omega$ un langage rationnel. On peut décider si L est finiment engendré et si c'est le cas, construire au moins un générateur fini.*

Preuve : On peut tester si L est une biadhérence et décider si $[L]$ n'est pas vide. Si c'est le cas, on peut construire les générateurs maximaux M (en nombre fini) et tester s'ils ont un sous-ensemble fini G tel que $F(G^*) = F(M^*)$.

Pour chaque M, on peut calculer un nombre $m(M)$ tel que, si M contient un générateur fini, M_m soit un générateur fini. •

La proposition suivante peut se déduire aisément des preuves des résultats précédemment énoncés. Nous en réécrivons cependant une preuve (reprenant les preuves précédentes) qui permet de voir où intervient le fait que ${}^\omega G^\omega$ est une biadhérence quand G est fini.

Corollaire 10.4.4 *Soit R un langage rationnel inclus dans A^*. Si il existe un sous-ensemble fini G de R tel que ${}^\omega G^\omega = {}^\omega R^\omega$, alors il existe $m > 0$ calculable tel que ${}^\omega R_m{}^\omega = {}^\omega R^\omega$.*

Preuve : Soit k l'index de la congruence syntaxique de R^*. Posons $d = k^2 + 2$. D'après [55], tout mot z de longueur d peut s'écrire $z = xvy$ où $x \sim xv^n$ et $y \sim v^n y$ pour tout n.

Soit w un Z-mot représentant un élément de ${}^\omega R^\omega$ et découpons w en facteurs de longueur d: $w = ...z_{-1} z_0 z_1 z_2 ...$ où $|z_i| = d$. Pour tout $i \in Z$, il existe un découpage $z_i = x_i v_i y_i$ tel que $x_i \sim x_i v_i^n$ et $y_i \sim v_i^n y_i$ pour tout n.

Soit n la longueur maximale des éléments de G. Posons $z'_i = x_i (v_i)^n y_i$ et $w' = ...z'_{-1} z'_0 z'_1 z'_2 ....$ Pour tout i, il existe un facteur de w, prolongeant $z_{-i} ... z_i$ et appartenant à R^*: $\alpha_i z_{-i} ... z_i \beta_i$. Le mot $\alpha_i z'_{-i} ... z'_i \beta_i$, congru au précédent, appartient aussi à R^*. Donc $z'_{-i} ... z'_i$ appartient à $F(R^*)$. Le mot w' appartient à $Bilim(F(R^*))$. Puisque ${}^\omega R^\omega$ est finiment engendré, c'est une biadhérence et ${}^\omega R^\omega = Biadh(R^*) = Bilim(F(R^*))$. Le mot w' est donc dans

${}^{\omega}R^{\omega} = {}^{\omega}G^{\omega}$, et a par conséquent une G-décomposition. Une telle G-décomposition a au moins un indice de découpage à l'intérieur de chacun des $(v_i)^n$. Cela donne des mots de la forme $v'_i(v_i)^k y_i x_{i+1}(v_{i+1})^h v''_{i+1}$ (où v'_i est un suffixe de v_i et v''_{i+1} un préfixe de v_{i+1}) qui sont dans G^*, donc dans R^*. Pour tout i, $v'_i(v_i)^k y_i x_{i+1}(v_{i+1})^h v''_{i+1}$ est congru à $v'_i y_i x_{i+1} v''_{i+1}$, donc ce mot appartient à R^*. Sa longueur est plus petite que $m = 2d$, donc il appartient à ${R_m}^*$. Finalement w représente un élément de ${}^{\omega}{R_m}^{\omega}$. •

Il est clair que tout générateur fini F de L contient un générateur minimal, et que l'on peut construire tous les générateurs minimaux contenus dans F : il suffit de tester pour tout sous-ensemble de F s'il engendre L.

On peut décider si un générateur donné contient un générateur fini, donc aussi décider s'il contient un générateur minimal fini.

La taille d'un ensemble fini est la somme des longueurs de ses éléments. On peut trouver tous les générateurs de plus petite taille d'un langage finiment engendré. Il suffit de trouver un générateur fini, de calculer sa taille T, de déterminer les ensembles de taille inférieure ou égale à T qui sont générateurs, et de les comparer.

Comme dans le cas infinitaire, on peut montrer que :

Proposition 10.4.5 *Si $L \subset {}^{\omega}A^{\omega}$ est finiment engendré, L a une infinité de générateurs finis minimaux deux à deux disjoints.*

Preuve : La preuve est la même que dans le cas infinitaire [42]. On construit, à partir d'un générateur fini minimal G_0, une suite de générateurs finis minimaux G_i. Il suffit de prendre pour G_{i+1} un générateur fini minimal inclus dans le générateur $(G_i)^{n_i}$. Si $n_i > Max\{|u| \mid u \in G_i\}$, les générateurs construits sont disjoints. •

Proposition 10.4.6 *Un langage finiment engendré peut avoir des générateurs minimaux infinis.*

Exemple : Posons $A = \{a, b\}$. A est un code fini générateur de ${}^{\omega}A^{\omega}$. Nous avons déjà vu (section 10.3.3) que le code infini $C = a + b^2 + ab(b^2)^*a + ab(b^2)^*ab$ est un générateur de L.
Le langage C est un générateur minimal de L : aucun mot ne peut être ôté si on veut pouvoir engendrer les mots : ${}^{\omega}a^{\omega}$, ${}^{\omega}b^{\omega}$ ou ceux de ${}^{\omega}aab(b^2)^*aa^{\omega}$ ou de ${}^{\omega}aab(b^2)^*aba^{\omega}$. •

Dans cet exemple, le langage considéré est ${}^{\omega}A^{\omega}$, engendré par l'alphabet fini $\{a, b\}$, et il a donc des générateurs minimaux qui sont des codes infinis. Le code considéré ici est maximal.

Tout code maximal qui engendre une biadhérence (en particulier, tout code maximal fini, mais il en existe aussi d'autres) engendre ${}^{\omega}A^{\omega}$. En effet, si ${}^{\omega}C^{\omega}$ est une biadhérence, comme C est complet, ${}^{\omega}C^{\omega} = Biadh(F(C^*)) = Biadh(A^*) = {}^{\omega}A^{\omega}$.

Il existe cependant des codes maximaux qui n'engendrent pas ${}^{\omega}A^{\omega}$. Par exemple, le code de Dyck D_1, ou encore le code $b + ab^*a$ qui n'engendrent pas le mot ${}^{\omega}bab^{\omega}$.

Nous ne savons pas s'il existe aussi des codes non maximaux qui engendrent ${}^{\omega}A^{\omega}$. Si la conjecture 2 est vraie, tout code engendrant ${}^{\omega}A^{\omega}$ sera maximal.

Pour terminer, rappelons les faits suivants, à propos du langage ${}^{\omega}A^{\omega}$ (où A est l'alphabet fini $\{a, b\}$) :
- ${}^{\omega}A^{\omega}$ a des générateurs minimaux finis qui ne sont pas des codes,
- ${}^{\omega}A^{\omega}$ a des générateurs minimaux finis qui sont des codes,
- ${}^{\omega}A^{\omega}$ a des générateurs minimaux infinis qui sont des codes.

Cet exemple montre encore que le cas bi-infinitaire est très différent du cas infinitaire, où tous les générateurs minimaux du langage engendré par un alphabet fini sont des codes finis (préfixes et maximaux) [42].

Conclusion

Les travaux réalisés dans cette thèse ont été motivés par l'étude des générateurs des langages infinitaires et celle des générateurs des langages bi-infinitaires.

Dans le cas des langages infinitaires, les problèmes intéressants sont relatifs à l'existence de générateurs particuliers. Parmi ces générateurs, on peut citer :
- les codes, qui sont des générateurs minimaux,
- les ω-codes, qui assurent un décodage unique des mots du langage donné,
- les codes à délai de déchiffrage borné (et parmi eux, les codes préfixes), qui permettent concrètement le décodage des mots du langage donné.

Seul le cas des générateurs préfixes des langages rationnels est pour l'instant résolu [44]. Nous avons pour notre part essayé de contribuer à l'étude des liens existant entre codes et factorisations des mots infinis.

Dans le cas des langages bi-infinitaires, l'étude de l'existence de générateurs particuliers intéressants semble, on l'a vu au dernier chapitre, plus difficile encore. D'ailleurs, peu d'auteurs s'y sont intéressés. Des problèmes, résolus dans le cas infinitaire, ne le sont pas dans le cas bi-infinitaire. Citons, par exemple :
- peut-on décider si un générateur donné est minimal?
- les codes sont-ils des générateurs minimaux?

Comme dans le cas infinitaire, se pose, dans le cas bi-infinitaire, la question de l'existence de générateurs qui assurent un décodage unique des éléments du langage donné. Nous avons vu, dans cette thèse, que deux notions de « code pour les mots bi-infinis » pouvaient être considérées. L'objet de ce travail a été d'éclaicir ce phénomène.

La notion de code précirculaire est apparue de façon naturelle. Elle est apparue d'une part comme une généralisation des codes circulaires, par l'unicité de la factorisation des mots périodiques. D'autre part, un code C étant générateur minimal du langage infinitaire C^ω et un code précirculaire C générateur minimal du langage bi-infinitaire ${}^\omega C^\omega$, la notion de code précirculaire est apparue, dans ce contexte, comme notion analogue à celle de code.

Dans le but d'étudier, pour un langage rationnel donné, l'existence de codes à délai (de déchiffrage, ou de synchronisation) borné générateurs, nous avons étudié les propriétés des langages engendrés par de tels codes. Cette étude a donné des résultats intéressants, en particulier que le langage ${}^{\omega}C^{\omega}$ est une bilimite quand C est un code à délai de synchronisation borné.

Cette étude n'a cependant pas suffi pour résoudre le problème de l'existence de codes à délai borné générateurs. Nous n'avons pas ici relaté les faits qui permettent d'avancer dans l'étude du problème, cependant une avancée très nette vers une solution serait réalisée si l'on savait étudier la famille des sous-monoïdes libres d'un sous-monoïde donné.

Finalement, nous avons étudié de façon plus précise la famille des générateurs d'un langage rationnel bi-infinitaire donné. Cela nous a permis de prouver que beaucoup de codes sont des générateurs minimaux (en particulier les codes rationnels), et de montrer la décidabilité de l'existence de générateurs finis.

Nous avons mené (presque) parallèlement l'étude du cas infinitaire et celle du cas bi-infinitaire. Cela nous a permis d'établir à quel point ces deux cas sont différents en ce qui concerne les factorisations des mots (infinis et bi-infinis) et les propriétés des familles de générateurs.

Bibliographie

[1] E. BARBIN-LE REST et M. LE REST, Sur la combinatoire des codes à deux mots, *Theoret. Comput. Sci.*, 41 (1985), 61-80.

[2] M. P. BEAL, Codages, automates locaux et entropie, Thèse, *Publications du LITP*, Univ. Paris VII, 38 (1988).

[3] M. P. BEAL et D. PERRIN, Une caractérisation des ensembles sofiques, *C. R. Acad. Sc. Paris*, 303 I 6 (1986), 255-257.

[4] D. BEAUQUIER, Automates sur les mots bi-infinis, Thèse d'état, *Publications du LITP*, Univ. Paris

[5] D. BEAUQUIER, Thin homogeneous sets of factors, *Found. of Soft. Techn. and Theor. Comput. Sci., Lectures notes in Computer Science*, 241 (1986), 239-251.

[6] D. BEAUQUIER, Ensembles homogènes minces, ICALP (1986).

[7] D. BEAUQUIER, Minimal automaton for a factorial, transitive and rational language, *Theoret. Comput. Sci.*, 67 (1989), 65-73.

[8] J. BERSTEL and D. PERRIN, *Theory of codes* (Academic Press, Orlando, 1985).

[9] F. BLANCHARD, Codes engendrant certains systèmes sofiques, *Theoret. Comput. Sci.*, 68 (1989), 253-265.

[10] F. BLANCHARD and G. HANSEL, Systèmes codés, *Theoret. Comput. Sci.*, 44, (1986),17-49.

[11] L. BOASSON and M. NIVAT, Adherences of languages, *J. Comput. System. Sci.*, 20, (1980), 285-309.

[12] J. R. BUCHI, On a decision method in restricted second-order arithmetic, *Proc. Congr. Logic*, (Stanford Univ. Press, Stanford, 1962) 1-11.

[13] V. BRUYERE, Codes préfixes - Codes à délai de déchiffrage borné, Thèse (Univ. Paris VII, 1989).

[14] V. BRUYERE, Codes, Thèse d'état (Univ. de Mons-Hainaut), (1991).

[15] V. BRUYERE, Research topics in the theory of codes, Asmics workshop Codes and Automata, (1991).

[16] V. BRUYERE, Wang Limin, Zhang Liang, On completion of codes with finite deciphering delay,*European J. Combinatorics*, 11(1990), 513-521.

[17] R. M. CAPOCELLI, Finite decipherability of weakly prefix codes, *Algebra, Combinatorics and Logic in Computer Science* , (Coll. Math. Soc. J. Bolyai 42, North Holland, Amsterdam, 1985) 175-184.

[18] R. CORI, Codes à délai borné maximaux, *Théorie des codes, Publications du LITP*, Univ. Paris VI et Paris VII, (1979), 57-74.

[19] J. DEVOLDER, Extension des morphismes aux mots bi-infinis, *Publications du LIFL*, I.T. 156 (Lille, 1988).

[20] J. DEVOLDER, Comportement des codes vis-à-vis des mots infinis et bi-infinis – Comparaison, *Actes des journées franco-belges – Codes et automates*, Rouen (1991).

[21] J. DEVOLDER, Precircular codes and periodic bi-infinite words, *Publications du LIFL*, I.T. 175 (Lille, 1989), à paraître dans *Information and Computation.*

[22] J. DEVOLDER, M. LATTEUX, I. LITOVSKY and L. STAIGER, Codes and infinite words, *Publications du LIFL*, I.T. 220 (Lille, 1991).

[23] J. DEVOLDER and I. LITOVSKY, Finitely generated biω- langages, *Theoret. Comput. Sci.*, 85 (1991), 33-52.

[24] J. DEVOLDER and E. TIMMERMAN, Finitary codes for biinfinite words, *RAIRO Inform. Theoret. Applic.* 26 (1992) tome 4, 363-386.

[25] DO LONG VAN, Codes avec des mots infinis, *R.A.I.R.O. inform. theor.*, 16 (1982), 371-386.

[26] NGUYEN HUONG LAM and DO LONG VAN, On a class of infinitary codes, *RAIRO Theor. Info. App.*, 24 (1990), 5, 441-458.

[27] DO LONG VAN, NGUYEN HUONG LAM and PHAN TRUNG HUY, On codes concerning bi-infinite words, préprint (Bordeaux, 1990).

[28] DO LONG VAN, D. G. THOMAS, K. G. SUBRAMANIAN and R. SIROMONEY, Bi-infinitary codes, ***RAIRO Theor. Info. App.***, 24 (1990), 1, 67-87.

[29] S. EILENBERG, ***Automata, Languages, and Machines***, (Academic press, New-York, 1974, 1976).

[30] A. EHRENFEUCHT and G. ROZENBERG, Each regular code is included in a regular maximal code, ***RAIRO Inform. Theoret. Applic.***, 17 (1983), 89-96.

[31] F. GIRE and M. NIVAT, Langages algébriques de mots bi-infinis, *Theoret. Comput. Sci.*, (1986).

[32] J. KARHUMAKI, On three-element codes,*Theoret. Comput. Sci.*, 40 (1985), 3-11.

[33] H. LANDWEBER, Decision problem for ω-automata, *Math. Syst. Theor*, 3 (1969), 376-384.

[34] J. L. LASSEZ, Circular codes and synchronisation, *Internat. Journal of Computer and syst. Sci.* , 5 (1976) 201-208.

[35] M. LATTEUX and E. TIMMERMAN, Finitely generated ω-languages, *Inform. Proc. Letters* 23 (1986), 171-175.

[36] M. LECONTE, K-th power-free codes, Automata on infinite words, *Lectures notes in Computer Science* 192 (1984), 172-187.

[37] M. LECONTE, Codes sans répétition, Thèse de 3$^{\text{ème}}$ cycle, *Publications du LITP*, Univ. Paris VII, 56 (1985).

[38] A. LENTIN and M. P. SCHUTZENBERGER, A combinatorial problem in the theory of free monoids, *Proc. Univ. of North Carolina*, (1967), 128-144.

[39] V. I. LEVENSHTEJN, Some properties of coding and self adjusting automata for decoding messages, *Problemy Kibernetiki*, 11 (1964) 63-121.

[40] R. LINDNER and L. STAIGER, Algebraische Codierungstheorie, *Theorie der sequentiellen Codierungen* (Akademie-Verlag, Berlin,1977).

[41] I. LITOVSKY and E. TIMMERMAN, On generators of rational ω-power languages, *Theoret. Comput. Sci.*, 53 (1987), 187-200.

[42] I. LITOVSKY, Générateurs des langages rationnels de mots infinis, Thèse Univ. Lille I (1988).

[43] I. LITOVSKY, Minimal generators of submonoids of A^∞, *RAIRO Inform. Theoret. Applic.*, 25 (1991), tome 1, 3-17.

[44] I. LITOVSKY, Prefix-free languages as ω-generators, *Inform. Proc. Letters* 37 (1991), 61-65.

[45] M. LOTHAIRE, Combinatorics on words, *Encyclopedia of mathematics and its applications*, 17 (Addison-Wesley 1983).

[46] A. de LUCA and A. RESTIVO, On some properties of very pure codes, *Theoret. Comput. Sci.*, (1980), 157-170.

[47] R.C. LYNDON and M.P. SCHUTZENBERGER, The equation $a^m = b^n c^p$ in a free group, *Michigan Math. J.* (1962) 289-298.

[48] R. J. McELIECE, The theory of Information and Coding, *Encyclopedia of mathematics and its applications*, 3 (Addison-Wesley 1977).

[49] R. McNAUGHTON, Testing and generating infinite sequences by a finite automaton, *Information and Control*, 9 (1966), 521-530.

[50] M. NIVAT and D. PERRIN, Ensembles reconnaissables de mots bi-infinis, *Proc. 14e Symp. on Theory of Computing* (1982), 47-59.

[51] D. PERRIN, Completing biprefix codes, *Theoret. Comput. Sci.*, 28 (1984), 329-336.

[52] D. PERRIN et J. E. PIN, Mots infinis, *Publications du LITP*, Univ. Paris VI et VII, 91.06 (1991).

[53] A. RESTIVO, A combinatorial property of codes having finite synchronisation delay, *Theoret. Comput. Sci.*, 1 (1975), 95-101.

[54] A. RESTIVO, On codes having no finite completions, *Discrete Math.*, 17 (1977), 309-316.

[55] A. RESTIVO, Finitely generated sofic systems, *Theoret. Comput. Sci.*, 65 (1989), 265-270.

[56] A. RESTIVO, Codes and automata, *Lectures notes in Computer Science*, 386 (1989), 186-198.

[57] A. RESTIVO, Codes and local constraints, *Theoret. Comput. Sci.*, 72 (1990), 55-64.

[58] M. P. SCHUTZENBERGER: Une théorie algébrique du codage, Séminaire Dubreil-Pisot, 15 (1955-56), Institut Henri Poincaré, Paris.

[59] M. P. SCHUTZENBERGER: On a question concerning certain free monoids, *J. Combin. Theory*, 1 (1966),437-442.

[60] C. E. SHANNON, A mathematical theory of Communication, *Bell. System Tech. J.* 27 (1948), 379-656.

[61] L. STAIGER, On infinitary finite length codes, *RAIRO Theor. Inform. and Applic.*, 20 (1986) 4, 483-494.

Table des matières

Introduction **3**

Préliminaires **11**
0.1 Définitions et notations 11
0.2 Factorisations et décompositions 12
0.3 Langages rationnels, langages reconnaissables 14
0.3.1 Reconnaissance 14
0.3.2 Rationalité 15
0.3.3 Automates *ad hoc* 15
0.4 Langages infinitaires 16
0.4.1 Langages infinitaires particuliers 16
0.4.2 Langages bi-infinitaires particuliers 17

1 Caractérisations des codes **21**
1.1 Codes et factorisation unique 22
1.2 Forme des factorisations des mots rationnels 23
1.3 Nombre de factorisations des mots rationnels 25

2 Codes précirculaires **29**
2.1 Notion de code précirculaire 30
2.2 Codes précirculaires et mots bi-infinis 31
2.3 Codes circulaires et Z-mots 32
2.4 Codes presque circulaires et codes précirculaires 34

3 Codes pour les mots infinis **37**
3.1 Notions de ω-code, de biω-code et de Z-code 38
3.2 Caractérisations des Z-codes 40
3.3 Caractérisations des biω-codes 42

4 Classification des codes **47**
4.1 Définitions de codes particuliers 48
4.2 morphismes de codage 49
4.3 Caractérisations en termes de mots infinis 51

4.4 Caractérisations en termes de mots bi-infinis 52
4.5 Codes suffixes 54
4.6 Tableaux récapitulatifs 55

5 Etude du cas rationnel **59**
5.1 Codes rationnels et mots infinis 60
5.2 ω-codes, biω-codes et Z-codes dans le cas rationnel 63
5.3 Tableaux récapitulatifs 65
5.4 Décidabilité 66
5.5 Caractérisation des biω-codes rationnels 69

6 Codes à délai de déchiffrage fini **73**
6.1 Codes à délai fini 74
6.2 Codes à délai borné 76
6.3 Etude du cas rationnel 77

7 Codes à délai de synchronisation borné **81**
7.1 Définitions 82
7.2 Propriétés des langages engendrés 84
7.3 Etude du cas rationnel 88

8 Etude du cas fini **93**
8.1 Codes finis et codes à trois éléments 93
8.2 Codes à deux éléments 95
8.2.1 Codes et ω-codes 95
8.2.2 Biω-codes 95
8.2.3 Z-codes 98
8.2.4 Dénombrement 98
8.2.5 Résultats 99

9 Composition ; Maximalité **101**
9.1 Composition des codes 102
9.2 Maximalité 106
9.2.1 Plongement dans un code P-maximal 106
9.2.2 Plongement dans un code complet 108
9.2.3 Plongement dans un code maximal 111

10 Générateurs de langages bi-infinitaires **113**
10.1 Générateurs d'un langage bi-infinitaire 116
10.2 Générateurs minimaux 119
10.3 Bissections d'un langage bi-infinitaire 123
10.3.1 Bissections 124
10.3.2 Bissections génératives 126

10.3.3 Bissections des biadhérences 128
10.4 Existence de générateurs finis 130

Conclusion **135**

Bibliographie **137**

Abstract

Codes can be characterized by their way of acting on infinite or biinfinite words. A classification of codes based on the number of factorizations for different kinds of infinite words is set up. The obtained classes are studied and compared each other. In this classification appear particular codes which can be considered as " codes for infinite words " and other ones which can be considered as " codes for biinfinite words ". We study also properties of the infinitary languages generated by codes with bounded deciphering delay and properties of the biinfinitary languages generated by codes with bounded synchronization delay. Complementary results are presented in the rational case.

Several facts about finitary generators of biinfinitary languages are studied. For example, very thin codes are shown to be minimal generators. Finally, we prove that one can decide whether a biinfinitary rational language has generators, and whether it is finitely generated.

Résumé

La factorisation des mots infinis ou bi-infinis permet de caractériser les codes parmi les langages de mots finis. Elle permet aussi de classifier les codes selon le nombre de factorisations de certains types de mots infinis ou bi-infinis (les mots périodiques, par exemple), et de définir une notion de « code pour mots infinis » et deux notions de « code pour mots bi-infinis ». Les diverses classes obtenues sont étudiées et comparées, dans le cas général et dans le cas rationnel.

Les codes à délai de déchiffrage borné sont des « codes pour mots infinis » particuliers. De façon analogue, les codes à délai de synchronisation borné sont des « codes pour mots bi-infinis ». On étudie les propriétés des langages infinitaires engendrés par des codes à délai de déchiffrage borné et celles des langages bi-infinitaires engendrés par des codes à délai de synchronisation borné. Par exemple, on démontre que ces derniers sont des bilimites. On caractérise les codes à délai borné dans le cas rationnel.

Finalement, on étudie les générateurs des langages bi-infinitaires. On montre, en particulier, que les codes très minces sont des générateurs minimaux. On montre aussi que l'on sait décider si la famille des générateurs d'un langage bi-infinitaire rationnel donné est vide ou non, et si elle contient un langage fini.

Mots-clés: Automate, Code, Mot bi-infini, Mot infini, Langage rationnel, Code à délai de déchiffrage borné, Code à délai de synchronisation borné, Biω-générateur.

UNIVERSITE DES Sc. et Techn. BIBLIOTHEQUE UNIVERSITAIRE LILLE