Codage d'automates et théorie des cubes intersectants

Christopher Duff

THESE

présentée par

Christopher DUFF

pour obtenir le titre de **DOCTEUR**

de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE
(arrêté ministériel du 23 Novembre 1988)

Spécialité: **Micro-électronique**

CODAGE D'AUTOMATES ET THEORIE DES CUBES INTERSECTANTS

Date de soutenance: 1er mars 1991

Composition du jury :

Madame	Gabrièle SAUCIER	
Messieurs	Jacques MOSSIERE	Président
	Giovanni DE MICHELI	Rapporteur
	Alain COSTES	Rapporteur
	Robert BRAYTON	
	Raul CAMPOSANO	

Thèse préparée au sein du Laboratoire Conception de Systèmes Intégrés

GRENOBLE 1
UNIVERSITE JOSEPH FOURIER
SCIENCES. TECHNOLOGIE. MEDECINE

Président de l'Université :
M. NEMOZ Alain

Année universitaire 1988-1990

MEMBRES DU CORPS ENSEIGNANT DE SCIENCES ET DE GÉOGRAPHIE

PROFESSEURS de 1ère Classe

ADIBA Michel	Informatique
ANTOINE Pierre	Géologie I.R.I.G.M.
ARNAUD Paul	Chimie Organique
ARVIEU Robert	Physique Nucléaire I.S.N.
AUBERT Guy	Physique C.N.R.S.
AURIAULT Jean-Louis	Mécanique
AYANT Yves	Physique Approfondie
BARBIER Marie-Jeanne	Electrochimie
BARJON Robert	Physique Nucléaire I.S.N.
BARNOUD Fernand	Biochimie Macromoléculaire Végétale
BARRA Jean-René	Statistiques-Mathématiques Appliquées
BECKER Pierre	Physique
BEGUIN Claude	Chimie Organique
BELORISKY Elie	Physique
BENZAKEN Claude	Mathématiques Pures
BERARD Pierre	Mathématiques Pures
BERNARD Alain	Mathématiques Pures
BERTRANDIAS Françoise	Mathématiques Pures
BERTRANDIAS Jean-Paul	Mathématiques Pures
BILLET Jean	Géographie
BOELHER Jean-Paul	Mécanique
BRAVARD Yves	Géographie
CARLIER Georges	Biologie Végétale
CASTAING Bernard	Physique
CAUQUIS Georges	Chimie Organique
CHARDON Michel	Géographie
CHIBON Pierre	Biologie Animale
COHEN ADDAD Jean-Pierre	Physique
COLIN DE VERDIERE Yves	Mathématiques Pures
CYROT Michel	Physique du Solide
DEBELMAS Jacques	Géologie Générale
DEGRANGE Charles	Zoologie
DEMAILLY Jean-Pierre	Mathématiques Pures
DENEUVILLE Alain	Physique
DEPORTES Charles	Chimie Minérale
DOLIQUE Jean-Michel	Physique des Plasmas
DOUCE Roland	Physiologie Végétale
DUCROS Pierre	Cristallographie
FINKE Gerd	Informatique
GAGNAIRE Didier	Chimie Physique
GAUTRON René	Chimie
GENIES Eugène	Chimie
GERMAIN Jean-Pierre	Mécanique
GIDON Maurice	Géologie
GUITTON Jacques	Chimie
HICTER Pierre	Chimie
IDELMAN Simon	Physiologie Animale
JANIN Bernard	Géographie
JOLY Jean-René	Mathématiques Pures

JOSELEAU Jean-Paul	Biochimie
KAHANE André, détaché	Physique
KAHANE Josette	Physique
KRAKOWIAK Sacha	Mathématiques Appliquées
LAJZEROWICZ Jeanine	Physique
LAJZEROWICZ Joseph	Physique
LAURENT Pierre-Jean	Mathématiques Appliquées
LEBRETON Alain	Mathématiques Appliquées
DE LEIRIS Joël	Biologie
LHOMME Jean	Chimie
LLIBOUTRY Louis	Géophysique
LOISEAUX Jean-Marie	Sciences Nucléaires I.S.N.
LONGEQUEUE Nicole	Physique
LUNA Domingo	Mathématiques Pures
MACHE Régis	Physiologie Végétale
MASCLE Georges	Géologie
MAYNARD Roger	Physique du solide
OMONT Alain	Astrophysique
OZENDA Paul	Botanique (Biologie Végétale)
PANNETIER Jean	Chimie
PAYAN Jean-Jacques	Mathématiques Pures
PEBAY-PEYROULA Jean-Claude	Physique
PERRIER Guy	Géophysique
PIERRE Jean-Louis	Chimie Organique
RENARD Michel	Thermodynamique
RIEDTMANN Christine	Mathématiques
RINAUDO Marguerite	Chimie CERMAV
ROSSI André	Biologie
SAXOD Raymond	Biologie Animale
SENGEL Philippe	Biologie Animale
SERGERAERT Francis	Mathématiques Pures
SOUCHIER Bernard	Biologie
SOUTIF Michel	Physique
STUTZ Pierre	Mécanique
TRILLING Laurent	Mathématiques Appliquées
VAN CUTSEM Bernard	Mathématiques Appliquées
VIALON Pierre	Géologie

PROFESSEURS de 2ème Classe

ARMAND Gilbert	Géographie
ATTANE Pierre	Mécanique
BARET Paul	Chimie
BERTIN José	Mathématiques
BLANCHI J. Pierre	STAPS
BLOCK Marc	Biologie
BLUM Jacques	Mathématiques Appliquées
BOITET Christian	Mathématiques Appliquées
BORNAREL Jean	Physique
BORRIONE Dominique	Automatique Informatique
BOUVET Jean	Biologie
BROSSARD Jean	Mathématiques
BRUANDET Jean-François	Physique
BRUGAL Gérard	Biologie
BRUN Gilbert	Biologie
CASTAING Bernard	Physique
CERFF Rudiger	Biologie
CHIARAMELLA Yves	Mathématiques Appliquées
CHOLLET Jean-Pierre	Mécanique
COLOMBEAU Jean-François	Mathématiques (ENSL)
COURT Jean	Chimie
CUNIN Pierre-Yves	Informatique
DAVID Jean	Géographie

DHOUAILLY Danielle	Biologie
DUFRESNOY Alain	Mathématiques Pures
GASPARD François	Physique
GIDON Maurice	Géologie
GIGNOUX Claude	Sciences Nucléaires
GILLARD Roland	Mathématiques Pures
GIORNI Alain	Sciences Nucléaires
GONZALEZ SPRINBERG Gérardo	Mathématiques Pures
GUIGO Maryse	Géographie
GUMUCHAIN Hervé	Géographie
HACQUES Gérard	Mathématiques Appliquées
HERBIN Jacky	Géographie
HERAULT Jeanny	Physique
HERINO Roland	Physique
JARDON Pierre	Chimie
KERCKHOVE Claude	Géologie
MANDARON Paul	Biologie
MARTINEZ Francis	Mathématiques Appliquées
MOREL Alain	Géographie
NEMOZ Alain	Thermodynamique CNRS - CRTBT
NGUYEN HUY Xuong	Informatique
OUDET Bruno	Mathématiques Appliquées
PAUTOU Guy	Biologie
PECHER Arnaud	Géologie
PELMONT Jean	Biochimie
PELLETIER Guy	Astrophysique
PERRIN Claude	Sciences Nucléaires I.S.N.
PIBOULE Michel	Géologie
RAYNAUD Hervé	Mathématiques Appliquées
REGNARD Jean-René	Physique
RICHARD Jean-Marc	Physique
RIEDTMANN Christine	Mathématiques Pures
ROBERT Danielle	Chimie
ROBERT Gilles	Mathématiques Pures
ROBERT Jean-Bernard	Chimie Physique
SARROT-REYNAULD Jean	Géologie
SAYETAT Françoise	Physique
SERVE Denis	Chimie
STOECKEL Frédéric	Physique
SCHOLL Pierre-Claude	Mathématiques Appliquées
SUBRA Robert	Chimie
VALLADE Marcel	Physique
VIDAL Michel	Chimie Organique
VINCENT Gilbert	Physique
VIVIAN Robert	Géographie
VOTTERO Philippe	Chimie

MEMBRES DU CORPS ENSEIGNANT DE L'IUT 1

PROFESSEURS de 1ère Classe

BUISSON Roger	Physique IUT 1
CHEHIKIAN Alain	E.E.A. IUT 1
DODU Jacques	Mécanique Appliquée IUT 1
NEGRE Robert	Génie Civil IUT 1
NOUGARET Marcel	Automatique IUT 1
PERARD Jacques	E.E.A. IUT 1

PROFESSEURS de 2ème Classe

BEE Marc	Physique IUT 1
BOUTHINON Michel	E.E.A. IUT 1
CHAMBON René	Génie Mécanique IUT 1
CHENAVAS Jean	Physique IUT 1

CHILO Jean	Physique IUT 1
CHOUTEAU Gérard	Physique IUT 1
CONTE René	Physique IUT 1
FOSTER Panayotis	Chimie IUT 1
GOSSE Jean-Pierre	E.E.A. IUT 1
GROS Yves	Physique IUT 1
HAMAR Roger	Chimie IUT 1
KUHN Gérard, (détaché)	Physique IUT 1
LEVIEL Jean-Louis	Physique IUT 1
MAZUER Jean	Physique IUT 1
MICHOULIER Jean	Physique IUT 1
MONLLOR Christian	E.E.A. IUT 1
PERRAUD Robert	Chimie IUT 1
PIERRE Gérard	Chimie IUT 1
TERRIEZ Jean-Michel	Génie Mécanique IUT 1
TOUZAIN Philippe	Chimie IUT 1
TURGEMAN Sylvain	Génie Civil
VINCENDON Marc	Chimie IUT 1
ZIGONE Michel	Physique IUT 1

PROFESSEURS DE PHARMACIE

AGNIUS-DELORD Claudine	Physique	Faculté La Tronche
ALARY Josette	Chimie Analytique	Faculté La Tronche
BERIEL Hélène	Physiologie et Pharmacologie	Faculté La Tronche
CUSSAC Max	Chimie Thérapeutique	Faculté La Tronche
DEMENGE Pierre	Pharmacodynamie	Faculté La Tronche
FAVIER Alain	Biochimie	C.H.R.G.
JEANNIN Charles	Pharmacie Galénique	Faculté Meylan
LATURAZE Jean	Biochimie	Faculté La Tronche
LUU DUC Cuong	Chimie Générale	Faculté La Tronche
MARIOTTE Anne-Marie	Pharmacognosie	Faculté La Tronche
MARZIN Daniel	Toxicologie	Faculté Meylan
RENAUDET Jacqueline	Bactériologie	Faculté La Tronche
ROCHAT Jacques	Hygiène et Hydrologie	Faculté La Tronche
SEIGLE-MURANDI Françoise	Botanique et Cryptogamie	Faculté Meylan
VERAIN Alice	Pharmacie Galénique	Faculté Meylan

MEMBRES DU CORPS ENSEIGNANT DE MEDECINE

PROFESSEURS Classe Exceptionnelle et 1ère Classe

AMBLARD Pierre	Dermatologie	C.H.R.G.
AMBROISE-THOMAS Pierre	Parasitologie	C.H.R.G.
BEAUDOING André	Pédiatrie-Puériculture	C.H.R.G.
BEREZ Henri	Orthopédie-Traumatologie	Hôpital Sud
BONNET Jean-Louis	Ophtalmologie	C.H.R.G.
BOUCHET Yves	Anatomie	Faculté La Merci
	Chirurgie Générale et Digestive	C.H.R.G.
BUTEL Jean	Orthopédie-Traumatologie	C.H.R.G.
CHAMBAZ Edmond	Biochimie	C.H.R.G.
CHAMPETIER Jean	Anatomie Topographique et Appliquée	C.H.R.G.
CHARACHON Robert	O.R.L.	C.H.R.G.
COLOMB Maurice	Immunologie	Hôpital Sud
COUDERC Pierre	Anatomie Pathologique	C.H.R.G.
DELORMAS Pierre	Pneumophtisiologie	C.H.R.G.
DENIS Bernard	Cardiologie	C.H.R.G.
GAVEND Michel	Pharmacologie	Faculté La Merci
HOLLARD Daniel	Hématologie	C.H.R.G.
LATREILLE René	Chirurgie Thoracique et Cardiovasculaire	C.H.R.G.
LE NOC Pierre	Bactériologie-Virologie	C.H.R.G.
MALINAS Yves	Gynécologie et Obstétrique	C.H.R.G.
MALLION Jean-Michel	Médecine du Travail	C.H.R.G.

MICOUD Max	Clinique Médicale et Maladies Infectieuses	C.H.R.G.
MOURIQUAND Claude	Histologie	Faculté La Merci
PARAMELLE Bernard	Pneumologie	C.H.R.G.
PERRET Jean	Neurologie	C.H.R.G.
RACHAIL Michel	Hépato-Gastro-Entérologie	C.H.R.G.
DE ROUGEMONT Jacques	Neurochirurgie	C.H.R.G.
SARRAZIN Roger	Clinique Chirurgicale	C.H.R.G.
STIEGLITZ Paul	Anesthésiologie	C.H.R.G.
TANCHE Maurice	Physiologie	Faculté La Merci
VIGNAIS Pierre	Biochimie	Faculté La Merci

PROFESSEURS 2ème Classe

BACHELOT Yvan	Endocrinologie	C.H.R.G.
BARGE Michel	Neurochirurgie	C.H.R.G.
BENABID Alim-Louis	Biophysique	Faculté La Merci
BENSA Jean-Claude	Immunologie	Hôpital Sud
BERNARD Pierre	Gynécologie Obstétrique	C.H.R.G.
BESSARD Germain	Pharmacologie	Abidjan
BOLLA Michel	Radiothérapie	C.H.R.G.
BOST Michel	Pédiatrie	C.H.R.G.
BOUCHARLAT Jacques	Psychiatrie Adultes	Hôpital Sud
BRAMBILLA Christian	Pneumologie	C.H.R.G.
CHIROSSEL Jean-Paul	Anatomie-Neurochirurgie	C.H.R.G.
COMET Michel	Biophysique	Faculté La Merci
CONTAMIN Charles	Chirurgie Thoracique et Cardiovasculaire	C.H.R.G.
CORDONNIER Daniel	Néphrologie	C.H.R.G.
COULOMB Max	Radiologie	C.H.R.G.
CROUZET Guy	Radiologie	C.H.R.G.
DEBRU Jean-Luc	Médecine Interne et Toxicologie	C.H.R.G.
DEMONGEOT Jacques	Biostatistiques et informatique médicale	Faculté La Merci
DUPRE Alain	Chirurgie Générale	C.H.R.G.
DYON Jean-François	Chirurgie Infantile	C.H.R.G.
ETERRADOSSI Jacqueline	Physiologie	Faculté La Merci
FAURE Claude	Anatomie et Organogénèse	C.H.R.G.
FAURE Gilbert	Urologie	C.H.R.G.
FOURNET Jacques	Hépato-Gastro-Entérologie	C.H.R.G.
FRANCO Alain	Médecine Interne	C.H.R.G.
GIRARDET Pierre	Anesthésiologie	C.H.R.G.
GUIDICELLI Henri	Chirurgie Générale et Vasculaire	C.H.R.G.
GUIGNIER Michel	Thérapeutique et Réanimation Médicale	C.H.R.G.
HADJIAN Arthur	Biochimie	Faculté La Merci
HALIMI Serge	Endocrinologie et Maladies Métaboliques	C.H.R.G.
HOSTEIN Jean	Hépato-Gastro-Entérologie	C.H.R.G.
HUGONOT Robert	Médecine Interne	C.H.R.G.
JALBERT Pierre	Histologie Cytogénétique	C.H.R.G.
JUNIEN-LAVILLAUROY Claude	O.R.L.	C.H.R.G.
KOLODIE Lucien	Hématologie Biologique	C.H.R.G.
LETOUBLON Christian	Chirurgie Générale	C.H.R.G.
MACHECOURT Jacques	Cardiologie et Maladies Vasculaires	C.H.R.G.
MAGNIN Robert	Hygiène	C.H.R.G.
MASSOT Christian	Médecine Interne	C.H.R.G.
MOUILLON Michel	Ophtalmologie	C.H.R.G.
PELLAT Jacques	Neurologie	C.H.R.G.
PHELIP Xavier	Rhumatologie	C.H.R.G.
RACINET Claude	Gynécologie Obstétrique	Hôpital Sud
RAMBAUD Pierre	Pédiatrie	C.H.R.G.
RAPHAEL Bernard	Stomatologie	C.H.R.G.
SCHAERER René	Cancérologie	C.H.R.G.
SEIGNEURIN Jean-Marie	Bactériologie-Virologie	Faculté La Merci
SELE Bernard	Cytogénétique	Faculté La Merci
SOTTO Jean-Jacques	Hématologie	C.H.R.G.
STOEBNER Pierre	Anatomie Pathologique	C.H.R.G.
VROUSOS Constantin	Radiothérapie	C.H.R.G.

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

46 avenue Felix Viallet
38031 GRENOBLE cedex

Tél. : 76.57.45.00

Année universitaire 1989

Président de l'Institut :
Monsieur Georges LESPINARD

Professeurs des Universités

BARIBAUD Michel	ENSERG
BARRAUD Alain	ENSIEG
BAUDELET Bernard	ENSPG
BEAUFILS Jean-Pierre	INPG
BLIMAN Samuel	ENSERG
BOIS Philippe	ENSHMG
BONNETAIN Lucien	ENSEEG
BONNET Guy	ENSPG
BRISSONNEAU Pierre	ENSIEG
BRUNET Yves	IUFA
CAILLERIE Denis	ENSHMG
CAVAIGNAC Jean-François	ENSPG
CHARTIER Germain	ENSPG
CHENEVIER Pierre	ENSERG
CHERADAME Hervé	UFR PGP
CHERUY Arlette	ENSIEG
CHOVET Alain	ENSERG
COHEN Joseph	ENSERG
COLINET Catherine	ENSEEG
CORNUT Bruno	ENSIEG
COULOMB Jean-Louis	ENSIEG
COUMES André	ENSERG
CROWLEY James	ENSIMAG
DARVE Félix	ENSHMG
DELLA-DORA Jean	ENSIMAG
DEPEY Maurice	ENSERG
DEPORTES Jacques	ENSPG
DEROO Daniel	ENSEEG
DESRE Pierre	ENSEEG
DOLMAZON Jean-Marc	ENSERG
DURAND Francis	ENSEEG
DURAND Jean-Louis	ENSPG
FAUTRELLE Yves	ENSHMG
FOGGIA Albert	ENSIEG
FONLUPTJean	ENSIMAG
FOULARD Claude	ENSIEG
GANDINI Alessandro	UFR PGP
GAUBERT Claude	ENSPG
GENTIL Pierre	ENSERG
GENTIL Sylviane	ENSIEG
GREVEN Hélène	IUFA
GUEGUEN Claude	ENSIEG
GUERIN Bernard	ENSERG
GUYOT Pierre	ENSEEG
IVANES Marcel	ENSIEG
JAUSSAUD Pierre	ENSIEG
JOST Rémy	ENSPG
JOUBERT Jean-Claude	ENSPG
JOURDAIN Geneviève	ENSIEG
LACOUME Jean-Louis	ENSIEG
LADET Pierre	ENSIEG
LESIEUR Marcel	ENSHMG
LESPINARD Georges	ENSHMG
LONGEQUEUE Jean-Pierre	ENSPG
LORET Benjamin	ENSHMG
LOUCHET François	ENSEEG
LUCAZEAU Guy	ENSEEG
MASSE Philippe	ENSIEG
MASSELOT Christian	ENSIEG
MAZARE Guy	ENSIMAG
MOHR Roger	ENSIMAG
MOREAU René	ENSHMG
MORET Roger	ENSIEG
MOSSIERE Jacques	ENSIMAG
OBLED Charles	ENSHMG
OZIL Patrick	ENSEEG
PA ULEAU Yves	ENSEEG
PERRET Robert	ENSIEG
PIAU Jean-Michel	ENSHMG
PIC Etienne	ENSERG
PLATEAU Brigitte	ENSIMAG
POUPOT Christian	ENSERG
RAMEAU Jean-Jacques	ENSEEG
REINISCH Raymond	ENSPG
RENAUD Maurice	UFR PGP
ROBERT André	UFR PGP
ROBERT François	ENSIMAG
SABONNADIERE Jean-Claude	ENSIEG
SAUCIER Gabrièle	ENSIMAG
SCHLENKER Claire	ENSPG
SCHLENKER Michel	ENSPG
SERMET Pierre	ENSERG
SILVY Jacques	UFR PGP
SIRIEYS Pierre	ENSHMG
SOHM Jean-Claude	ENSEEG
SOLER Jean-Louis	ENSIMAG
SOUQUET Jean-Louis	ENSEEG
TROMPETTE Philippe	ENSHMG
VINCENT Henri	ENSPG
ZADWORNY François	ENSERG

Personnes ayant obtenu le diplôme d'HABILITATION A DIRIGER DES RECHERCHES

BECKER Monique
BINDER Zdenek
CHASSERY Jean-Marc
CHOLLET Jean-Pierre
COEY John
COLINET Catherine
COMMAULT Christian
CORNUEJOLS Gérard
COULOMB Jean- Louis
COURNIL M.
DALARD Francis
DANES Florin
DEROO Daniel
DIARD Jean-Paul
DION Jean-Michel
DUGARD Luc
DURAND Madeleine
DURAND Robert
GALERIE Alain
GAUTHIER Jean-Paul
GENTIL Sylviane
GHIBAUDO Gérard
HAMAR Sylvaine
HAMAR Roger
LACHENAL D.
LADET Pierre
LATOMBE Claudine
LE HUY H.
LE GORREC Bernard
MADAR Roland
MEUNIER G.
MULLER Jean
NGUYEN TRONG Bernadette
NIEZ J.J.
PASTUREL Alain
PLA Fernand
ROGNON J.P.
ROUGER Jean
TCHUENTE Maurice
VINCENT Henri
YAVARI A.R.

Chercheurs du C.N.R.S

DIRECTEURS DE RECHERCHE CLASSE 0

LANDEAU	Ioan
NAYROLLES	Bernard

Directeurs de recherche 1ère Classe

ANSARA Ibrahim
CARRE René
FRUCHART Robert
HOPFINGER Emile
JORRAND Philippe
KRAKOWIAK Sacha
LEPROVOST Christian
VACHAUD Georges
VERJUS Jean-Pierre

Directeurs de recherche 2ème Classe

ALEMANY Antoine
ALLIBERT Colette
ALLIBERT Michel
ARMAND Michel
AUDIER Marc
BERNARD Claude
BINDER Gilbert
BONNET Roland
BORNARD Guy
CAILLET Marcel
CALMET Jacques
CHATILLON Chritiant
CLERMONT Jean-Robert
COURTOIS Bernard
DAVID René
DION Jean-Michel
DRIOLE Jean
DURAND Robert
ESCUDIER Pierre
EUSTATHOPOULOS Nicolas
GARNIER Marcel
GUELIN Pierre
JOUD Jean-Charles
KAMARINOS Georges
KLEITZ Michel
KOFMAN Walter
LEJEUNE Gérard
MADAR Roland
MERMET Jean
MICHEL Jean-Marie
MEUNIER Jacques
PEUZIN Jean-Claude
PIAU Monique
RENOUARD Dominique
SENATEUR Jean-Pierre
SIFAKIS Joseph
SIMON Jean-Paul
SUERY Michel
TEODOSIU Christian
VAUCLIN Michel
VENNEREAU Pierre
WACK Bernard
YONNET Jean-Paul

Personnalités agrées à titre permanent à diriger des travaux de recherche
(décision du conseil scientifique)

E.N.S.E.E.G
HAMMOU Abdelkader
MARTIN-GARIN Régina
SARRAZIN Pierre
SIMON Jean-Paul

E.N.S.E.R.G
BOREL Joseph

E.N.S.I.E.G
DESCHIZEAUX Pierre
GLANGEAUD François
PERARD Jacques
REINISCH Raymond

E.N.S.H.M.G
ROWE Alain

E.N.S.I.M.A.G
COURTIN Jacques

C.E.N.G
CADET Jean
COEURE Philippe
DELHAYE Jean-Marc
DUPUY Michel
JOUVE Hubert
NICOLAU Yvan
NIFENECKER Hervé
PERROUD Paul
PEUZIN Jean-Claude
TAIEB Maurice
VINCENDON Marc

Laboratoires extérieurs :

C.N.E.T
DEVINE Rodericq
GERBER Roland
MERCKEL Gérard
PAULEAU Yves

Situation particulière

PROFESSEURS D'UNIVERSITE

DETACHEMENT

ENSIMAG	LATOMBE	J..Claude	Détachement	21/10/1989
ENSHMG	PIERRARD	J.Marie	Détachement	30/04/1989
ENSIMAG	VEILLON	Gérard	Détachement	30/09/1990
ENSIMAG	VERJUS	J.Pierre	Détachement	30/09/1989
ENSPG	BLOCH	Daniel	Recteur à c/	21/12/1988

SURNOMBRE

INPG	CHIAVERINA	Jean	30/09/1989
ENSHMG	BOUVARD	Maurice	30/09/1991
ENSEEG	PARIAUD	J.Charles	30/09/1991

Avant-Propos

Je tiens à remercier tout d'abord Madame Gabrièle SAUCIER, Professeur à l'ENSIMAG et directrice du Laboratoire Conception de Systèmes Intégrés, pour m'avoir accueilli dans son laboratoire et pour avoir guidé mes recherches. L'aboutissement de ce travail a été grandement possible grâce à ses nombreuses remarques et suggestions et aux innombrables discussions sur le sujet.

Je remercie également:

Monsieur Jacques MOSSIERE, Professeur et directeur de l'ENSIMAG, de me faire l'honneur de présider ce jury,

Monsieur Giovanni DE MICHELI, Professeur à l'Université de Stanford, d'avoir accepté d'être rapporteur de cette thèse,

Monsieur Alain COSTES, Professeur et directeur du LAAS, d'avoir accepté d'être rapporteur de cette thèse,

Monsieur Robert BRAYTON, Professeur à l'Université de Berkeley, d'avoir accepté de faire partie de mon jury,

et Monsieur Raul CAMPOSANO, Docteur de IBM Yorktown Heights, d'avoir accepté de faire partie de mon jury.

Je remercie également tous mes collègues du CSI pour l'agréable ambiance de travail, toute l'équipe de synthèse logique pour son aide, plus particulièrement Pierre ABOUZEID et Jérôme BLANC pour l'aide qu'ils m'ont apportée pour la phase d'évaluation. Je voudrais aussi remercier Xavier DELORD qui nous permet de travailler dans de très bonnes conditions par la bonne gestion de notre outil de travail, et Régis LEVEUGLE pour ses nombreux conseils, suggestions et explications au cours de discussions tardives. Je pense aussi à mon grand "yaya" (frère en lingala) Edmond KOUKA.

Je ne voudrais pas oublier Pascale RENAUD et Martine DELMAS pour leur agréable accueil à chaque visite au secrétariat et pour leur aide. Mon bon souvenir va aussi à Marie-Jo SCHNEIDER.

Finalement, et non des moindres, je voudrais remercier Raul VELAZCO pour son encouragement lors de ma scolarité à l'ENSIMAG, et pour m'avoir mis en contact avec Madame SAUCIER.

A mes parents et à Grannie
pour m'avoir permis d'en arriver là

A ma soeur Jennyfer pour son encouragement, sa gentillesse et son soutien

A Anne pour sa patience et son amour, et à Annaëlle pour ses grands sourires

Toto avait l'habitude de se lever tard. Pour le sermonner, son père lui raconta l'histoire d'un petit garçon qui s'était levé tôt et avait trouvé un billet de cinq cents francs dans la rue. "Tu vois," lui dit son père, "s'il était arrivé un peu plus tard, quelqu'un d'autre aurait trouvé le billet avant lui". Toto acquiesça, puis, après avoir réfléchi un instant, dit triomphalement: "D'accord, Papa, mais celui qui a perdu le billet, il a dû se lever encore plus tôt ?!!!".

Résumé

Cette thèse propose une méthode de codage optimisé d'automate synchrone dans un environnement de type compilateur de silicium. Dans une première partie, on recherche sur le graphe d'états des situations prédictives de minimisation des équations des variables internes et des sorties. Ceci définit des contraintes sur le codage en terme d'une liste de "groupes d'adjacence" d'états à immerger sur des faces ou cubes de l'hypercube. Dans une deuxième partie, le codage est réalisé en satisfaisant au mieux ces affectations de faces et leurs intersections.

Les principes de base de cette approche sont les suivants:

(i) pour la première phase, la recherche de situations prédictives de minimisation est fondée sur la théorie des "paires de partition" de Hartmanis. Les situations sont recherchées entrée par entrée; cette approche locale permettra de faire face aux grandes complexités. Remarquons que le codage des entrées n'est pas abordé. La priorité est donnée aux fusions potentielles de monômes dans les équations. On ne recherchera pas de façon indifférenciée comme dans d'autres approches (Mustang, par exemple) toutes les minimisations possibles incluant les factorisations. En effet, il est raisonnablement estimé que seule la fusion de monômes assure un gain en surface et en connectique.

(ii) pour la deuxième phase, les techniques d'immersion dans l'hypercube seront très sophistiquées. Elles reprendront une représentation de l'hypercube par le treillis de l'ensemble des parties d'un ensemble à N éléments. Pour résoudre les problèmes des "contraintes intersectantes", c'est-à-dire des contraintes impliquant des sous-ensembles d'états en commun, une théorie dite théorie des cubes intersectants sera proposée.

Les résultats de cette thèse ont donné lieu à un logiciel intégré dans le système ASYL. Les résultats obtenus en gain de surface sur silicium, de chemin critique et de facteur de routage sont parmi les meilleurs actuellement connus.

Mots-clés

codage d'automates synchrones, groupes d'adjacence, immersion, hypercube, cube intersectant

Abstract

This thesis proposes a state assignment method for finite state machines based on a sophisticated and powerful embedding theory called intersecting cube theory.

In a first part, predictive minimization situations are looked for in the control flowgraph. The research is based on the partition pair theory of Stearns and Hartmanis aiming at reduced dependency of the output and next state equations towards inputs and present state variables. This leads to the definition of encoding constraints in terms of adjacency groups defined as sets of states to be put bijectively on faces or cubes of the hypercube. These sets of states are generally intersecting ones. Therefore, the second part proposes a solution of the generalized intersecting face or cube embedding problem. The hypercube is viewn as the lattice of the parts of a set of N elements, and the intersecting cube embedding problem is processed within the frame of the corresponding theory. This approach has been implemented by a software in the ASYL system, experimented on the official as well as on industrial examples. The gain in silicon area, critical path, and routing factor with respect to random assignment, appears to be one of the best presently known.

Keywords

state assignment, synchronous finite state machine, adjacency groups, embedding, hypercube, intersecting cube

TABLE DES MATIERES

INTRODUCTION

Le codage des états d'un automate d'états finis a été largement étudié dans le passé. L'objectif était d'une part d'obtenir des équations des variables internes et de sortie simples, et d'autre part d'étudier les aléas ou courses critiques dans les systèmes asynchrones.

En ce qui concerne le premier objectif, une des approches les plus fructueuses était fondée sur la théorie des "couples de partitions" exposée dans [HAR66]. A partir de l'analyse du tableau des états, cette théorie établit une relation directe entre le codage des états et la simplicité des équations des variables internes et des sorties, ceci en décomposant le problème pour chaque valeur d'entrée. En ce qui concerne le deuxième objectif, l'approche utilisée dans [SAU87] consiste à identifier des ensembles de groupes d'états devant être immergés dans des faces ou cubes disjoints de l'hypercube. Cette approche a permis de dégager un certain nombre de techniques de représentation de l'hypercube et d'immersion de groupes d'états. Comme résultat direct, elle a permis d'améliorer les résultats d'Huffman [HUF54] sur le codage universel des automates asynchrones. Notons qu'aucune des ces deux approches ne visait vraiment le traitement des problèmes de grande complexité. Elle se préoccupait essentiellement de dégager les notions de base sous-jacentes à ces problèmes.

Au cours des 5 dernières années, un regain d'intérêt très vif s'est manifesté sur ce problème de codage d'automates d'états finis. En effet, avec l'avènement des compilateurs de silicium, les outils de synthèse se sont vus propulsés au premier rang des enjeux stratégiques des outils de conception assistée par ordinateur de VLSI. Dans ces outils le but d'un codage optimisé de contrôleur est principalement la réduction de complexité des équations des variables internes et des sorties mais les méthodes doivent cette fois faire face à de très grandes complexités (automate à plus de 100 états, équations logiques à plus de 1000 monômes, etc.) et donner lieu à des logiciels performants. Les méthodes seront donc des méthodes heuristiques dont l'efficacité sera testée sur des cas types ou benchmarks internationaux.

Les heuristiques proposées procèdent toutes en 2 temps. Une première phase analysera la structure de l'automate (tableau ou graphe des états) et en déduira des "situations" de minimisation potentielle des équations de sortie ou des variables internes. Ces situations donneront lieu à des contraintes sur le codage des états impliqués dans ces situations. Si ces contraintes sont respectées, on pourra démontrer qu'il y a gain de complexité en terme de nombres de

monômes ou de littéraux dans des équations. Dans un deuxième temps, des techniques de codage ou d'immersion dans l'hypercube chercheront à respecter un nombre maximal de ces contraintes.

Il est à noter que le concept de simplification des équations des variables internes et des sorties doit être lié à une simplification réelle dans la réalisation physique (circuit plus petit et plus rapide). Cette réalisation physique concerne plusieurs cibles technologiques, les deux cibles les plus classiques étant la logique 2 couches (NOR/NOR) du type réseaux programmables (PLA) et la logique multicouche réalisée sur cellules standards de bibliothèque précaractérisée. Les méthodes de prédiction de minimisation vont dépendre des cibles car les techniques de minimisation en dépendent elles-mêmes.

La première approche, dans cette nouvelle série de travaux, est sans aucun doute, celle concernant les réseaux programmables (PLA) dans [DEM83], [DEM84]. Elle s'appuie fortement sur les progrès en minimisation 2-couches, appelée minimisation globale, obtenue quelques années auparavant pour cette même cible ([BRA85] [SIC88]).

L'approche de cette thèse vise en priorité les réalisations multicouche sur cellules standards tout restant applicable pour la cible 2-couches. Les principes de base de notre approche sont les suivants :

(i) pour la première phase, la recherche de situations prédictives de minimisation est fondée sur la théorie des paires de "partition" de Hartmanis [HAR66]. Les situations sont recherchées entrée par entrée; cette approche locale permettra de faire face aux grandes complexités; remarquons que le codage des entrées n'est pas abordé. La priorité est donnée aux fusions potentielles de monômes dans les équations. On ne recherchera pas de façon indifférenciée comme dans d'autres approches (Mustang) toutes les minimisations possibles incluant les factorisations. En effet, il est raisonnablement estimé que seule la fusion de monômes assure un gain en surface et en connectique.

(ii) pour la deuxième phase, les techniques d'immersion dans l'hypercube seront très sophistiquées. Elles reprendront une représentation de l'hypercube par le treillis de l'ensemble des parties d'un ensemble à N éléments. Pour résoudre les problèmes des "contraintes intersectantes", c'est-à-dire des

contraintes impliquant des sous-ensembles d'états en commun, une théorie dite théorie des cubes intersectants sera proposée ([SAU89abc], [SAU90ab]).

Le plan de cette thèse sera donc le suivant. Après une brève présentation du problème, les techniques de recherche de situations sont présentées. Ces situations sont d'abord les situations de fusion potentielle des monômes (situations privilégiées comme nous l'avons dit), puis les situations à factorisation des équations qui sont prises en compte seulement si le codage de certains états est encore libre. Dans un chapitre suivant, la théorie des cubes intersectants est présentée. Un autre chapitre donnera alors l'algorithme de codage proposé et implanté. Le chapitre suivant donnera les détails sur les résultats pratiques obtenus sur les "benchmarks" de référence, sur plusieurs sites industriels et sur plusieurs cibles technologiques. Un dernier chapitre commentera les similitudes et différences par rapport aux autres approches.

CHAPITRE I

DÉFINITIONS PRÉALABLES ET SITUATION DU PROBLEME

1.1. AUTOMATE D'ÉTATS FINI ET CODAGE

Un automate d'états fini est défini par un quintuplet (I, O, S, λ, δ) où

I est un ensemble fini de variables d'entrée $\{i_1, ..., i_{|I|}\}$

O est un ensemble fini de variables de sortie $\{o_1, ..., o_{|O|}\}$

S est un ensemble fini d'états $\{s_1, ..., s_{|S|}\}$

δ est une application de $I \times S \rightarrow S$, appelée fonction d'état suivant

λ est l'une des deux applications suivantes:

$\lambda : I \times S \rightarrow O$ dans le cas d'un automate de Mealy

$\lambda : S \rightarrow O$ dans le cas d'un automate de Moore

λ est la fonction de sortie.

A un instant donné, l'automate se trouve dans un état s, appelé état courant. Il évoluera en fonction des entrées vers un état s', appelé état suivant. Une transition est définie par un triplet (s, s', i) vérifiant $s' = \delta(s, i)$. Dans la suite, on ne considèrera que les automates synchrones. Le temps est représenté par des instants discrets, et on peut également écrire la fonction d'état suivant et la fonction de sortie de la manière suivante:

$= \delta(s(t), i(t))$

$= \lambda(s(t))$ pour un automate de Moore

ou

$o(t+1) = \delta(s(t), i(t))$ pour un automate de Mealy

1.2. REPRÉSENTATIONS CLASSIQUES

Les trois façons les plus classiques de représenter un automate d'états finis, sont la représentation par un graphe de transitions, la représentation par un tableau d'états et la représentation par une liste normalisée de transitions qui est un format standard d'échange de benchmarks.

1.2.1. Graphe de Transitions

1.2.1.1. Représentation générale

L'automate est représenté par un graphe G(V, T) où

V est l'ensemble des noeuds du graphe associé bijectivement à l'ensemble des états,

T est l'ensemble des arcs associés bijectivement aux transitions.

Dans le cas d'un automate de Moore, chaque état s est étiqueté par un vecteur de sortie $\lambda(s)$ et chaque transition est étiquetée par un vecteur d'entrée i. Dans le cas d'un automate de Mealy, seules les transitions sont étiquetées et ce, par un couple constitué d'un vecteur d'entrée i et d'un vecteur de sortie $\lambda(s, i)$.

Chaque composante de i et de $\lambda(s)$ peut prendre ses valeurs dans {0, 1, -}, la valeur "-" représentant indifféremment 0 ou 1.

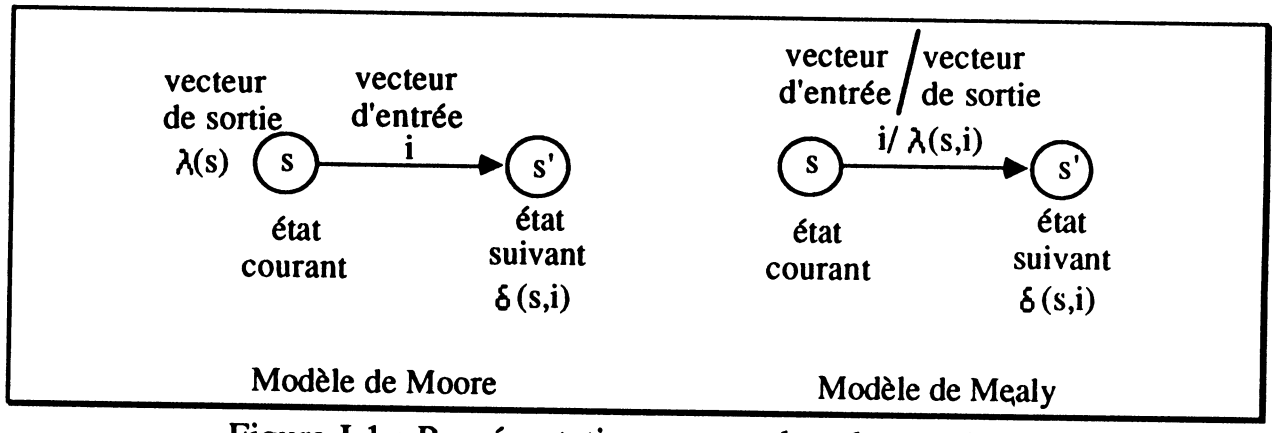

Figure I-1 : Représentation par graphes de transition

1.2.1.2. Représentation compacte

Cette représentation est une représentation condensée de la précédente. Pour les vecteurs d'entrée, on ne spécifie que les composantes égales à 0 et à 1. Les monômes associés aux valeurs d'entrée sont appelés prédicats ou conditions sur les entrées. Pour les vecteurs de sortie, seules les composantes égales à 1 dans un état ou sur une transition seront indiquées. La figure I-2 donne un exemple d'automate dans une telle représentation.

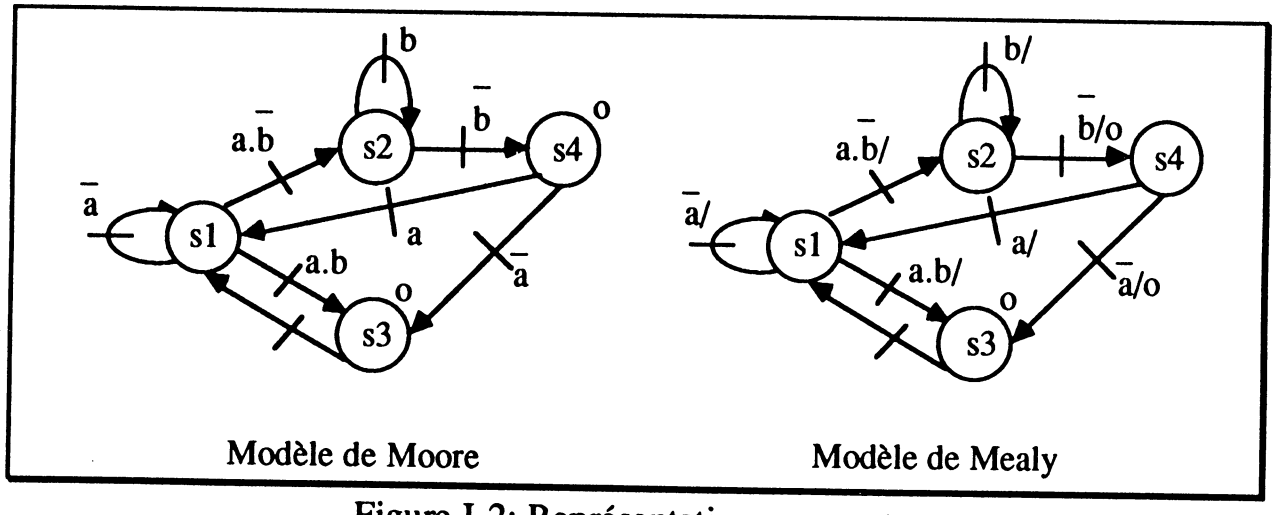

Figure I-2: Représentation compacte

1.2.2. Tableau d'états

C'est un tableau dans lequel une ligne est associée à un état et une colonne est associée à un vecteur d'entrée. Chaque case indiquera l'état suivant, et la sortie émise dans le cas d'un automate de Mealy. Pour un automate de Moore, une colonne supplémentaire indiquera le vecteur de sortie à émettre en fonction de l'état courant. Le tableau d'états correspondant à l'automate de la figure I-2 est donnée dans la figure suivante.

état courant \ ab	00	01	10	11	o
s1	s1	s1	s2	s3	0
s2	s4	s2	s4	s2	0
s3	s1	s1	s1	s1	1
s4	s3	s3	s1	s1	1

Modèle de Moore

état courant / o \ ab	00	01	10	11
s1	s1/0	s1/0	s2/0	s3/1
s2	s4/1	s2/0	s4/1	s2/0
s3	s1/0	s1/0	s1/0	s1/0
s4	s3/1	s3/1	s1/0	s1/0

Modèle de Mealy

Figure I-3: Représentation par tableau d'états

1.2.3. Liste normalisée de transitions

L'automate est représenté par un tableau dont chaque ligne représente une transition et la sortie émise. C'est un tableau mixte, indiquant les valeurs dans {0, 1, -} pour les variables de I et O, et des symboles pour l'ensemble S pour chaque transition. Chaque ligne possède ($|I| + |O| + 2$) colonnes décrivant dans l'ordre une configuration d'entrée, l'état courant, l'état suivant et la configuration de sortie correspondante. Pour un automate de Mealy, la sortie est associée à la transition alors que pour un automate de Moore, la sortie est associée à l'état courant; dans ce dernier cas, la sortie est donc identique pour toutes les transitions ayant le même état origine.

entrée (ab)	état courant	état suivant	sortie
0-	s1	s1	0
10	s1	s2	0
11	s1	s3	0
-1	s2	s2	0
-0	s2	s4	0
--	s3	s1	1
1-	s4	s1	1
0-	s4	s3	1

Modèle de Moore

entrée (ab)	état courant	état suivant	sortie
0-	s1	s1	0
10	s1	s2	0
11	s1	s3	1
-1	s2	s2	0
-0	s2	s4	1
--	s3	s1	0
1-	s4	s1	0
0-	s4	s3	1

Modèle de Mealy

Figure I-4: Représentation par liste de transitions

1.3. RELATIONS DE SUCCESSION

1.3.1. Relation de succession étendue à un sous-ensemble d'états

Soit $B = \{s_1, \ldots, s_k\}$ un sous-ensemble de l'ensemble des états S.

Pour une entrée i donnée, on définit $\delta(B, i) = \{\delta(s_1, i), \ldots, \delta(s_k, i)\}$ comme l'ensemble des états successeurs des états de l'ensemble B pour cette entrée.

1.3.2. Relation de succession étendue à une partition de l'ensemble des états

Soit Π une partition de l'ensemble des états avec $\Pi = \{B_1, \ldots, B_m\}$. Pour une entrée i donnée, on définit $\delta(\Pi, i) = \{\delta(B_1, i), \ldots, \delta(B_m, i)\}$ comme l'ensemble des blocs successeurs des blocs de Π.

1.3.3. Relation de succession étendue à un sous-ensemble des prédicats d'entrée

Soit P un sous-ensemble des prédicats d'entrée avec $P = \{p_1, \ldots, p_k\}$.

Pour un état s donné, on définit $\delta(s, P) = \{\delta(s, p_1), \ldots, \delta(s, p_k)\}$ comme l'ensemble des états successeurs de l'état s pour l'ensemble P de prédicats d'entrée.

1.3.4. Relation de succession étendue à une partition de l'ensemble des prédicats d'entrée

Soit $\Pi(i)$ une partition de l'ensemble des prédicats d'entrée avec $\Pi(i) = \{P_1, \ldots, P_m\}$. Pour un état s donné, on définit $\delta(s, \Pi(i))$ par $\delta(s, \Pi(i)) = \{\delta(s, P_1), \ldots, \delta(s, P_m)\}$, l'ensemble des blocs successeurs de l'état s pour les blocs de $\Pi(i)$.

1.4. DÉFINITION DU CODAGE

Coder un automate consiste à associer injectivement à chaque état une valeur d'une variable booléenne générale Y à N composantes appelées variables internes. Ceci implique que $2^N \geq |S|$, ou encore $N \geq \log_2(|S|)$.

La valeur de Y associée à un état est appelée code de cet état.

Les états de l'automate étant codés, les applications δ et λ se traduisent par les applications suivantes:

$$\times\ I(t) \xrightarrow{\delta} Y(t+1)$$

$$Y(t) \xrightarrow{\lambda} O(t+1)$$

ou

$$Y(t) \times I(t) \xrightarrow{\lambda} O(t+1).$$

Les équations correspondantes définies par $Y(t+1) = f(Y(t), I(t))$ et par l'une des deux fonctions $O(t) = z(Y(t))$ ou $O(t+1) = z(Y(t), I(t))$ sont appelées équations des variables internes et des sorties. Pour alléger l'écriture, on écrira Y pour Y(t+1) et y pour Y(t).

Différents types de codage sont connus; les deux plus classiques sont le codage "1 parmi N" et le codage compact.

1.4.1. Codage dit "1 parmi N"

C'est un codage où une seule des composantes de Y vaut 1, ceci pour tout Y.

1.4.2. Codage compact

C'est un codage où N atteint sa plus petite valeur, c'est-à-dire, le plus petit entier supérieur ou égal à $\log_2(|S|)$.

1.5. CODAGE D'UN ENSEMBLE DE 2^K ÉTATS SUIVANT K BIPARTITIONS

Soient $\{\Pi_1, ..., \Pi_k\}$ k bipartitions de l'ensemble des états avec $\Pi_j = (\Pi_j^0, \Pi_j^1)$. On appelle codage d'un ensemble de 2^k états suivant k bipartitions le codage consistant à affecter la valeur 0 (resp. 1) à la $j^{ième}$ variable interne Y_j du code de tout sommet appartenant à Π_j^0 (resp. Π_j^1), avec la condition que les codes de tous ces états sont distincts.

exemple

Considérons l'ensemble des états {s1, s2, s3, s4}; soient {(s1, s2), (s3, s4)} et {(s1, s3), (s2, s4)} deux bipartitions de cet ensemble. Un codage de ces états suivant les deux partitions est l'ensemble {10, 11, 00, 01} associé bijectivement à {s1, s2, s3, s4}.

L'objet de cette thèse est l'étude de l'impact du codage sur la simplification de ces équations. La variable booléenne Y est une variable booléenne générale; chacune de ses composantes Y_j sera appelée variable interne.

1.6. PROBLEME DU CODAGE

1.6.1. Définitions préalables

1.6.1.1. Noeud actif pour une variable interne

C'est un noeud correspondant à un état dont le code contient un 1 pour cette variable interne.

Le noeud associé à l'état s est actif pour la variable interne Y_k si et seulement si la $k^{ième}$ composante de Y vaut 1.

1.6.1.2. Noeud actif pour une sortie d'un automate de Moore

C'est un noeud correspondant à un état où cette sortie est égale à 1.

Le noeud associé à l'état s est actif pour la sortie o_j si et seulement si $(\lambda(s))_j = 1$.

1.6.1.3. Arc actif pour une sortie d'un automate de Mealy

C'est un arc correspondant à une transition où cette sortie est émise.

L'arc correspondant à la transition (s, s', i) est actif pour la sortie o_j si et seulement si $(\lambda(s, i))_j = 1$.

1.6.1.4. Monôme associé à une valeur d'une variable booléenne générale à N composantes

On appelle monôme associé à une valeur d'une variable booléenne générale à N composantes le monôme canonique à N variables, dont la $i^{ème}$ variable apparaît sous forme directe (complémentée) si le $i^{ème}$ bit de cette valeur est égal à 1 (0 respectivement). Par exemple, si Y = 101, alors le monôme associé à Y est $y3.\ \overline{y2}\ .y1$.

1.6.1.5. Monôme associé à un code d'un état ou à un noeud du graphe

C'est le monôme associé à la valeur de la variable Y associée à cet état et au noeud correspondant du graphe. On notera le monôme associé au code de l'état s MAC(s).

1.6.1.6. Monôme associé à un arc du graphe des états

C'est le produit du monôme associé au noeud source de l'arc et de la condition d'entrée apparaissant sur l'arc.

Le monôme associé à l'arc correspondant à la transition (s, s', i) est égal à MAC(s).i.

1.6.1.7. Expression booléenne associée à un ensemble de noeuds

C'est la somme des monômes associés à chacun des noeuds.

1.6.1.8. Expression booléenne associée à un ensemble d'arcs

C'est la somme des monômes associés à chacun des arcs.

1.6.1.9. Expression booléenne produite par un noeud

C'est l'expression booléenne associée à l'ensemble de ses arcs entrants.

L'expression produite par un noeud correspondant à un état S s'écrit
$\Sigma_{\{\text{transitions } (s', S, i)\}} MAC(s').i$

1.6.2. Génération des équations des variables internes et des sorties

1.6.2.1. Equation d'une variable interne

C'est la somme des expressions booléennes produites par tous les noeuds actifs pour cette variable.

L'équation pour la variable interne Y_k s'écrit

$$Y_k = \Sigma_{\{\text{transitions } (s', s, i) / C_k(s) = 1\}} MAC(s').i.$$

1.6.2.2. Equation d'une sortie pour un automate de Moore

C'est l'expression booléenne associée à l'ensemble des noeuds actifs pour cette sortie.

L'équation de la sortie o_j s'écrit

$$o_j = \Sigma_{\{\text{états } s/(\lambda(s))_j = 1\}} MAC(s)$$

1.6.2.3. Equation d'une sortie pour un automate de Mealy

C'est l'expression booléenne associée à l'ensemble des arcs actifs pour cette sortie.

L'équation de la sortie o_j s'écrit

$$o_j = \Sigma_{\{\text{transitions } (s, s', i)/(\lambda(s, i))_j = 1\}} MAC(s).i.$$

exemple

Considérons l'automate de Moore de la figure I-5. Affectons, par exemple, les codes 00, 01, 10 et 11 aux états s1, s2, s3 et s4 respectivement; les variables internes sont Y2 et Y1.

entrée	état courant	état suivant	sortie
0	s1	s2	0
1	s1	s3	0
0	s2	s4	0
1	s2	s2	0
-	s3	s1	1
-	s4	s1	1

Figure I-5: Exemple d'automate

Les noeuds actifs pour Y2 sont s3 et s4 et les noeuds actifs pour Y1 sont s2 et s4; les noeuds actifs pour la sortie o sont s3 et s4. On obtient alors les équations non minimisées suivantes:

$$Y2 = \overline{y2}\,.\,\overline{y1}\,.i + \overline{y2}\,.y1.\,\overline{i}$$

$$Y1 = \overline{y2}\,.\,\overline{y1}\,.\,\overline{i} + \overline{y2}\,.y1.i + \overline{y2}\,.y1.\,\overline{i}$$

$$o = y2.\,\overline{y1} + y2.y1$$

1.7. IMPACT DU CODAGE SUR LES ÉQUATIONS DES VARIABLES INTERNES ET DES SORTIES

1.7.1. Définitions préalables

1.7.1.1. Hypercube Booléen

On appelle hypercube Booléen de dimension N l'espace $\{0, 1\}^N$. L'hypercube sera aussi appelé N-cube.

1.7.1.2. Sommet

Un sommet de l'hypercube de dimension N est associé bijectivement à une valeur d'un vecteur à N composantes $(Y_1, ..., Y_N)$ où $Y_i \in \{0, 1\}$. Par exemple, (0, 1, 0) est un sommet de l'hypercube de dimension 3. La valeur booléenne associée à ce sommet est appelée code de ce sommet.

1.7.1.3. Sous-cube de dimension k

Un sous-cube de dimension k, ou k-cube, d'un hypercube de dimension N ($k < N$) est un ensemble de 2^k sommets distincts tels que (N - k) composantes soient invariantes (c'est-à-dire, égales à 0 ou 1) dans le code de ces sommets. Par exemple, pour N = 4, les sommets, définis par leurs codes (Y4, Y3, Y2, Y1), de l'ensemble {0010, 0011, 0110, 0111} constituent un 2-cube; les composantes invariantes sont Y4 et Y2.

1.7.1.4. Monôme caractéristique d'un sous-cube de dimension k

C'est le monôme à (N - k) variables associé aux (N - k) composantes invariantes pour ce k-cube. Par exemple, le monôme caractéristique du 2-cube du paragraphe précédent est $\overline{Y4}.Y2$.

1.7.1.5. Distance de Hamming

La distance de Hamming entre deux sommets v et v' d'un hypercube, noté H(v, v'), est défini par le nombre de composantes dont les valeurs sont différentes. Par exemple, H(010, 111) est égal à 2.

1.7.2. Impact du codage sur les équations des variables internes et des sorties

Un codage optimisé a pour but de simplifier les équations de variables internes et des variables de sortie. Pour cela, des propriétés ou situations intéressantes donnant lieu à des minimisations potentielles seront détectées dans le graphe des états. Une approche globale étant trop complexe, on recherche des situations locales, et une notion de coût permettra de définir une heuristisque de type glouton pour optimiser ce gain.

1.7.2.1. Impact d'un codage sur l'écriture des variables internes et des sorties

Considérons l'automate de Moore donné en figure I-5. Si les codes 10, 01, 11 et 00 sont affectés respectivement aux états s1, s2, s3 et s4, on obtient les équations minimisées suivantes composées d'un total de 14 littéraux; un littéral correspond à une occurrence de variable directe ou complémentée:

$$Y2 = \overline{y2}\,.\,\overline{y1} + y2.(y1 + i)$$
$$Y1 = y2.\,\overline{y1} + \overline{y2}\,.y1.i$$
$$o = \overline{y2}\,.\,\overline{y1} + y2.y1$$

Par contre, si on choisit les codes {00, 01, 11, 10}, on obtient de nouvelles équations minimisées comprenant 9 littéraux:

$$o = y2$$
$$Y2 = \overline{y2}\,.(\overline{y1}\,.i + y1.\overline{i}\,)$$
$$Y1 = \overline{y2}\,.(\overline{y1} + i)$$

On constate ainsi que le deuxième codage conduit à des équations dont la complexité est beaucoup plus réduite en termes de nombre de littéraux. Analysons cet exemple et pour cela, écrivons les équations symboliques des variables internes et des sorties. Les équations symboliques sont des équations écrites en fonction des codes des états non encore connus; s_i regroupe symboliquement les variables internes du code de s_i.

$$s1 = MAC(s3) + MAC(s4)$$
$$s2 = MAC(s1).\overline{i} + MAC(s2).i$$
$$s3 = MAC(s1).i$$
$$s4 = MAC(s2).\overline{i}$$
$$o = MAC(s3) + MAC(s4)$$

Au vu de ces équations, un certain nombre de remarques peuvent être faites.

Considérons l'expression de s1; si les états s3 et s4 ont des codes adjacents, l'expression de s1 se réduit à un seul monôme. On dit qu'il y a **fusion de monômes potentiels**. Pour de telles situations, on parlera de situations à fusion potentielle de monômes. Les noeuds impliqués dans une telle situation constituent un groupe d'adjacence; les contraintes de codage associées à ces noeuds consiste à les placer sur un cube de l'hypercube à l'exclusion de tout autre noeud. En respectant cette contrainte, la fusion de monômes devient effective.

Considérons l'expression de s2; les monômes impliqués ne peuvent pas se réduire à un seul monôme. Par contre, si on donne à s1 et s2 des codes ayant

plusieurs bits en commun, alors on peut effectuer une factorisation. Appelons M le monôme commun à MAC(s1) et MAC(s2); alors MAC(s1) peut s'écrire M.M1 et MAC(s2) peut s'écrire M.M2. L'expression de s1 devient alors M.(M1.$\bar{i}$ + M2.i). On parle pour ce type de situation de **situation à factorisation potentielle de monômes**. D'une manière générale, dans une telle situation, pour l'équation d'une variable interne ou de sortie, on trouve une somme de la forme MAC(s1).e1 + MAC(s2).e2. Si, d'une part, MAC(s1) et MAC(s2), et d'autre part, e1 et e2 ont des parties communes M12 et e12 respectivement, alors cette somme peut se réécrire M12.e12.(M1'.e1' + M2'.e2'). La partie M12 dépendra du nombre de bits en commun dans les codes de s1 et s2. Pour augmenter le gain d'une telle situation, il est souhaitable que s1 et s2 aient des codes les plus proches possibles au sens de la distance de Hamming.

Considérons maintenant les expressions de s1 et o; elles comportent toutes les deux l'expression MAC(s3) + MAC(s4). Cette expression constitue donc une **partie commune pour les deux fonctions** s1 et o; elle peut ainsi être redéfinie en tant que sous-fonction, et être utilisée dans ces deux fonctions. D'une manière générale, dans une telle situation, on est en présence de deux fonctions dont les expressions partielles sont données de la manière suivante:

F1 = MAC(s1).e1
F2 = MAC(s2).e2

De la même manière ici, si, d'une part, MAC(s1) et MAC(s2), et d'autre part, e1 et e2 ont des parties communes M12 et e12 respectivement, alors ces deux fonctions peuvent être réécrites de la manière suivante:

F1 = M12.e12.M1.e1' (MAC(s1) = M12.M1 et e1 = e12.e1')
F2 = M12.e12.M2.e2' (MAC(s2) = M12.M2 et e2 = e12.e2')

M12.e12 peut être redéfinie comme sous-fonction commune à F1 et F2 dans une phase de factorisation ultérieure. Or, dans la réalisation globale d'un ensemble de fonctions, une sous-fonction n'est implantée qu'une fois, et son utilisation pour plusieurs fonctions constituera un gain important.

1.7.3. Stratégie générale

L'algorithme de codage que nous proposons repose essentiellement sur deux étapes:

A - Recherche des situations à fusion de monômes; Obtention d'un ensemble de groupes d'adjacence et déduction de contraintes de codage, et codage des états respectant un maximum de contraintes

B - Recherche des situations à factorisation de monômes; Déductions de contraintes en termes de distance et codage final

L'algorithme commence par la recherche des situations pour lesquelles des simplifications par fusion de monômes sont possibles. Ceci donne lieu à des ensembles de noeuds, appelés groupes d'adjacence, devant être affectés à des sous-cubes de l'hypercube. Le nombre généralement élevé de tels groupes fait que les contraintes de codage associées à ces groupes d'adjacence ne peuvent pas toutes être respectées en général. Il s'avère alors nécessaire de définir une pondération de ces groupes afin de définir une priorité de respect des contraintes associées. Cette pondération comporte un coût local intrinsèque au groupe considéré, et une contribution globale dûe aux intersections pouvant exister avec d'autres groupes d'adjacence.

Après ces étapes, il convient d'affecter chaque groupe d'adjacence à un sous-cube de l'hypercube. Les groupes sont traités successivement par ordre de coût décroissant; pour chaque groupe, la recherche d'un cube vérifiant certaines propriétés du groupe est effectuée. Si aucun cube n'est trouvé, le groupe est abandonné et on passe au prochain groupe, et ceci jusqu'à ce que tous les groupes aient été considérés. L'immersion des groupes d'adjacence étant terminée, on peut alors déterminer les codes possibles pour chaque état. A ce niveau, certains états peuvent ne pas avoir de code unique. La détermination d'un code unique pour ces états sera alors effectuée au vu des étapes suivantes.

L'étape suivante consiste à rechercher les situations à factorisation de monômes. De la même manière ici, pour définir une priorité de respect des contraintes associées à ces situations, une pondération est nécessaire. On montrera par la suite que la pondération utilisée sera un facteur d'attraction calculé pour tout couple d'états. Ainsi, des codes proches au sens de la distance

de Hamming sont donnés aux couples d'états ayant les plus fortes attractions. Ceci est effectué en considérant pour chaque état l'ensemble des codes pouvant lui être affecté et en choisissant celui pour lequel l'attraction est maximale. Cette dernière étape a pour effet de figer le code de chaque état.

La philosophie générale de l'approche consiste à privilégier les fusions de monômes car dans ce cas seulement, une réduction en termes de surface, de connectique et de chemin critique sera assurée dans la réalisation finale. Seulement quand ces situations ont été traitées, on considère les situations de deuxième type, et l'on pourra constater que le gain supplémentaire sera beaucoup plus réduit.

1.8. RÉALISATION MATÉRIELLE

1.8.1. Schéma logique

Pour la réalisation matérielle d'un automate, il sera nécessaire de choisir N points de mémorisation (bascules) qui serviront à mémoriser l'état courant de l'automate. Divers types de bascules sont disponibles; citons les bascules D, JK et T. Les équations de sortie ne dépendant que de l'état courant, et éventuellement des entrées (Mealy), seront indépendantes du type de bascule choisi. Par contre, les équations de calcul des variables internes (fonction d'état suivant) seront calculées en fonction du type de la bascule. Dans cette étude, on ne considèrera que le cas des bascules D; dans ce cas, les logiques à synthétiser pour le calcul d'état suivant et des sorties sont les logiques associées aux applications δ et λ.

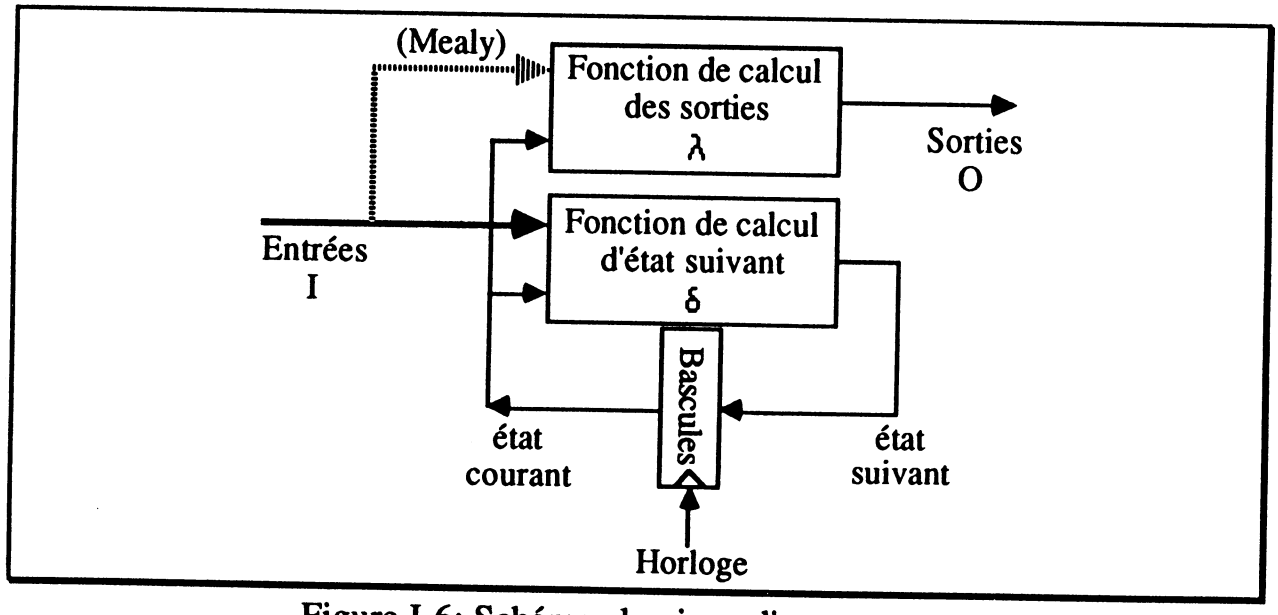

Figure I-6: Schéma classique d'un automate

Le schéma classique du circuit matériel pour la synthèse d'un automate est donné dans la figure précédente.

1.8.2. Cibles logiques

La logique utilisée pour la synthèse des fonctions δ et λ peut être l'un des deux types suivants, soit de la logique deux-couches, soit de la logique multicouche.

Dans une logique de type deux-couches, le nombre de portes traversées par un signal d'entrée jusqu'à la sortie est au plus égal à deux. Par contre, dans une logique de type multicouche, le nombre de portes traversées peut être quelconque.

1.8.2.1. Logique deux-couches

La logique deux-couches est généralement utilisée pour la réalisation de fonctions écrites sous forme de sommes de monômes. Une première couche réalise les monômes au vu des variables d'entrée au moyen de portes ET, et une deuxième couche réalise la fonction au vu des monômes générés au moyen de portes OU. Un circuit classique de type deux-couches est le PLA, qui est un circuit de structure régulière composé de deux étages dits étage ET et étage OU. Une représentation formelle d'un PLA est donné dans l'exemple suivant. Le critère principal d'optimisation d'un PLA est le nombre de monômes qu'il contient car la surface de ce PLA est directement proportionnel à ce nombre.

exemple

La figure suivante donne la représentation formelle d'un PLA. Le PLA décrit l'ensembles des fonctions f1, f2 et f3 définies par

$$f1 = a + b.\overline{c} + a.b.\overline{c}$$

$$f2 = \overline{a}.\overline{d} + b.\overline{c}.d + a.c + a.b.\overline{c}$$

$$f3 = \overline{a}.\overline{d} + a + \overline{b}.\overline{c} + .a.c$$

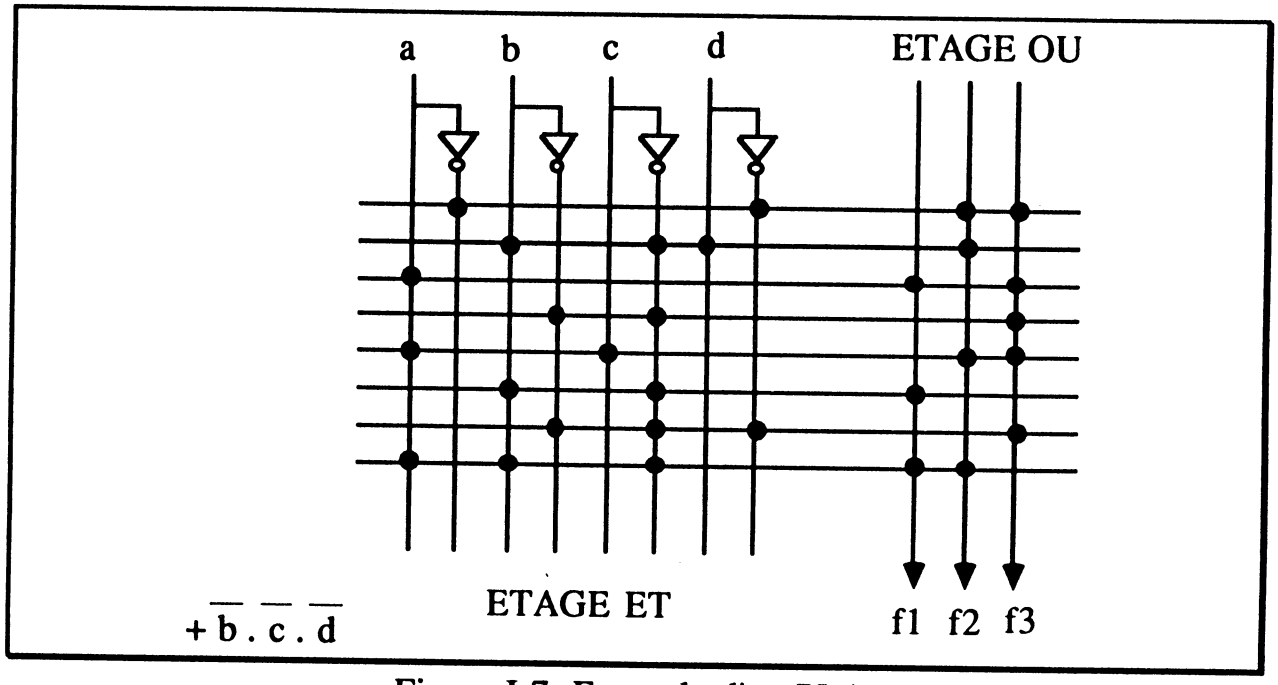

Figure I-7: Exemple d'un PLA

1.8.2.2. Logique multicouche

La logique multicouche est généralement utilisée pour la réalisation de fonctions écrites sous forme d'expressions booléennes quelconques à partir des opérateurs classiques de somme, produit et négation.

exemple

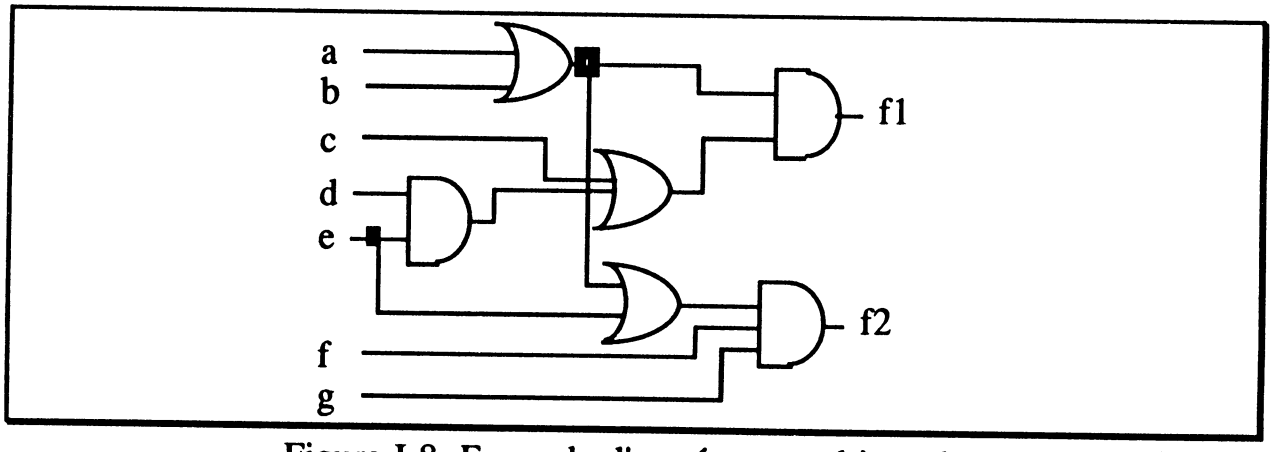

Figure I-8: Exemple d'un réseau multicouche

La logique le plus généralement utilisée pour la réalisation consiste en un ensemble de cellules dites "cellules standards". D'autres types de circuits qui sont de plus en plus utilisés actuellement sont les composants à logique programmable

(PLD: Programmable Logic Devices); citons par exemple les PALs (Programmable Array Logic) et les PGAs (Programmable Gate Arrays).

Dans cette étude, nous nous concentrerons sur le codage optimisé d'automates synthétisés sur cellules standards. Nous nous intéresserons aussi à évaluer ce codage pour d'autres cibles telles que les PLAs, PALs et PGAs de type Xilinx.

CHAPITRE II

RECONNAISSANCE DE SITUATIONS

2.1. RECONNAISSANCE DE SITUATIONS À FUSION POTENTIELLE DE MONOMES

Comme annoncé dans le chapitre précédent, deux types de situations seront recherchées au niveau du graphe de contrôle. Ce chapitre a pour objet de présenter les diverses situations retenues et montrer comment elles doivent être traitées.

2.1.1. Définitions préliminaires

2.1.1.1. Expressions produites par un sous-graphe partiel

Ce sont les expressions produites par les noeuds de ce sous-graphe partiel.

2.1.1.2. Situation prédictive de minimisation

Une situation prédictive de minimisation est un sous-graphe partiel du graphe d'états tel que les expressions produites par ce sous-graphe partiel donnent lieu à des fusions de monômes ou à des factorisations pour les équations des variables internes et/ou des sorties. Dans le premier cas, on parlera de situations à fusion potentielle de monômes et dans le second cas, de situations à factorisation potentielle de monômes.

Les situations correspondent à des sous-graphes partiels définissant des (sous-) automates pour lesquels il existe des propriétés de dépendance réduites pour les variables internes et/ou les sorties.

Situations à fusion potentielle de monômes

Il s'agit de façon générale d'un ensemble de noeuds tel que des contraintes imposées sur les codes des états correspondants amènent des fusions de monômes dans l'expression des sorties et des variables internes produite par ces noeuds. Cette fusion peut se produire à l'intérieur de l'expression booléenne produite par un noeud ou entre des monômes appartenant à des expressions booléennes produites par des noeuds distincts actifs pour une variable interne ou pour une sortie.

Situations à factorisation potentielle de monômes

Il s'agit d'un ensemble de noeuds tel que des contraintes associées aux codes des états correspondants amènent des factorisations potentielles dans les équations des variables internes et/ou des sorties. De la même manière que dans

le cas précédent, des factorisations peuvent avoir lieu à l'intérieur de l'expression booléenne produite par un noeud, ou entre des monômes appartenant à des expressions booléennes produites par des noeuds distincts.

2.1.2. Situations à fusion de monômes

Quatre situations de ce type sont présentées, et pour chacune d'elles, on montre quelles sont les contraintes à imposer sur les codes des états et quelles sont les simplifications qui en découlent.

2.1.2.1. Situations concernant les sorties

Situation 1 : Situation relative à un ensemble de noeuds et concernant les fonctions de sortie d'un automate de Moore

Il s'agit du cas le plus simple où le sous-graphe partiel est réduit à un ensemble de noeuds sans arc. Cette situation concerne les équations de sortie pour un automate de Moore, car dans ce cas-là, les sorties sont associées aux états.

Considérons un ensemble de $p = 2^k$ noeuds actifs pour la même sortie o.

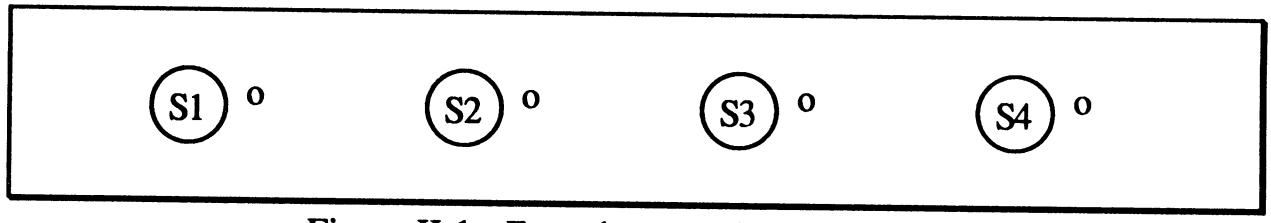

Figure II-1 : Etats émettant la même sortie

L'expression de o est égale à l'expression booléenne associée à l'ensemble de ces noeuds, à savoir $o = \Sigma_{\{n = 1 \text{ à } p\}} MAC(s_n)$.

Le but recherché est de remplacer cette expression par un unique monôme. Pour fusionner les monômes, on cherchera à placer ces noeuds sur un k-cube de l'hypercube, avec $k \geq \log_2(p)$, à l'exclusion de tout autre noeud.

Les p monômes de l'expression produite par la sortie o sont alors réduits au monôme caractéristique du k-cube utilisé. Ces p noeuds forment un groupe d'adjacence cubique ou plus simplement un groupe d'adjacence.

Situation 1' : Situation relative à un ensemble de noeuds source d'arcs étiquetés par la même condition et concernant les fonctions de sortie d'un automate de Mealy

Le sous-graphe partiel est ici réduit à un ensemble d'arcs sans noeud. Cette situation concerne les équations de sortie pour un automate de Mealy, car dans ce cas-là, les sorties sont associées aux transitions.

Considérons un ensemble de $p = 2^k$ arcs actifs étiquetés par la même condition d'entrée C et émettant la même sortie o.

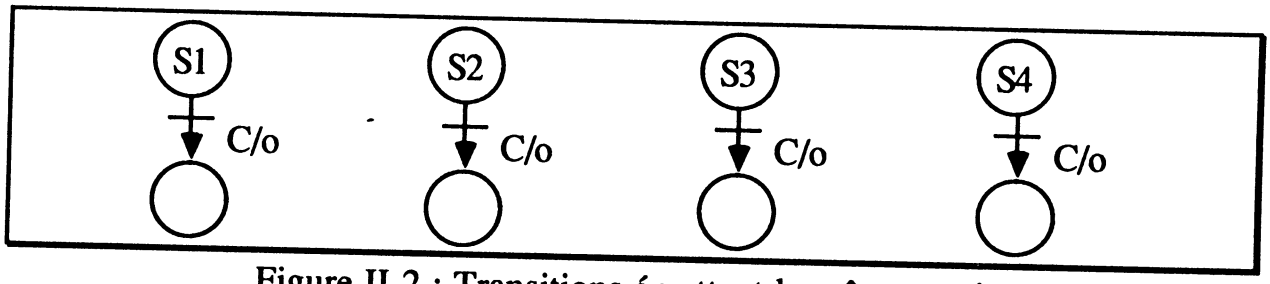

Figure II-2 : Transitions émettant la même sortie

La sous-expression produite pour la sortie o par ces noeuds est égale à l'expression booléenne associée à l'ensemble de ces arcs, à savoir $C.\Sigma_{\{n = 1 \text{ à } p\}} MAC(s_n)$, ou encore $C.\Sigma_{\{n = 1 \text{ à } p\}} MAC(s_n)$.

Les mêmes contraintes de codage s'appliquent donc aux p noeuds source des arcs concernés; le monôme unique obtenu pour la sous-expression de la sortie o est égal au produit de la condition C et du monôme caractéristique du k-cube.

2.1.2.2. Situations concernant les variables internes

Théorème 1 de dépendance réduite [HAR66]

Si, dans un automate d'états fini, la variable Y_j est codée suivant une bipartition Π_j et Y_j' suivant une bipartition Π_j', et si $\Pi_j' = \delta(\Pi_j, i)$, alors Y_j' ne dépend que de Y_j pour l'entrée i.

En d'autres termes, Y_j' s'écrit $i.\tilde{y}_j + SF$, où $\tilde{y}_j$ est égal à y_j ou $\overline{y_j}$.

L'utilisation de ce théorème sera illustrée dans le paragraphe suivant.

Application à la recherche de la situation 2: Situation concernant un ensemble d'arcs étiquetés par une même condition

Considérons dans le graphe d'états (figure II-3), un ensemble de 2^k arcs distincts étiquetés par une même condition C et leurs noeuds destination. Si on code les noeuds source de ces arcs par k variables $\{Y_1, ..., Y_k\}$ suivant k bipartitions $\{\Pi_1, ..., \Pi_k\}$, et les noeuds destination par k variables $\{Y_1', ..., Y_k'\}$ suivant k partitions $\{\Pi_1', ..., \Pi_k'\}$, avec $\Pi_j' = \delta(\Pi_j, C)$ pour j = 1 à k, alors les expressions produites par les noeuds destination de ces arcs pour chacune des variables $\{Y_1', ..., Y_k'\}$ se réduisent à un seul monôme sous la forme suivante:

$$Y_j' = C.\,\tilde{y}_j \text{ pour } j = 1 \text{ à } k$$

On peut aussi exprimer ceci en disant que l'on place les 2^k noeuds source sur un k-cube et les 2^k noeuds destination sur un k-cube en respectant les relations d'adjacence du cube source dans le cube destination lors de l'application état présent → état suivant pour cette entrée.

Les équations données plus haut sont déduites du théorème de dépendance réduite, et ne concernent que le sous-graphe considéré, soit k variables internes $\{Y_1, ..., Y_k\}$ pour les noeuds source et k variables internes $\{Y_1', ..., Y_k'\}$ pour les noeuds destination. Comme le codage est effectué dans un hypercube de dimension N, les N - k autres variables internes sont laissées invariantes pour l'ensemble des états source ainsi que pour les états destination. Les expressions obtenues pour les variables internes sont donc complétées de la manière suivante:

$$Y_j' = M.C.\,\tilde{y}_j \text{ pour } j = 1 \text{ à } k$$
$$Y_j' = 0 \text{ ou } M.C \text{ pour } j = k + 1 \text{ à } N$$

où M représente le monôme caractéristique du k-cube contenant les noeuds source. De façon plus précise, dans la dernière équation, $Y_j' = 0$ (resp.$Y_j' = M.C$) si le $j^{ième}$ bit invariant des codes des noeuds destination est 0 (resp. 1).

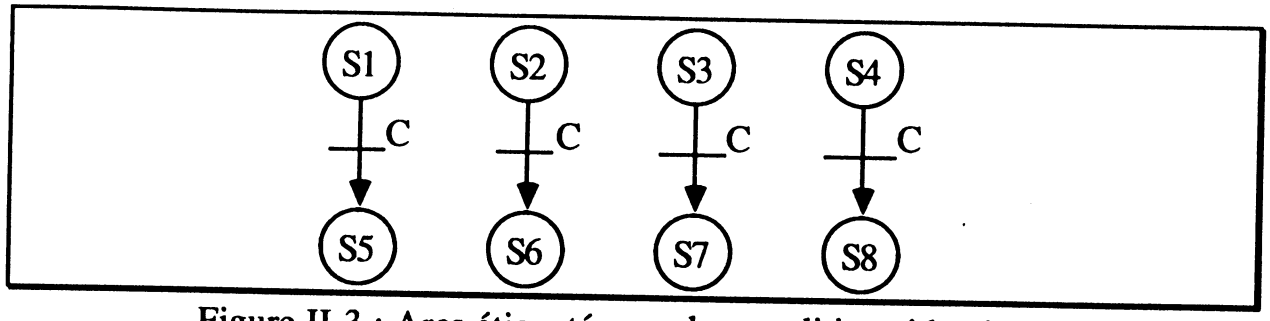

Figure II-3 : Arcs étiquetés par des conditions identiques

Considérons l'exemple de la figure II-3. Supposons que les 4 noeuds origine soient affectés à un 2-cube défini par les deux bipartitions {(s1, s2), (s3, s4)} et {(s1, s3), (s2, s4)}. On construit l'image de ces deux bipartitions par δ: les deux bipartitions {(s5, s6), (s7, s8)} et {(s5, s7), (s6, s8)} sont obtenues. Le but est donc de placer les noeuds s5, s6, s7 et s8 sur un 2-cube, défini par les bipartitions obtenues. Ceci est illustré dans la figure suivante.

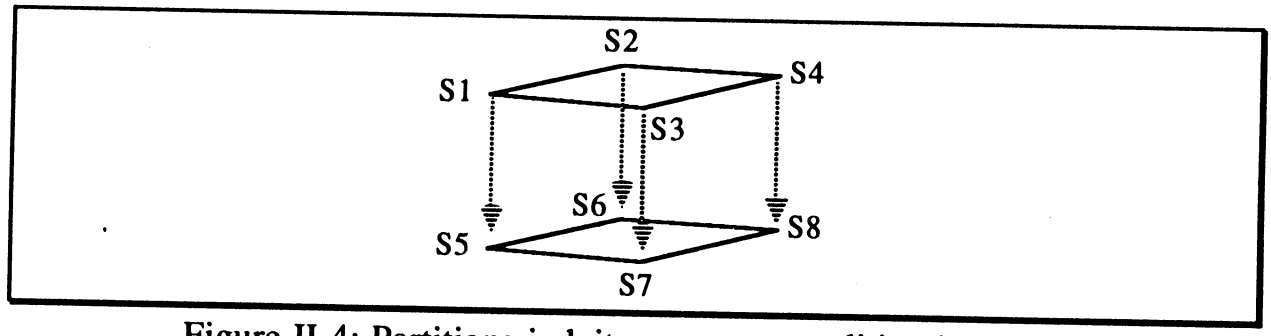

Figure II-4: Partitions induites par une condition identique

En résumé, les contraintes imposées sur les codes des états sont les suivantes:

Placer les 2^k noeuds source sur un k-cube; ceci définit les k bipartitions de ce k-cube

Construire les k bipartitions correspondantes pour le deuxième ensemble de 2^k noeuds en respectant la relation état courant → état suivant

Placer ces noeuds dans un k-cube définies par les k bipartitions obtenues

Considérons l'exemple de la figure II-3. Si les codes 0101, 0100, 0111 et 0110 sont affectés aux états s1, s2, s3 et s4 respectivement, un codage possible pour s5, s6, s7 et s8 est 1101, 1100, 1001 et 1000. M, le monôme caractéristique du cube source, est égal à $\overline{y_4} . y_3$

Les équations obtenues sont donc

$$Y4 = \overline{y_4} \cdot y_3$$
$$Y_3 = \overline{y_4} \cdot y_3 \cdot \overline{y_2}$$
$$Y_2 = 0$$
$$Y_1 = \overline{y_4} \cdot y_3 \cdot \overline{y_2}$$

En résumé, pour chacune des variables invariantes égale à 1 des codes des états successeurs, le monôme associé au k-cube contenant les états prédécesseurs sera obtenu. Les autres variables Y_j' produisent chacune un monôme égal au produit du monôme précédent et de $\tilde{y}_j$, y_j étant la variable invariante dans la partition correspondante.

Forme dégénérée 1 (situation 3): Situation réduite à un seul noeud destinataire d'arcs étiquetés par une même condition d'entrée et concernant les équations des variables internes

Cette situation correspond à un sous-graphe réduit à un noeud et à ses arcs entrants. Il s'agit d'un ensemble de $p = 2^k$ états qui, pour une condition identique C, ont le même état suivant.

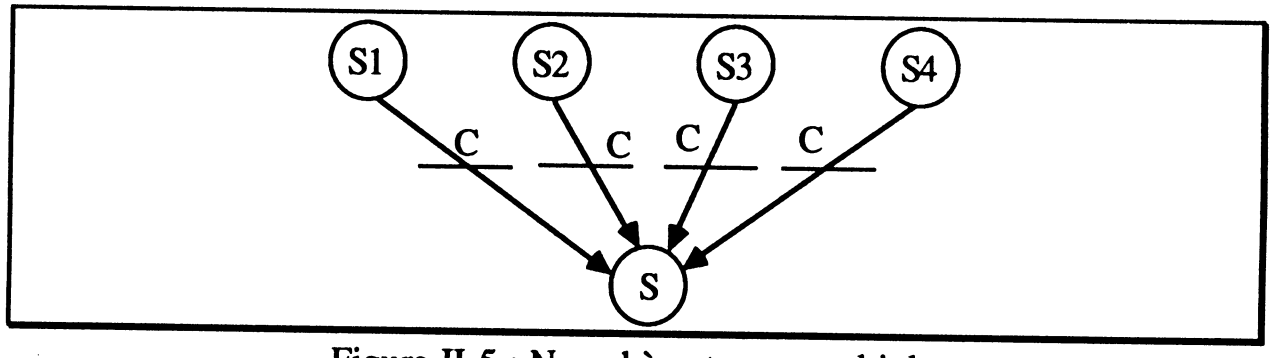

Figure II-5 : Noeud à entrance multiple

Le noeud concerné ici est le noeud s vers lequel convergent les arcs. L'expression booléenne produite par le noeud s est égale à $\Sigma_{\{n = 1 \text{ à } p\}} C.MAC(s_n)$, ou encore $C.\Sigma_{\{n = 1 \text{ à } p\}} MAC(s_n)$. C'est une expression incluse dans l'équation de toutes les variables internes pour lesquelles ce noeud est actif. Le but recherché est, comme dans le cas de la situation 1, de remplacer cette expression par un unique monôme. Pour ceci, il faut placer comme précédemment les p noeuds sur un k-cube; l'ensemble de ces p noeuds constituent donc un groupe d'adjacence. Les p monômes de l'expression produite

par le noeud s sont alors réduits au produit de la condition C et du monôme caractéristique du k-cube utilisé.

exemple

Dans l'exemple de la figure II-5, un ensemble de codes possibles pour les états s1, s2, s3 et s4 est {1000, 1001, 1010, 1011}. L'expression initiale $C.(y4.\overline{y3}.\overline{y2}.\overline{y1} + y4.\overline{y3}.\overline{y2}.y1 + y4.\overline{y3}.y2.\overline{y1} + y4.\overline{y3}.y2.y1)$ est ainsi réduite à $C.y4.\overline{y3}$.

Forme dégénérée 2:

Un cas particulier intéressant, car fréquent dans les graphes de contrôle, est celui d'un ensemble de-boucles étiquetées par une même condition d'entrée C. Il suffira alors de placer les états associés à ces boucles sur un k-cubes. En effet, chaque état s_n ici vérifie $s_n = \delta(s_n, C)$. Ainsi, toute partition Π de ces états vérifie $\Pi = \delta(\Pi, C)$. Donc, toute bipartition Π_j associée à une variable Y_j vérifie les hypothèses du théorème de Hartmanis. Les simplifications s'appliquent donc ici pour tout ensemble de bipartitions.

Théorème 2 de dépendance réduite.

Ce théorème est comparable au théorème 1, mais ici les rôles des entrées et des états sont inversés.

Considérons 2^k arcs ayant comme origine un même noeud s et étiquetés par 2^k prédicats correspondant aux 2^k monômes canoniques à k variables. On peut également dire que les 2^k prédicats sont codés suivant k bipartitions $\{\Pi_1, ..., \Pi_k\}$ des prédicats à k variables d'entrée.

Si les 2^k noeuds extrémités de ces arcs sont placés sur un k-cube selon l'ensemble des bipartitions $\{\Pi_1', ..., \Pi_k'\}$ définies par $\Pi_j' = \delta(s, \Pi_j)$ pour j = 1 à k, alors les expressions produites par les 2^k noeuds se réduisent à un seul monôme pour chacune des variables internes de l'ensemble $\{Y_1, ..., Y_k\}$ associé à $\{\Pi_1', ..., \Pi_k'\}$.

Les équations de ces variables s'écrivent alors

$Y_j = MAC(s).\tilde{i}_j$ pour j = 1 à k

$Y_j = 0$ ou MAC(s) pour j = k + 1 à N

Preuve:

Y_j étant codé suivant $\Pi_j' = \delta(s, \Pi_j)$, cela signifie que les noeuds actifs pour la variable Y_j sont les extrémités de l'ensemble des arcs étiquetés par une entrée i_j égale à 1 (ou 0), les autres entrées prenant les 2^{k-1} valeurs possibles. Ceci implique que $Y_j = MAC(s).\tilde{i_j}$, pour j = 1 à k.

Les autres variables internes, étant invariantes, leur équation est donnée par $Y_j = 0$ ou $Y_j = MAC(s)$ pour j = k + 1 à N

Application à la recherche de la situation 4: Situation relative à un ensemble d'arcs ayant un même noeud origine et concernant les fonctions de variables internes

Dans cette situation (figure II-6), on considère le cas de $p = 2^k$ arcs ayant un même noeud origine et étiquetés par des prédicats de la forme $C.C_n$, où $\{C_n\}$ est un ensemble de 2^k monômes canoniques à k variables. Ces monômes définissent ainsi k bipartitions pour les k variables d'entrée $\{i_1, ..., i_k\}$.

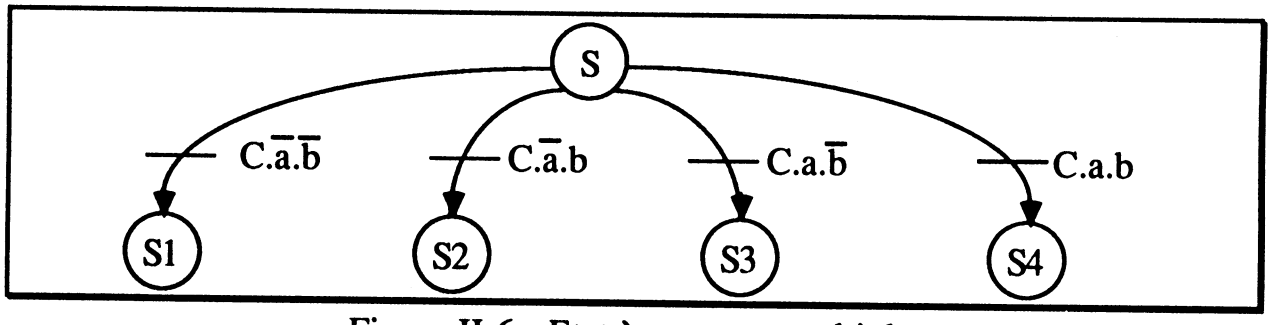

Figure II-6 : Etat à sortance multiple

L'application du théorème 2 impose de placer les 2^k noeuds extrémités de ces arcs sur un k-cube défini par les k bipartitions de noeuds induites par les k bipartitions des prédicats d'entrée. Ainsi, les expressions produites se réduisent toutes à un seul monôme comme suit (le prédicat C apparaît dans toutes les équations car il est présent sur tous les arcs):

$Y_j = MAC(s).C.\tilde{i_j}$ pour j = 1 à k

$Y_j = 0$ ou $MAC(s).C$ pour j = k + 1 à N

On note aussi que cette situation introduit une sous-fonction commune à toutes ces équations, qui est le monôme MAC(s).C.

Pour l'exemple donné à la figure II-6, cette situation impose de placer les noeuds associés à l'ensemble des états {s1, s2, s3, s4} sur un 2-cube. Les bipartitions des noeuds induites par a et b sont respectivement {(s1, s2), (s3, s4)} et {(s1, s3), (s2, s4)}. Il convient donc de placer chacun de ces quatre blocs sur un 1-cube.

Affectons, par exemple, les codes 0110, 0111, 0100 et 0101 aux états s1, s2, s3 et s4 respectivement. Ceci conduit à la génération des équations suivantes:

$$Y_4 = 0$$

$$Y_3 = MAC(s).(\overline{a}.\overline{b} + \overline{a}.b + a.\overline{b} + a.b)$$
$$= MAC(s)$$

$$Y_2 = MAC(s).(\overline{a}.\overline{b} + \overline{a}.b)$$
$$= MAC(s).\overline{a}$$

$$Y_1 = MAC(s).(\overline{a}.b + a.b)$$
$$= MAC(s).b$$

2.1.3. Extension des groupes d'adjacence de cardinalité différente de 2^k

Dans certains cas, les groupes d'adjacence ne contiennent pas un nombre de noeuds égal à une puissance de 2. Dans ce cas, si cela est possible, on les complètera par des états virtuels n'apparaissant pas dans l'automate. Nous montrerons ultérieurement comment effectuer cette opération et quelles sont les conditions nécessaires pour cela.

2.1.3.1. Etats fantômes

Lors du codage compact d'un automate, on utilise un nombre de bits N tel que N soit le plus petit entier vérifiant $2^N \geq |S|$. Pour certains automates, cette inégalité est stricte; il y a alors $2^N - |S|$ codes inutilisés. De manière à utiliser l'espace complet des codes, on complètera l'ensemble des états par un ensemble d'états virtuels noté $F = \{f_1, ..., f_{|F|}\}$ avec $|F| = 2^N - |S|$; ces états sont appelés états fantômes.

Cela permet de définir un sur-automate (I, O, S', δ', λ') où

= S ∪ F

δ' est définie par

- si s ∈ S alors $\delta'(s, i) = \delta(s, i)$
- si s ∈ F alors $\delta'(s, i)$ est un élément quelconque de S'

λ' est définie par

- si s ∈ S alors $\lambda'(s, i) = \lambda(s, i)$
- si s ∈ F alors $\lambda'(s, i)$ est un élément quelconque de O

ou

- si s ∈ S alors $\lambda'(s) = \lambda(s)$
- si s ∈ F alors $\lambda'(s)$ est un élément quelconque de O

L'extension de l'automate initial entraîne l'existence de valeurs non définies dans les fonctions booléennes associées aux variables internes et aux sorties, et permet une meilleure simplification de ces fonctions.

2.1.3.2. Groupes d'adjacence incomplets

Dans certaines situations, le nombre de noeuds impliqués dans un groupe d'adjacence peut être inférieur à 2^k. Or, les contraintes de codage imposent que les noeuds impliqués dans chaque contrainte soient placés sur un k-cube. Dans ce cas-là, si cela est possible, on complètera le groupe d'adjacence avec un nombre d'états fantômes de façon à ce que le nouveau groupe contienne 2^k noeuds. On montrera ultérieurement comment cette opération est réalisée et quelles sont les conditions nécessaires pour le faire.

2.1.4. Conclusion sur la phase de reconnaissance de situations

Au vu du graphe d'états, un certain nombre de contraintes sont imposées sur le codage des groupes d'adjacence associés aux différentes situations détectées pour préparer un maximum de simplifications ultérieures. Néanmoins, le nombre de groupes d'adjacence peut être suffisamment grand pour qu'il ne soit pas possible de respecter les contraintes induites par tous ces groupes. Dans ce cas-là, il s'avèrera nécessaire d'en satisfaire un certain nombre de façon à obtenir le plus grand gain possible dans une approche de type glouton. Pour cela, il est nécessaire d'introduire une pondération des groupes d'adjacence dans le but de définir des priorités sur ces groupes.

2.1.5. Gain associé à un groupe d'adjacence

La pondération utilisée pour chaque groupe sera le gain qu'on espère obtenir en satisfaisant les contraintes de codage associées. Ce gain sera évalué en faisant la somme de deux gains, un gain local évalué en fonction du groupe isolé, et un gain global prenant en compte les interactions avec d'autres groupes. En effet, deux groupes peuvent contenir des noeuds communs. La satisfaction des contraintes associées à un groupe peut induire la satisfaction partielle des contraintes associées à un autre groupe en plaçant sur un sous-cube l'ensemble des noeuds communs. La prise en compte des intersections entre groupes permet ainsi de créer des sous-fonctions communes entre différentes fonctions. Cette considération nous amène à donner une grande importance dans cette étude à l'intersection de groupes d'adjacence.

2.1.5.1. Gain local associé aux situations

Le gain local est le nombre de littéraux gagnés en plaçant un groupe d'adjacence sur un cube. Cette évaluation se fera en considérant le coût induit par le meilleur codage de ce groupe comparé au coût d'un mauvais codage (pire cas ou cas moyen). Le gain utilisé sera alors égal à la différence de ces deux coûts.

L'évaluation des gains est faite en fonction du nombre de littéraux car ce nombre reflète de manière assez correcte la "quantité" de logique utilisée. De plus, dans le cas de fusion de monômes, une réduction significative de la connectique est également observée.

Situation 1 et 1'

Dans la situation 1', l'ensemble des monômes utilisés est $\{C.MAC(s_n)$ pour n = 1 à p$\}$. Le monôme C étant en commun, il ne sera généré qu'une fois. Par contre, les p monômes $MAC(s_n)$ seront tous générés. Chacun de ces monômes étant construit sur N variables, le nombre de littéraux impliqués est égal à pN.

Dans le meilleur cas, on ne génère que le produit de C par un monôme composé de N - k littéraux (associé au k-cube).

Le gain s'obtient en faisant la différence entre les littéraux générés par les monômes seulement, car dans les deux cas, l'expression C est générée, soit

(p - 1)N + k. Dans le cas de la situation 1, le raisonnement est analogue et conduit au même gain.

Situation 2

Pour cette situation, une évaluation aussi précise que pour les situations précédentes n'est pas possible, car elle concerne plusieurs variables internes à la fois. On effectuera donc une évaluation moyenne et certaines hypothèses seront faites sur le nombre de bits à 1 dans le code des états.

Pour rendre l'évaluation des coûts plus claire, on va considérer les équations dans les deux cas.

Dans le meilleur cas, les équations des variables internes s'expriment sous la forme:

$SF = C.MS$ où MS est le monôme associé au k-cube des noeuds prédécesseurs

$Y_j = SF.\tilde{y_j}'$ pour $1 \leq j \leq k$ (les k variables internes associées aux partitions)

$Y_j = 0$ ou SF pour $k < j \leq N$ (en moyenne, la moitié des variables sont égales à 1)

Dans le pire cas, les expressions seront de la forme :

$SF = C$

$SF_r = SF.MAC(s_r)$ où s_r dénote le $r^{ième}$ noeud prédécesseur

$Y_j = \Sigma_{\{r \in R_j\}} SF_r$ pour $1 \leq j \leq N$ où $R_j = \{r / C_j(s_r') = 1\}$ et s_r' dénote le $r^{ième}$ noeud successeur

Dans le meilleur cas, si les contraintes de codage sont satisfaites, k variables internes génèrent 2 littéraux chacune, et en moyenne, (N - k)/2 autres variables internes génèrent 1 littéral chacune. On ne prendra pas en compte la contribution de C car ce monôme apparaît dans les deux cas. MS étant un monôme à N - k littéraux, le coût total est donc égal à (3N + k)/2.

Dans le pire cas, d'une manière générale, tous les monômes associés aux p arcs sont utilisés et redéfinis comme sous-fonctions (SF_r). Chacune de ces p

sous-fonctions génèrent (1 + N) littéraux. De plus, on peut supposer qu'en moyenne, N/2 bits sont égaux à 1 dans le code de chaque état successeur. Au total, pour les p états successeurs, le nombre de littéraux utilisés est en moyenne égal à pN/2, d'où un total de pN/2 + p(1 + N), ou encore p(1 + 3N/2) littéraux.

Le gain obtenu est donc égal à 3N(p - 1)/2 + p - k/2, ce qui peut être approché par 3N(p - 1)/2, le terme (p - k/2) étant négligeable pour de petites valeurs de p et de k.

Situation 3

L'évaluation du gain pour cette situation est la même que pour la situation 1', à la différence près que dans cette dernière, seule la fonction de sortie bénéficie de ce gain. Dans le cas présent, le gain est obtenu pour chacune des variables internes égale à 1 dans le code du noeud successeur, et doit être donc multiplié par le nombre de variables pour lesquelles ce noeud est actif. Ce nombre n'étant pas connu d'avance, on supposera qu'une moyenne de N/2 bits sont égaux à 1 d'où le gain final N[(p - 1)N + k]/2.

Situation 4

Dans le meilleur cas, on en arrivera à une écriture des expressions des variables internes sous la forme:

$SF = C.MAC(S)$

$Y_j = SF.\tilde{i}_j$ pour $1 \leq j \leq k$ (les k variables internes associées aux partitions)

$Y_j = 0$ ou SF pour $k < j \leq N$ (en moyenne, la moitié des variables sont égales à 1)

Dans le pire cas, les expressions seront de la forme:

$SF = C.MAC(S)$

$SF_r = SF.M_r$ où M_r représente le monôme canonique à k variables associé au $r^{ième}$ arc pour $1 \leq r \leq 2^k$

$Y_j = \Sigma_{\{r \in R_j\}} SF_r$ pour $1 \leq j \leq N$ où $R_j = \{r / C_j(s_r) = 1\}$

Dans le meilleur cas, si les contraintes de codage sont satisfaites, k variables internes génèrent 2 littéraux chacune, et en moyenne, (N - k)/2 variables

génèrent 1 littéral chacune. On ne prendra pas en compte la contribution de SF car elle apparaît dans les deux cas. Le coût total est donc égal à (N + 3k)/2.

Dans le pire cas, d'une manière générale, tous les monômes associés aux 2^k arcs sont utilisés et redéfinis comme sous-fonctions. Chacune de ces sous-fonctions génèrent (k + 1) littéraux. De plus, on peut supposer qu'en moyenne, N/2 bits sont égaux à 1 dans le code de chaque état successeur. Au total, pour les 2^k états successeurs, le nombre de littéraux utilisés est en moyenne égal à $N/2.2^k$, d'où un total de $N/2.2^k + 2^k(k + 1)$, ou encore $2^k(N/2 + k + 1)$ littéraux.

En réécrivant le coût dans le meilleur cas par (N/2 + k + 1 + (k/2 - 1)), le gain obtenu est pris égal à $(2^k - 1)(N/2 + k + 1)$, en négligeant le terme (k/2 - 1), les valeurs de k n'étant jamais très grandes.

2.1.5.2. Gain global

Les coûts locaux définis plus haut permettent de prédire le gain attendu pour la logique produite isolément par les noeuds de chacune des situations si les contraintes de codage imposées aux états associés sont respectées. Néanmoins, ce coût local ne permet pas une prise en compte du gain réel obtenu en satisfaisant les contraintes d'une situation, si en ce faisant, les contraintes d'une autre situation peuvent être satisfaites ou partiellement satisfaites.

Par exemple, considérons deux groupes d'adjacence {s1, s2, s3, s4} et {s1, s2, s5, s6}. Si on réussit à placer les noeuds du premier groupe sur un 2-cube, alors le placement de {s1, s2} sur un 1-cube, sous-cube du 2-cube obtenu, contribue partiellement au placement des états du deuxième groupe sur un 2-cube.

Le gain global pour un groupe d'adjacence est égal au gain local pour ce groupe augmenté de la somme des gains des intersections de ce groupe avec les autres groupes d'adjacence.

Pour l'exemple donné, le gain global du groupe d'adjacence {s1, s2, s3, s4} est égal à GainLocal1({s1, s2, s3, s4}) + GainLocal2({s1, s2}) où GainLocal1 et GainLocal2 sont les fonctions de gain local associées aux types des deux groupes d'adjacence respectivement. Ceci aura pour conséquence de traiter en premier lieu les contraintes ayant beaucoup de parties intersectantes.

2.2. RECONNAISSANCE DE SITUATIONS À FACTORISATION POTENTIELLE DE MONOMES

Les situations que l'on considèrera ici sont celles pour lesquelles une simplification maximale par fusion de monômes n'est pas possible, mais où des factorisations ou des mises en commun d'expressions sont possibles. Ces situations sont analogues aux quatre situations précédentes sauf pour les conditions apparaissant sur les arcs.

2.2.1. Définitions préalables

Nous allons donner dans cette partie les définitions de base nécessaires pour comprendre quel type de factorisation nous recherchons [BRA82], [BRA87].

2.2.1.1. Produit algébrique

Soient f et g deux expressions polynomiales de deux fonctions booléennes. Le produit algébrique f.g existe et est unique si f et g dépendent d'ensembles de variables disjoints. Il est défini par :

$f.g = \Sigma\, m_i.m_j$ pour i=1..n et pour j=1 à m

avec $f = \Sigma\, m_i$ pour i=1 à n

et $g = \Sigma\, m_j$ pour j=1 à m

exemple

f = a + b + c

g = d + !e

f.g = a.d + a.!e + b.d +b.!e + c.d + c.!e

Remarquons que le produit de deux expressions polynomiales dont les ensembles de variables ne sont pas disjoints est défini par les relations suivantes :

a.!a = 0

a.a = a

Ainsi on peut étendre le produit algébrique de deux expressions polynomiales.

exemple

exp1 = a.c + b.!e

exp2 = a.e

exp1.exp2 = a.c.a.e + b.!e.e = a.c.e + 0 = a.c.e

2.2.1.2. Division algébrique

g est un diviseur algébrique de f si :

f = g.h + r où g.h est un produit algébrique,

il n'existe pas h' tel que h < h' et f = g.h' + r

et r est une expression polynomiale (h<h' <=> h≤h' et h ≠ h').

Le quotient h et le reste r sont uniques.

g est un facteur algébrique de f si f = g.h

exemples

f = a.b.c + a.b.d +a.b.e + d

g = c + d + e est un diviseur algébrique de f.

= a.b est le quotient de la division.

r = d est le reste de la division.

2.2.1.3. Expression libre

Une expression polynomiale f est libre s'il n'existe pas de monôme m (avec m différent de 1) qui soit un facteur algébrique de f.

exemples

+ a.c n'est pas libre car a est un facteur algébrique

+ c est libre

n'est pas libre car a et b sont des facteurs algébriques triviaux.

2.2.1.4. Noyaux algébriques

k est un noyau algébrique d'une expression polynomiale f si k = f/c (division algébrique) où c est un monôme et k est une expression polynomiale libre. c est appelé le conoyau de k.

On note K(f) l'ensemble des noyaux algébriques de f. Il est intéressant d'ordonner cet ensemble en le partitionnant en sous-ensembles $K_i(f)$. Ainsi définit-on le degré d'un noyau de la façon suivante :

- un noyau k est dit un noyau de degré 0 s'il n'accepte comme noyau que lui-même.

- un noyau k est de degré n s'il a au moins un noyau de degré (n-1) mais aucun noyau de degré (n) ou plus excepté lui-même.

$K_0(f)$ est l'ensemble des noyaux de degré 0 et $K_i(f)$ est l'ensemble des noyaux de degré i. On obtient la partition suivante des noyaux de f, K(f) :

$\{K_0(f) , K_1(f) , ... K_{n-1}(f) , K_n(f)\} = K(f)$

On peut remarquer aussi qu'un noyau est formé de plus d'un monôme car un monôme n'est pas une expression libre. On peut considérer que f est un noyau algébrique d'elle-même et que le conoyau associé est 1.

2.2.1.5. Noyau global

On appelle k noyau global s'il est le noyau de plusieurs fonctions booléennes.

exemples

$f_1 = a.b.c + a.b.d + a.e.f + g$

$k_0 = c + d$ conoyau : a.b degré 0

$k_1 = b.c + b.d + e.f$ conoyau : a degré 1

k_0 est un noyau de degré 0 de f_1 et aussi un noyau de degré 0 de k_1.

$f_2 = a.b.c + a.b.e + d.f.c + d.f.e$

$k_0 = c + e$ conoyau : a.b degré 0

conoyau : d.f

Un même noyau peut avoir plusieurs conoyaux associés.

2.2.2. Application à la recherche de situations

Les situations étudiées concernent les arcs étiquetés par des **entrées intersectantes** en aval d'états ayant des codes intersectants. Un ensemble d'arcs sont étiquetés par des entrées intersectantes s'il existe au moins un littéral commun dans les conditions apparaissant sur ces arcs.

On appelle **monôme de proximité d'un ensemble de noeuds** le monôme associé aux composantes invariantes des codes des états associés à ces noeuds.

On appelle **monôme de proximité d'un ensemble d'arcs** le monôme composé des littéraux apparaissant dans toutes les conditions associées aux arcs.

2.2.2.1. Situation 1: Arcs à entrées intersectantes

Considérons un ensemble d'arcs étiquetés par des entrées intersectantes. Soit M le monôme de proximité des noeuds source et soit C le monôme de proximité des arcs ayant ces noeuds comme origines. Alors pour toute variable interne égale à 1 dans au moins deux noeuds distincts, il y a création d'une partie de conoyau égal à M.C dans l'expression de cette variable. Si plusieurs variables internes sont actives dans au moins deux noeuds distincts, alors le monôme M.C devient une partie de conoyau commune à plusieurs fonctions.

Cette situation est illustrée dans la figure suivante:

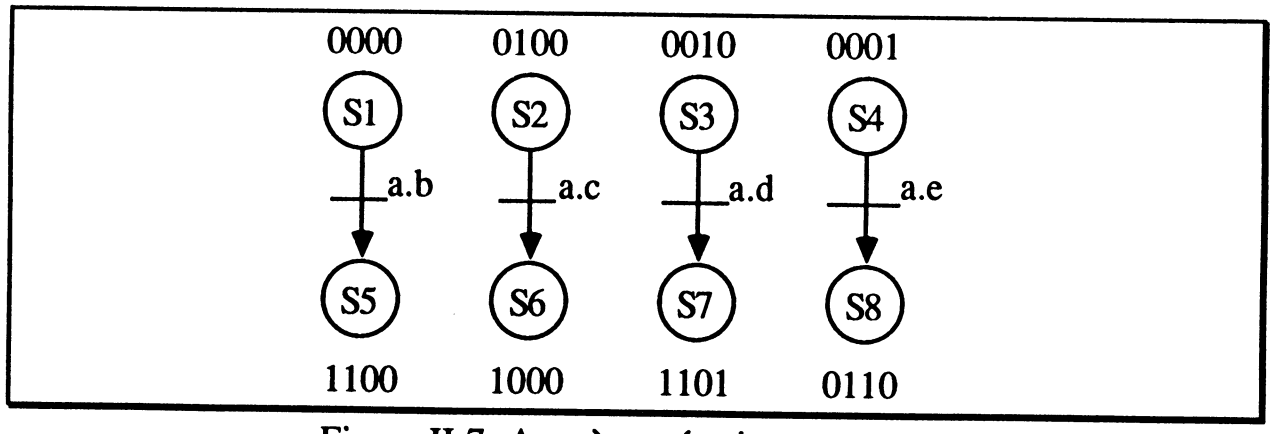

Figure II-7: Arcs à entrées intersectantes

Ici M est égal à $\overline{y4}$ et C est égal à a. Ecrivons les équations des variables internes Y4 et Y3.

$$Y4 = a.\overline{y4}.\overline{y1}\ (\overline{y3}.\overline{y2}.b + y3.\overline{y2}.c + \overline{y3}.y2.d)$$
$$Y3 = a.\overline{y4}.\overline{y3}.(\overline{y2}.\overline{y1}.b + y2.\overline{y1}.d + \overline{y2}.y1.e)$$

On note que $a.\overline{y4}$ est une partie de conoyau commune à Y3 et Y4.

2.2.2.2. Situation 2: Noeud à sortance multiple

Cette situation est un cas particulier de la situation 1, où les noeuds source sont confondus en un seul noeud s. Soit C le monôme de proximité des arcs ayant ce noeud comme origine. Si p noeuds sont actifs pour une variable interne Y_j, alors le monôme MAC(s).C est une partie de conoyau de l'expression de Y_j. Si plusieurs variables sont égales à 1 dans au moins deux noeuds successeurs, alors ce monôme constitue une partie de conoyau commune à plusieurs fonctions.

La figure suivante illustre ce cas.

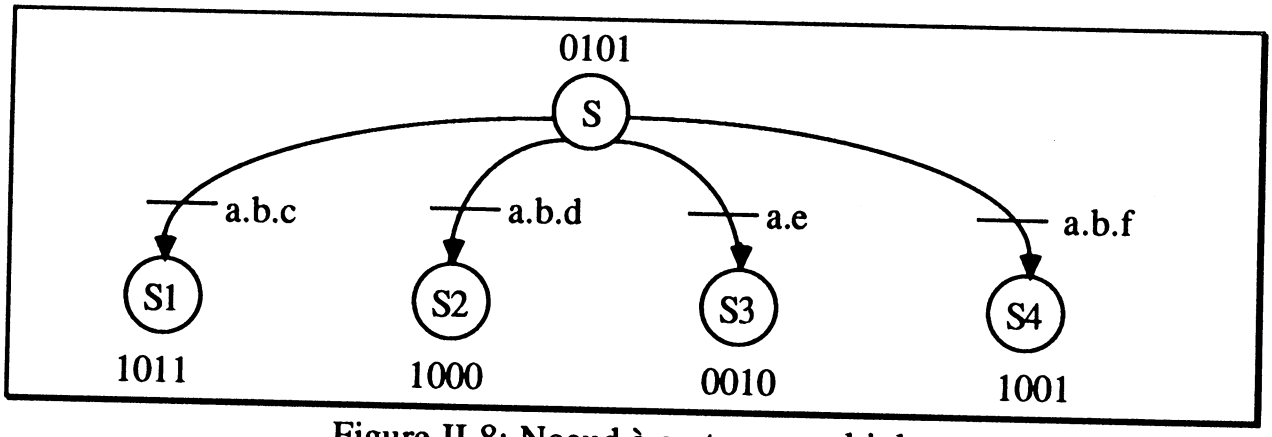

Figure II-8: Noeud à sortance multiple

Le monôme de proximité des arcs est égal à a; la partie de conoyau obtenue est égale à $\overline{y4}.y3.\overline{y2}.y1.a$. Illustrons ceci en écrivant les équations de Y4 et Y2 par exemple.

$$Y4 = \overline{y4}.y3.\overline{y2}.y1.a.b.(c + d + f)$$
$$Y2 = \overline{y4}.y3.\overline{y2}.y1.a.(b.c + e)$$

2.2.2.3. Situation 3: Noeud à entrance multiple

Cette situation est un autre cas particulier de la première situation où les noeuds successeurs sont confondus en un seul noeud s. Soient M le monôme de proximité des noeuds source et C le monôme de proximité des arcs ayant ce noeud comme destination, alors M.C est un conoyau commun pour toute

variable pour laquelle le noeud s est actif. Ceci est illustré dans la figure suivante, où M est égal à $\overline{y4} . \overline{y1}$ et C égal à a.

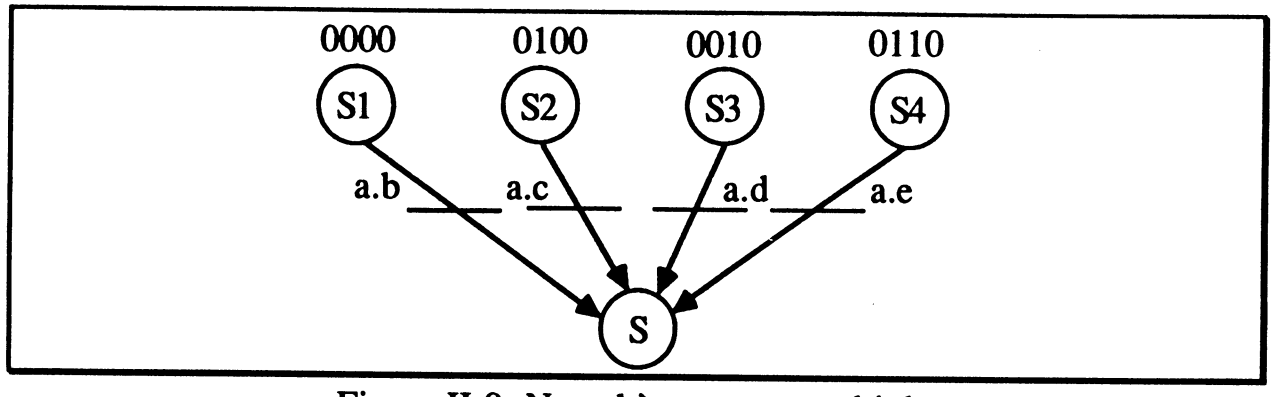

Figure II-9: Noeud à entrance multiple

Toute variable Y_j égale à 1 dans le code de S admet pour sous-expression

$$Y_j = \overline{y4} . \overline{y1} .a.(\overline{y3} . \overline{y2} .b + y3. \overline{y2} .c + \overline{y3} .y2.d + y3.y2.e)$$

2.2.2.4. Situation 4: Arcs actifs pour une sortie d'un automate de Mealy avec entrées intersectantes

Cette situation analogue à la première concerne ici les sorties d'un automate de Mealy au lieu des variables internes. De la même manière ici, si M et C sont les monômes de proximité des noeuds et des arcs actifs pour la sortie o respectivement, alors le monôme M.C est un conoyau dans l'expression de o. Dans l'exemple de la figure II-10, $\overline{y4}$ .a est un noyau de l'expression de o.

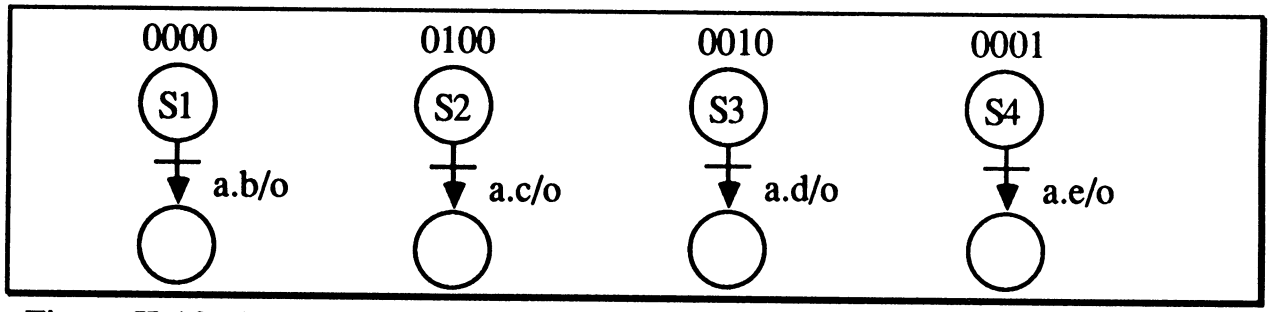

Figure II-10: Arcs actifs pour un automate de Mealy avec entrées intersectantes

2.2.3. Conclusion sur la phase de reconnaissance de situations à factorisation

Les situations énoncées ci-dessus ne sont pas traitées globalement; en effet, le gain obtenu par la factorisation pourra être exprimé pour tout couple de noeuds cités dans les situations précédentes. On parlera donc de gain associé à un

couple de noeuds et on parlera d'attraction; le gain sera proportionnel aux littéraux communs, donc à leur proximité au sens de la distance de Hamming.

Illustrons ceci dans l'exemple de la figure II-11.

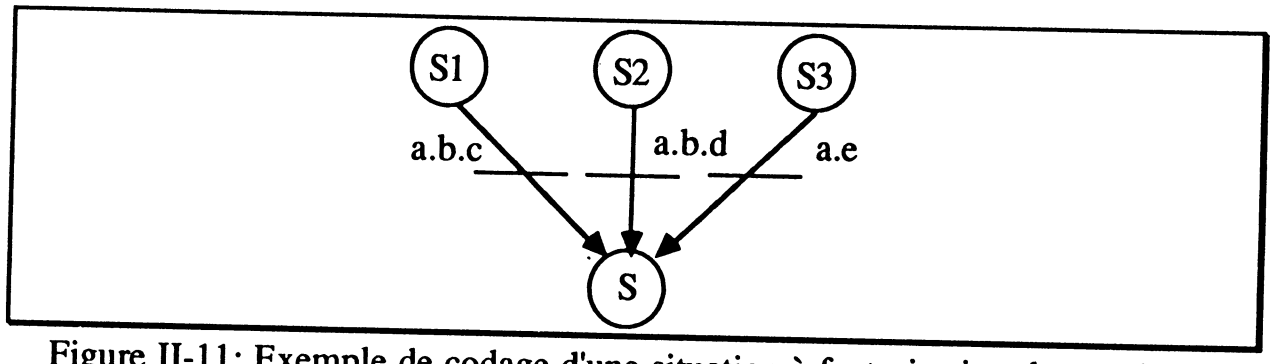

Figure II-11: Exemple de codage d'une situation à factorisation de monômes

On note ici que le couple d'arcs en aval des noeuds s1 et s2 comportent 2 littéraux en commun contrairement aux autres couples d'arcs qui n'en comportent qu'un. Affectons les codes 0000, 0001 et 0111 respectivement à s1, s2 et s3. Alors l'expression d'une variable Y_j égale à 1 dans le code de s pour cette situation est égale à

$$Y_j = \overline{y4}.a.(\overline{y3}.\overline{y2}.b.(\overline{y1}.c + y1.d) + y3.y2.y1.e)$$

Affectons les mêmes codes 0000, 0111 et 0001 respectivement à s1, s2 et s3. On note cette fois qu'on a "éloigné" les codes de s1 et s2. L'expression de Y_j obtenue cette fois est égale à

$$Y_j = \overline{y4}.a.(\overline{y3}.\overline{y2}.(\overline{y1}.b.c + y1.e) + y3.y2.y1.b.d)$$

On note donc que la première expression possède 13 littéraux alors que la deuxième en possède 14. Le littéral en moins dans la première expression provient du fait que le littéral b en commun a pu être mis en facteur dans celle-ci. On en conclut donc qu'il est plus intéressant de rapprocher les codes des états s1 et s2 car les arcs en aval des noeuds associés admettent plus de littéraux en commun.

2.2.4. Pondération par attraction de couples d'éléments

La conclusion précédente nous amène à pondérer les situations par la définition d'une "attraction" pour tout couple d'états. Cette attraction est définie par le nombre de littéraux mis en facteur en rapprochant le code des deux états impliqués. Nous allons montrer dans ce qui suit que cette attraction dépend du

monôme de proximité entre les noeuds impliqués et du monôme de proximité des arcs correspondants.

Notons M le monôme de proximité des noeuds source, M' le monôme de proximité des noeuds successeurs, et C le monôme de proximité des arcs associés. On dénotera le nombre de littéraux d'un monôme m par NbLit(m).

2.2.4.1. Situation 1

Le nombre de littéraux gagnés pour une variable interne égale à 1 dans les codes des deux états successeurs est égale à NbLit(M) + NbLit(C). Ce gain est multiplié par le nombre de variables internes pour lesquelles les noeuds successeurs sont actifs; on peut supposer que la moitié de ces variables invariantes sont égales à 1, soit NbLit(M')/2 variables. Le gain total est donc (NbLit(M) + NbLit(C))*NbLit(M')/2.

2.2.4.2. Situation 2

Le raisonnement est le même ici sauf que le monôme M doit ici être remplacé par MAC(s), le monôme associé au noeud source. Or NbLit(MAC(s)) est égal à N, ce qui donne le gain (N + NbLit(C))*NbLit(M')/2.

2.2.4.3. Situation 3

La variation s'applique ici au noeud successeur. Pour chaque variable interne pour laquelle ce noeud est actif, le gain en littéraux est NbLit(M) + NbLit(C). On suppose ici que N/2 variables sont égales à 1 dans le code de l'état associé à ce noeud. Le gain final est donc (NbLit(M) + NbLit(C))*N/2.

2.2.4.4. Situation 4

Cette situation est analogue à la précédente sauf qu'elle ne concerne qu'une variable de sortie. Le gain n'est donc pas pondéré par N/2 et est pris égal à NbLit(M) + NbLit(C).

L'évaluation des gains présentés dépend des monômes M, M' et C. Or les monômes M et M' dépendent des codes donnés aux états impliqués et ne sont pas connus; c'est pourquoi il ne sera possible de ne donner qu'une évaluation moyenne de ces gains en posant NbLit(M) = NbLit(M') = N/2.

2.3. CONCLUSION GÉNÉRALE SUR LA RECONNAISSANCE DES SITUATIONS

Deux types de situations sont détectées dans un graphe de contrôle. Dans une première phase, des situations à fusion de monômes sont recherchées, et un certain nombre de groupes d'adjacence sont obtenus. Ces groupes d'adjacence sont pondérés et l'immersion d'un maximum de ces groupes d'adjacence est effectuée en affectant chacun de ces groupes à un sous-cube de l'hypercube. Dans une deuxième phase, les situations à factorisation de monômes sont recherchées et les attractions entre couple d'états sont évaluées en fonction des situations détectées. L'affectation des codes aux états se fait en minimisant les distances entre états fortement attirés.

CHAPITRE III

THÉORIE DES CUBES INTERSECTANTS

Dans le chapitre concernant la recherche de situations à fusion de monômes, nous avons vu que les contraintes imposées à de telles situations consistent à affecter les groupes d'adjacence à des sous-cubes ou cubes de l'hypercube. Un groupe d'adjacence donné peut avoir des noeuds en commun avec certains groupes d'adjacence et être disjoint des autres. L'affectation des groupes d'adjacence à des cubes devra respecter les intersections et les disjonctions existant entre ces groupes. Cela implique que les noeuds communs à deux groupes d'adjacence G_1 et G_2 devront se retrouver sur le cube intersection des deux cubes auxquels G_1 et G_2 auront été affectés. De même, si deux groupes d'adjacence sont disjoints, les cubes auxquels ils seront affectés devront être disjoints.

Cette phase d'immersion consistera à affecter les groupes d'adjacence à des cubes un par un dans une approche de type glouton en commençant par les groupes de plus fort gain. La recherche d'un cube pour un groupe d'adjacence se fera en respectant les relations d'intersection et de disjonction avec les cubes auxquels auront été affectés les groupes d'adjacence déjà considérés. Cette recherche est fondée sur la théorie des cubes intersectants que nous allons présenter ici.

On distinguera deux parties principales; la première donnera un certain nombre de définitions et de propriétés sur les cubes ou sous-cubes de l'hypercube et la seconde partie traitera des conditions nécessaires et/ou suffisantes pour la recherche de cubes vérifiant un certain nombre de relations d'intersection et de disjonction avec d'autres cubes.

3.1. THÉORIE DES CUBES: DÉFINITIONS PRÉLIMINAIRES

3.1.1. Rappels sur la théorie des ensembles

L'hypercube étant par la suite représenté par le treillis des sous-ensembles d'un ensemble à N éléments, nous allons rappeler au préalable des définitions et des propriétés classiques de la théorie des ensembles; ces propriétés seront utilisées dans les démonstrations qui seront présentées.

Pour des raisons de simplicité d'écriture on notera un ensemble d'entiers par une chaîne ordonnée de ses éléments; le caractère ordonné n'est ici qu'un artifice d'écriture. Par exemple, {1, 2, 4} sera noté 124.

$\overline{A}$ est l'ensemble complémentaire de A dans un ensemble universel E, c'est-à-dire, $C_E A$.

$A \cap \overline{B}$ dénote l'ensemble des éléments de A n'appartenant pas à B, c'est-à-dire, $A \cap \overline{B}$.

Tout ensemble A peut s'écrire $A = (A \cap B) \cup (A/B)$.

Toute union $A \cup B$ peut s'écrire $A \cup B = A \cup (B/A)$.

3.1.1.1. Relation d'ordre et équivalences

Par définition, $A \subseteq B \Leftrightarrow A \cap B = A$.

$$A \subseteq B \Leftrightarrow A \cap B = A$$
$$\Leftrightarrow A \cup \overline{B} = B$$
$$\Leftrightarrow A \cap \overline{B} = \emptyset$$
$$\Leftrightarrow \overline{B} \subseteq \overline{A}$$

$$A \subseteq B \Rightarrow A \cap C \subseteq B \cap C$$

$$A \subseteq B \text{ et } A' \subseteq B' \Rightarrow A \cup A' \subseteq B \cup B'$$

3.1.1.2. Propriétés sur les cardinalités des ensembles

$\forall A \forall B \; |A \cap B| \leq |A|$

$\forall A \forall B \; A \subseteq B \Leftrightarrow |A \cap B| = |A|$

$\forall A \forall B \; A \cap B = \emptyset \Leftrightarrow |A \cup B| = |A| + |B|$

$\forall A \forall B \; |A/B| = |A| - |A \cap B|$

$\forall A \forall B \; A \subseteq B \Leftrightarrow |A/B| = |A| - |B|$

$\forall A \forall B \; (A \subseteq B \text{ et } |A| = |B|) \Leftrightarrow A = B$

$\forall A \; |\overline{A}| = |E| - |A|$

3.1.2. L'hypercube représenté par le treillis de Boole

Un treillis est un ensemble ordonné (P, <) tel que toute partie de P admette une borne inférieure et une borne supérieure dans P.

En particulier, P admet un élément minimal égal à sa borne inférieure, et un élément maximal égal à sa borne supérieure.

L'hypercube booléen B^N ($B = \{0, 1\}$) muni de la relation d'ordre habituelle entre vecteurs booléens [KUN68] est un treillis d'élément minimal (0, ..., 0) et d'élément maximal (1, ..., 1). La borne supérieure d'un ensemble de vecteurs de B^N est obtenue en faisant la somme (+) de ces vecteurs, et la borne inférieure en faisant le produit bit à bit (.) de ces vecteurs.

L'ensemble des parties de {1, ..., N}, $\mathfrak{P}(\{1, ..., N\})$, muni de la relation d'inclusion entre parties d'éléments est un treillis d'élément minimal Ø et d'élément maximal {1, ..., N}. La borne supérieure d'un ensemble de sous-ensembles de {1, ..., N} est obtenue en faisant l'union ($\cup$) de ces sous-ensembles, et la borne inférieure en faisant l'intersection ($\cap$) de ces sous-ensembles.

Il est connu que $(B^N, +, .)$ est isomorphe à $(\mathfrak{P}(\{1, ..., N\}), \cup, \cap)$

Les représentations du 3-cube données dans la figure III-1 illustrent l'hypercube vu comme un treillis de Boole. Dans la représentation par un treillis de Boole, les parties d'éléments associées aux noeuds du cube peuvent être interprétés comme donnant le numéro des bits égaux à 1 dans les coordonnées binaires. De façon plus générale, en tenant compte des propriétés bien connues de permutations possibles dans le cube ou de changement d'origine, ces parties d'éléments peuvent être interprétés comme donnant l'identification des bits différant par leurs valeurs de ceux d'une origine de référence noté Ø. Ces parties d'éléments seront appelées coordonnées relatives et seront notées $\mathfrak{L}$. On parlera également d'ensembles d'indices.

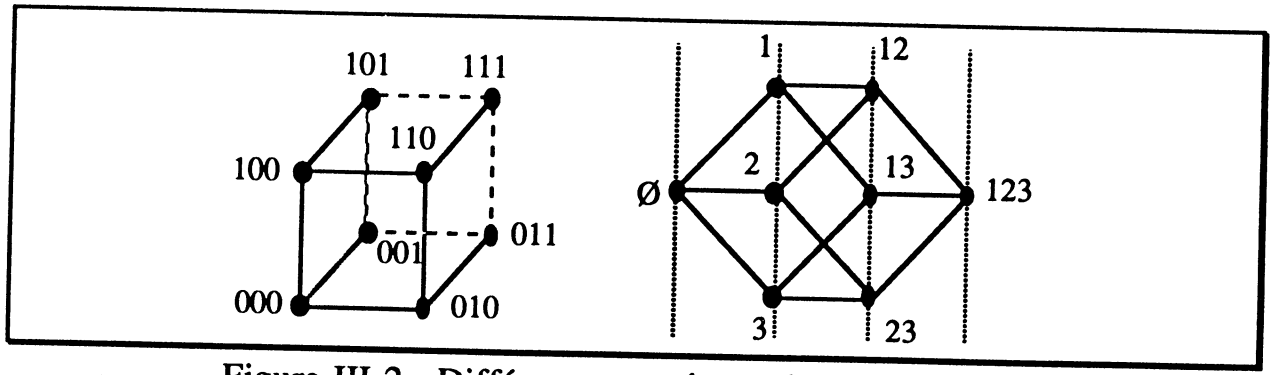

Figure III-2 : Différentes représentations d'un 3-cube

3.1.2.1. Face ou cube de l'hypercube

Un cube, ou une face, est défini par un couple $[\mathfrak{x}_{min}, \mathfrak{x}_{max}]$ avec $\mathfrak{x}_{min} \subseteq \mathfrak{x}_{max}$, et contient tous les sommets $\mathfrak{x}$ tels que $\mathfrak{x}_{min} \subseteq \mathfrak{x} \subseteq \mathfrak{x}_{max}$. Ainsi, un cube est défini de manière unique par ses coordonnées relatives minimale $\mathfrak{x}_{min}$ (qu'on appellera aussi origine) et maximale $\mathfrak{x}_{max}$.

3.1.2.2. Ensemble caractéristique d'un cube

On définit $\mathfrak{x}_C$, l'ensemble caractéristique d'un cube $[\mathfrak{x}_{min}, \mathfrak{x}_{max}]$, par $\mathfrak{x}_C = \mathfrak{x}_{max}/\mathfrak{x}_{min}$. On note ainsi que $\mathfrak{x}_{min} \cap \mathfrak{x}_C = \emptyset$. On peut aussi noter un cube par $[\mathfrak{x}_{min}, \mathfrak{x}_{min} \cup \mathfrak{x}_C]$.

exemple

Dans le 3-cube de la figure III-2, [2, 123], [1, 123] sont des sous-cubes de l'hypercube, [Ø, 123] est l'hypercube lui-même. 13 et 23 sont respectivement les ensembles caractéristiques des deux cubes [2, 123] et [1, 123].

3.1.2.3. Dimension d'un cube

La dimension d'un cube est égale à $|\mathfrak{x}_C|$. Le nombre de sommets contenus dans un cube est alors égal à $2^{|\mathfrak{x}_C|}$.

exemple

Le 2-cube [2, 123] contient 2^2 sommets, notamment 2, 12, 23 et 123.

3.1.3. Intersection de cubes

3.1.3.1. Définition

L'intersection de deux cubes $[\mathfrak{x}_{min}, \mathfrak{x}_{max}]$ et $[\mathfrak{x}'_{min}, \mathfrak{x}'_{max}]$ est égal à l'ensemble des sommets appartenant à la fois aux deux cubes.

3.1.3.2. Propriété 1

L'intersection de deux cubes $[\mathfrak{x}_{min}, \mathfrak{x}_{max}]$ et $[\mathfrak{x}'_{min}, \mathfrak{x}'_{max}]$, si elle existe, est égale à $[\mathfrak{x}_{min} \cup \mathfrak{x}'_{min}, \mathfrak{x}_{max} \cap \mathfrak{x}'_{max}]$. Cette intersection existe si et seulement $[\mathfrak{x}_{min} \cup \mathfrak{x}'_{min}, \mathfrak{x}_{max} \cap \mathfrak{x}'_{max}]$ est un cube, c'est-à-dire, si

$\mathfrak{E}_{min} \cup \mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max} \cap \mathfrak{E}'_{max}$. Cette intersection est illustrée dans la figure III-3.

Preuve

$\mathfrak{E} \in [\mathfrak{E}_{min}, \mathfrak{E}_{max}]$ et $\mathfrak{E} \in [\mathfrak{E}'_{min}, \mathfrak{E}'_{max}]$
$\Leftrightarrow$ $(\mathfrak{E}_{min} \subseteq \mathfrak{E} \subseteq \mathfrak{E}_{max})$ et $(\mathfrak{E}'_{min} \subseteq \mathfrak{E} \subseteq \mathfrak{E}'_{max})$
$\Leftrightarrow$ $\mathfrak{E}_{min} \cup \mathfrak{E}'_{min} \subseteq \mathfrak{E} \subseteq \mathfrak{E}_{max} \cap \mathfrak{E}'_{max}$

cqfd

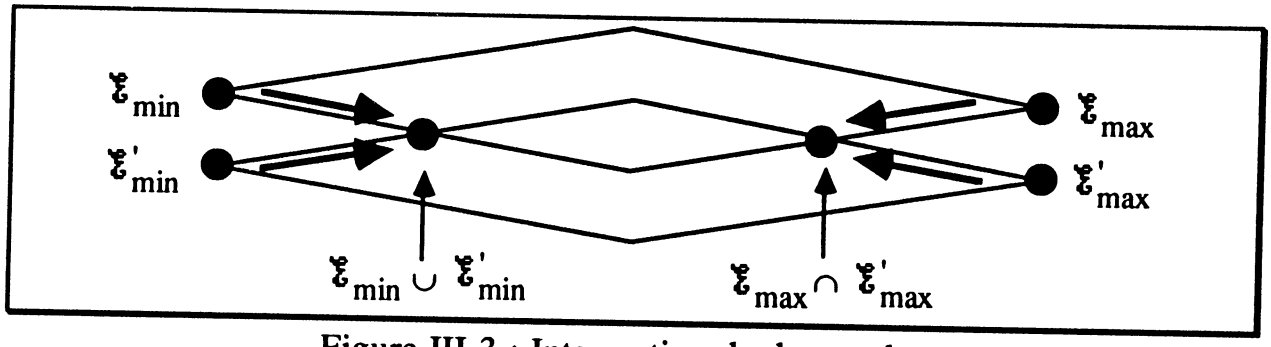

Figure III-3 : Intersection de deux cubes

3.1.3.3. Condition nécessaire et suffisante d'intersection de deux cubes

Deux cubes $[\mathfrak{E}_{min}, \mathfrak{E}_{max}]$ et $[\mathfrak{E}'_{min}, \mathfrak{E}'_{max}]$ ont une intersection si et seulement si $\mathfrak{E}_{min} \subseteq \mathfrak{E}'_{max}$ et $\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max}$.

Preuve:

Pour que l'intersection existe, il faut et il suffit que $\mathfrak{E}_{min} \cup \mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max} \cap \mathfrak{E}'_{max}$, ce qui est équivalent à $\mathfrak{E}_{min} \subseteq \mathfrak{E}'_{max}$ et $\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max}$.

$(\mathfrak{E}_{min} \cup \mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max} \cap \mathfrak{E}'_{max})$
$\Leftrightarrow$ $(\mathfrak{E}_{min} \subseteq \mathfrak{E}_{max} \cap \mathfrak{E}'_{max})$ et $(\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max} \cap \mathfrak{E}'_{max})$
$\Leftrightarrow$ $(\mathfrak{E}_{min} \subseteq \mathfrak{E}'_{max}$ et $\mathfrak{E}_{min} \subseteq \mathfrak{E}_{max})$ et $(\mathfrak{E}'_{min} \subseteq \mathfrak{E}'_{max}$ et $\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max})$

Les propriétés $\mathfrak{E}_{min} \subseteq \mathfrak{E}_{max}$ et $\mathfrak{E}'_{min} \subseteq \mathfrak{E}'max$ sont vérifiées de par la définition même des cubes.

cqfd

3.1.3.4. Propriété 2: Disjonction de deux cubes

Deux cubes $[\mathfrak{E}_{min}, \mathfrak{E}_{max}]$ et $[\mathfrak{E}'_{min}, \mathfrak{E}'_{max}]$ sont disjoints si et seulement si $\mathfrak{E}_{min} \not\subseteq \mathfrak{E}'_{max}$ ou $\mathfrak{E}'_{min} \not\subseteq \mathfrak{E}_{max}$.La preuve est immédiate.

exemple

Les deux 2-cubes [Ø, 12] et [2, 123] ont pour intersection l'arête [2, 12].

Les cubes [3, 123] et [2, 12] n'ont pas d'intersection car 3 n'est pas inclus dans 12.

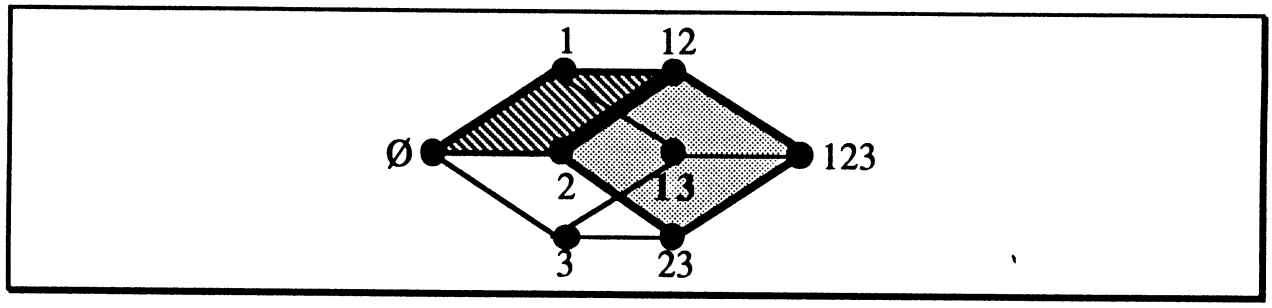

Figure III-4: Exemple d'intersection

3.1.3.5. Propriété 3

La partie caractéristique du cube intersection de deux cubes s'il existe est égale à l'intersection des parties caractéristiques des cubes intersectants.

Preuve:

Soient deux cubes $[\mathfrak{E}_{min}, \mathfrak{E}_{min} \cup \mathfrak{E}_C]$ et $[\mathfrak{E}'_{min}, \mathfrak{E}'_{min} \cup \mathfrak{E}'_C]$; calculons leur intersection. C'est le cube $[\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}, (\mathfrak{E}_{min} \cup \mathfrak{E}_C) \cap (\mathfrak{E}'_{min} \cup \mathfrak{E}'_C)]$.

$$
\begin{aligned}
&(\mathfrak{E}_{min} \cup \mathfrak{E}_C) \cap (\mathfrak{E}'_{min} \cup \mathfrak{E}'_C) \\
&= (\mathfrak{E}_{min} \cap (\mathfrak{E}'_{min} \cup \mathfrak{E}'_C)) \cup (\mathfrak{E}'_{min} \cap (\mathfrak{E}_{min} \cup \mathfrak{E}_C)) \cup (\mathfrak{E}_C \cap \mathfrak{E}'_C) \\
&= (\mathfrak{E}_{min} \cap \mathfrak{E}'_{max}) \cup (\mathfrak{E}'_{min} \cap \mathfrak{E}_{max}) \cup (\mathfrak{E}_C \cap \mathfrak{E}'_C)
\end{aligned}
$$

Or $\mathfrak{E}_{min} \cap \mathfrak{E}'_{max} = \mathfrak{E}_{min}$ car $\mathfrak{E}_{min} \subseteq \mathfrak{E}'_{max}$ est une condition d'intersection

De la même manière, on montre que $\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max}$

On vérifie de plus que $\mathfrak{E}_C \cap \mathfrak{E}'_C$ est disjoint de $\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}$. En effet, $(\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}) \cap \mathfrak{E}_C \cap \mathfrak{E}'_C = (\mathfrak{E}_{min} \cap \mathfrak{E}_C \cap \mathfrak{E}'_C) \cup (\mathfrak{E}'_{min} \cap \mathfrak{E}_C \cap \mathfrak{E}'_C) = Ø$

Le cube intersection peut donc s'écrire $[\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}, (\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}) \cup (\mathfrak{E}_C \cap \mathfrak{E}'_C)]$, $\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}$ étant l'origine du cube et $\mathfrak{E}_C \cap \mathfrak{E}'_C$ son ensemble caractéristique.

exemple

Le cube [2, 12] intersection des deux cubes [Ø, 12] et [2, 123] a pour ensemble caractéristique 1, qui est l'intersection des ensembles caractéristiques 12 et 13 des cubes initiaux.

3.1.3.6. Corollaire

La dimension du cube intersection de deux cubes $[\mathfrak{E}_{min}, \mathfrak{E}_{min} \cup \mathfrak{E}_C]$ et $[\mathfrak{E}'_{min}, \mathfrak{E}'_{min} \cup \mathfrak{E}'_C]$ est égale à $|\mathfrak{E}_C \cap \mathfrak{E}'_C|$.

3.1.4. Inclusion de cubes

3.1.4.1. Définition

Le cube $K = [\mathfrak{E}_{min}, \mathfrak{E}_{max}]$ est inclus dans le cube $K' = [\mathfrak{E}'_{min}, \mathfrak{E}'_{max}]$ si et seulement si l'intersection des deux cubes K.K' est égal à K.

3.1.4.2. Condition nécessaire et suffisante d'inclusion

Le cube $K = [\mathfrak{E}_{min}, \mathfrak{E}_{max}]$ est inclus dans le cube $K' = [\mathfrak{E}'_{min}, \mathfrak{E}'_{max}]$ si et seulement si $\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{min}$ et $\mathfrak{E}_{max} \subseteq \mathfrak{E}'_{max}$. On notera cette inclusion $K \subseteq K'$.

Preuve:

Le cube intersection K.K' est égal à $[\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}, \mathfrak{E}_{max} \cap \mathfrak{E}'_{max}]$.
Donc, $(K \subseteq K') \Leftrightarrow (\mathfrak{E}_{min} \cup \mathfrak{E}'_{min} = \mathfrak{E}_{min}$ et $\mathfrak{E}_{max} \cap \mathfrak{E}'_{max} = \mathfrak{E}_{max})$.
Or, $(\mathfrak{E}_{min} \cup \mathfrak{E}'_{min} = \mathfrak{E}_{min}) \Leftrightarrow (\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{min})$
et $(\mathfrak{E}_{max} \cap \mathfrak{E}'_{max} = \mathfrak{E}_{max}) \Leftrightarrow (\mathfrak{E}_{max} \subseteq \mathfrak{E}'_{max})$.

cqfd

Dans ce cas-ci, on a aussi la relation $\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{min} \subseteq \mathfrak{E}_{max} \subseteq \mathfrak{E}'_{max}$. On peut aussi énoncer la condition nécessaire et suffisante $\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{min}$ et $\mathfrak{E}_C \subseteq \mathfrak{E}'_C$.

Preuve

Montrons aussi que $[(\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{min})$ et $(\mathfrak{E}_{max} \subseteq \mathfrak{E}'_{max})]$
$\Leftrightarrow [(\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{min})$ et $(\mathfrak{E}_C \subseteq \mathfrak{E}'_C)]$

$\Rightarrow$:

$(\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{min})$ et $(\mathfrak{E}_{max} \subseteq \mathfrak{E}'_{max})$
$\Rightarrow \quad \mathfrak{E}_C = \mathfrak{E}_{max}/\mathfrak{E}_{min} \subseteq \mathfrak{E}'_{max}/\mathfrak{E}_{min} \subseteq \mathfrak{E}'_{max}/\mathfrak{E}'_{min} = \mathfrak{E}'_C$
Donc $(\mathfrak{E}_C \subseteq \mathfrak{E}'_C)$

$\Leftarrow$:

$(\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{min})$ et $(\mathfrak{E}_C \subseteq \mathfrak{E}'_C) \Rightarrow \mathfrak{E}_{max} = \mathfrak{E}_{min} \cup \mathfrak{E}_C \subseteq \mathfrak{E}_{min} \cup \mathfrak{E}'_C$
Or $\mathfrak{E}_{min} \subseteq \mathfrak{E}'_{max} \Rightarrow \mathfrak{E}_{min} \cup \mathfrak{E}'_C \subseteq \mathfrak{E}'_{max} \cup \mathfrak{E}'_C = \mathfrak{E}'_{max}$
Donc $(\mathfrak{E}_{max} \subseteq \mathfrak{E}'_{max})$

cqfd

Les propriétés qui ont été énoncées jusqu'ici vont maintenant nous permettre de résoudre un ensemble de problèmes liés à la recherche de cubes vérifiant un certain nombre de conditions de dimensions, d'intersections et de disjonctions.

3.2. RECHERCHE DE CUBES RESPECTANT CERTAINES CONTRAINTES

Dans l'introduction, on a mentionné le fait qu'un groupe d'adjacence doit être affecté à un cube. Ce cube doit vérifier les relations d'intersection et de disjonction avec les cubes auxquels sont affectés les groupes d'adjacence déjà immergés dans une approche progressive de type glouton. Cette partie a pour but de donner des conditions nécessaires et/ou suffisantes pour la recherche de tels cubes, ceci afin de pouvoir définir des algorithmes rapides de recherche de cubes.

Trois problèmes principaux vont être soulevés. Le premier concerne la recherche d'un cube ayant une intersection de dimension fixée avec un cube

donné. Le deuxième tentera de généraliser le premier problème; il concerne la recherche d'un cube ayant des intersections avec un ensemble de cubes connus et ceci pour des dimensions d'intersection connues. Le dernier problème concerne la recherche d'un cube disjoint d'un cube donné.

3.2.1. Recherche d'un cube de dimension fixée ayant une intersection avec un cube donné

Le problème posé ici est de trouver un cube $[\mathfrak{E}'_{min}, \mathfrak{E}'_{max}]$ de dimension k' intersectant avec un cube donné $[\mathfrak{E}_{min}, \mathfrak{E}_{max}]$ de dimension k. Montrons la construction de ce cube.

Pour ce cube, on recherche son couple de coordonnées relatives $[\mathfrak{E}'_{min}, \mathfrak{E}'_{max}]$ en distinguant trois sous-ensembles d'indices de $\mathfrak{E}'_{max}$ qui peut alors être réécrit $\mathfrak{E}'_{max1} \cup \mathfrak{E}'_{max2} \cup \mathfrak{E}'_{max3}$. Ces sous-ensembles sont détaillés dans ce qui suit.

$\mathfrak{E}'_{max1}$ est l'ensemble d'indices hérités localement à l'intérieur du cube; il est égal à $\mathfrak{E}'_{min}$, pour $\mathfrak{E}'_{min}$ choisi convenablement. Cela provient du fait que $\mathfrak{E}'_{max} \supseteq \mathfrak{E}'_{min}$ pour que $[\mathfrak{E}'_{min}, \mathfrak{E}'_{max}]$ soit un cube.

$\mathfrak{E}'_{max2}$ est l'ensemble d'indices hérités de l'autre cube; il est égal à $\mathfrak{E}'_{min}/\mathfrak{E}_{min}$. En effet, une des conditions d'intersection impose que $\mathfrak{E}'_{max} \supseteq \mathfrak{E}_{min}$. Donc des indices de $\mathfrak{E}_{min}$ autres que ceux de $\mathfrak{E}'_{min}$ sont rajoutés à $\mathfrak{E}'_{max}$. $\mathfrak{E}'_{max2}$ est ainsi une partie de l'ensemble caractéristique du cube recherché.

$\mathfrak{E}'_{max3}$, l'ensemble d'indices restants de $\mathfrak{E}'_{max}$, contient $(k' - |\mathfrak{E}'_{max2}|)$ indices. Pour des raisons d'existence et d'intersection, $\mathfrak{E}'_{max}$ contient l'ensemble $\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}$; les indices de ces ensembles sont donc imposés. Une fois $\mathfrak{E}'_{max1}$ et $\mathfrak{E}'_{max2}$ construits, il convient de compléter l'ensemble caractéristique par $(k' - |\mathfrak{E}'_{max2}|)$ indices n'apparaissant pas dans $\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}$.

exemple

Dans un hypercube de dimension 5, on cherche un 2-cube $[\mathfrak{E}'_{min}, \mathfrak{E}'_{max}]$ intersectant avec [12, 1234]. Une solution est, par exemple, [23, 1235]. Ici,

$\mathfrak{E}'_{max1} = \mathfrak{E}'_{min} = 23$; notons que $\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max} = 1234$. Ceci impose que $\mathfrak{E}'_{max2} = 1$. Pour constituer $\mathfrak{E}'_{max3}$, il reste 1 indice à trouver parmi 45; on choisit 5 par exemple.

3.2.1.1. Condition de place suffisante pour la construction du cube

On choisit un ensemble $\mathfrak{E}'_{min}$ vérifiant $\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max}$. L'ensemble $\mathfrak{E}'_{max2}$ de cardinal $| \mathfrak{E}_{min}/\mathfrak{E}'_{min} |$ est obtenu. Pour que la construction du deuxième cube soit possible, il est nécessaire que la dimension de l'héritage, c.a.d $| \mathfrak{E}_{min}/\mathfrak{E}'_{min} |$, ne dépasse pas k' d'où la condition nécessaire $| \mathfrak{E}_{min}/\mathfrak{E}'_{min} | \leq k'$. $| \mathfrak{E}_{min}/\mathfrak{E}'_{min} |$ s'écrit aussi $| \mathfrak{E}_{min} | - | \mathfrak{E}_{min} \cap \mathfrak{E}'_{min} |$, ce qui donne $| \mathfrak{E}_{min} \cap \mathfrak{E}'_{min} | \geq | \mathfrak{E}_{min} | - k'$.

3.2.1.2. Condition d'existence d'un nombre suffisant d'indices pour compléter le cube

Une fois cette condition vérifiée, il faudra fournir les $k' - | \mathfrak{E}_{min}/\mathfrak{E}'_{min} |$ indices nécessaires pour compléter le cube. Ces indices n'appartiennent forcément ni à $\mathfrak{E}_{min}$ ni à $\mathfrak{E}'_{min}$. Ils seront donc pris dans $(\mathfrak{E}_{min} \cup \mathfrak{E}'_{min})$; le nombre d'indices disponibles est donc égal à $| \overline{(\mathfrak{E}_{min} \cup \mathfrak{E}'_{min})} |$, soit $N - | \mathfrak{E}_{min} \cup \mathfrak{E}'_{min} |$, ou encore $N - | \mathfrak{E}'_{min} | - | \mathfrak{E}_{min}/\mathfrak{E}'_{min} |$. On en déduit donc que $k' - | \mathfrak{E}_{min}/\mathfrak{E}'_{min} | \leq N - | \mathfrak{E}'_{min} | - | \mathfrak{E}_{min}/\mathfrak{E}'_{min} |$, ou encore $| \mathfrak{E}'_{min} | \leq N - k'$.

Les conditions nécessaires et suffisantes d'existence d'un k'-cube $[\mathfrak{E}'_{min}, \mathfrak{E}'_{max}]$ intersectant avec un k-cube $[\mathfrak{E}_{min}, \mathfrak{E}_{max}]$ donné sont:

$$(| \mathfrak{E}_{min} \cap \mathfrak{E}'_{min} | \geq | \mathfrak{E}_{min} | - k') \wedge (| \mathfrak{E}'_{min} | \leq N - k')$$

Autrement dit, l'intersection des deux cubes n'est possible que si l'origine $\mathfrak{E}'_{min}$ du deuxième cube, minorant de $\mathfrak{E}_{max}$, admet au plus (N - k') indices, et comporte au moins $(| \mathfrak{E}_{min} | - k')$ indices en commun avec $\mathfrak{E}_{min}$.

exemple

Dans un hypercube de dimension 5, on cherche un 2-cube $[\mathfrak{E}'_{min}, \mathfrak{E}'_{max}]$ intersectant avec [123, 1234]. On cherche $\mathfrak{E}'_{min}$ qui a au plus 3 indices et a au moins 1 indice en commun avec 123. Une solution pour $\mathfrak{E}'_{min}$ pourrait être 124, par exemple, ce qui donnerait le cube [124, 12345].

3.2.2. Recherche d'un cube de dimension fixée ayant une intersection de dimension fixée avec un cube donné

Le problème posé ici est le suivant : étant donné un k-cube $K = [\mathfrak{E}_{min}, \mathfrak{E}_{max}]$, déterminer un k'-cube $K' = [\mathfrak{E}'_{min}, \mathfrak{E}'_{max}]$ ayant une intersection avec K, qui soit un cube K" de dimension k".

De la même manière que précédemment, décomposons le deuxième cube. $\mathfrak{E}'_{max1}$ et $\mathfrak{E}'_{max2}$ sont les mêmes ensembles que précédemment; par contre une décomposition plus fine pour la partie restante de $\mathfrak{E}'_{max}$ est nécessaire. Les deux sous-ensembles sont les suivants:

$\mathfrak{E}'_{max3}$ est l'ensemble d'indices caractéristiques de l'intersection et est donc composé de k" indices. On a vu que cet ensemble est égal à $\mathfrak{E}_C \cap \mathfrak{E}'_C$. En construisant $\mathfrak{E}'_C$, on doit veiller à ce que $\mathfrak{E}'_C$ soit disjoint de $\mathfrak{E}'_{min}$, ce qui impose que les indices composant l'intersection sont des indices de $\mathfrak{E}_C$ autres que ceux apparaissant dans $\mathfrak{E}'_{min}$. Il convient alors de choisir k" indices de $\mathfrak{E}_C / \mathfrak{E}'_{min}$ pour composer cette intersection.

$\mathfrak{E}'_{max4}$ est l'ensemble d'indices restants de $\mathfrak{E}'_{max}$ ne participant pas à l'intersection; il contient $(k' - |\mathfrak{E}'_{max2}| - k")$ indices. Ces indices n'appartiennent ni à $\mathfrak{E}_{min}$, ni à $\mathfrak{E}'_{min}$, et ne peuvent plus être choisis dans $\mathfrak{E}_C$ car l'intersection a déjà été constituée lors de la construction de $\mathfrak{E}'_{max3}$. $\mathfrak{E}'_{max4}$ est donc un sous-ensemble de $\overline{\mathfrak{E}_{min} \cup \mathfrak{E}'_{min} \cup \mathfrak{E}_C}$. Or $\mathfrak{E}_{min} \cup \mathfrak{E}_C \cup \mathfrak{E}'_{min}$ est égal à $\mathfrak{E}_{max} \cup \mathfrak{E}'_{min}$, ce qui peut se simplifier en $\mathfrak{E}_{max}$ car $\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max}$. $\mathfrak{E}'_{max4}$ est donc un sous-ensemble de $\overline{\mathfrak{E}_{max}}$.

La construction de $\mathfrak{E}'_{max}$ est résumée dans la figure suivante (figure III-5):

$\mathfrak{E}'_{max1}$	$\mathfrak{E}'_{max2}$	$\mathfrak{E}'_{max3}$	$\mathfrak{E}'_{max4}$
$\mathfrak{E}'_{min}$	$\mathfrak{E}_{min}/\mathfrak{E}'_{min}$	k" indices de $\mathfrak{E}_C/\mathfrak{E}'_{min}$	k'-$\lvert\mathfrak{E}'_{max2}\rvert$-k" indices de $\overline{\mathfrak{E}_{max}}$

Figure III-5 : Structure du cube intersectant

exemple

Supposons que dans un 5-cube, on veuille trouver un 3-cube intersectant avec le 2-cube [12, 1234] de telle sorte que l'intersection soit un 1-cube. Montrons la construction d'un tel cube.

On doit choisir un minorant de 1234 pour $\mathfrak{E}'_{min}$ pour qu'une intersection existe ($\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max}$). Prenons par exemple $\mathfrak{E}'_{max1} = \mathfrak{E}'_{min} = 14$.

Pour les mêmes raisons d'intersection, on a $\mathfrak{E}_{min} \subseteq \mathfrak{E}'_{max}$, donc $\mathfrak{E}'_{max} \supseteq 12$. L'ensemble d'indices additionnel qui provient de $\mathfrak{E}_{min}$ est $\mathfrak{E}'_{max2} = \mathfrak{E}_{min}/\mathfrak{E}'_{min}$, c'est-à-dire 2.

On veut maintenant obtenir une intersection de dimension 1. Les indices de $\mathfrak{E}'_C$ constituant l'intersection ne peuvent provenir que de $\mathfrak{E}_C$, à savoir 34. Or l'indice 4 a déjà été utilisé dans $\mathfrak{E}'_{min}$. L'ensemble des indices disponibles est donc égal à $\mathfrak{E}_C/\mathfrak{E}'_{min}$, c'est-à-dire 3. Il faut donc en choisir 1 élément; ceci donne $\mathfrak{E}'_{max3} = 3$.

A ce point, $\mathfrak{E}'_C$ est partiellement constitué et contient les 2 indices 23. Comme on veut un 3-cube, il reste à fournir encore un indice. Cet indice ne peut provenir ni de $\mathfrak{E}_{min}$ ni de $\mathfrak{E}'_{min}$ qui ont déjà été utilisés. Aucun autre indice de $\mathfrak{E}_C$ ne peut être utilisé car l'intersection est déjà constituée. Comme montrée précédemment, l'union de ces trois ensembles est égal à $\mathfrak{E}_{max}$; l'ensemble des indices disponibles est donc 5. Il faut en choisir 1 indice; donc $\mathfrak{E}'_{max4} = 5$.

Le cube obtenu est donc [14, 12345] et l'intersection qui en découle est [124, 1234]. La construction de ce cube est résumée dans la figure suivante.

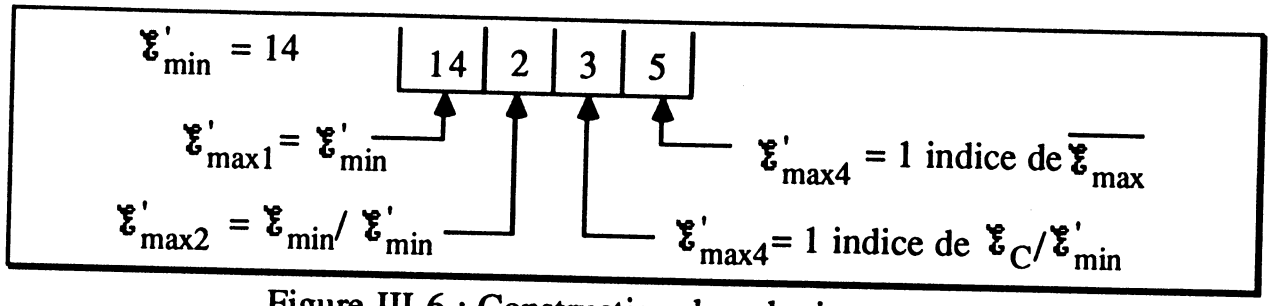

Figure III-6 : Construction du cube intersectant

3.2.2.1. Condition d'existence d'un nombre suffisant d'indices pour construire l'intersection

Pour construire l'intersection et constituer le sous-ensemble $\mathfrak{E}'_{max3}$, il faut pouvoir choisir k" indices à partir de $\mathfrak{E}_C/\mathfrak{E}'_{min}$; ceci impose donc que $| \mathfrak{E}_C/\mathfrak{E}'_{min} | \geq k"$.

Or $\mathfrak{E}_C$ peut aussi s'écrire $\mathfrak{E}_C = (\mathfrak{E}_C/\mathfrak{E}'_{min}) \cup (\mathfrak{E}_C \cap \mathfrak{E}'_{min})$. Ces ensembles étant disjoints, on a aussi $| \mathfrak{E}_C/\mathfrak{E}'_{min} | = | \mathfrak{E}_C | - | \mathfrak{E}_C \cap \mathfrak{E}'_{min} |$

On obtient donc la condition nécessaire

$$| \mathfrak{E}_C \cap \mathfrak{E}'_{min} | \leq | \mathfrak{E}_C | - k" = k - k"$$

Cette condition impose que le nombre d'indices que $\mathfrak{E}'_{min}$ a au plus (k - k") indices en commun avec $\mathfrak{E}_C$.

3.2.2.2. Condition de place suffisante pour les indices d'intersection

La condition précédente permet de savoir s'il y a suffisamment d'indices pour construire l'intersection. Néanmoins, pour garantir une intersection de dimension k", il faut que k" soit plus petit que la place restante après la constitution de $\mathfrak{E}'_{max2}$, soit $k" \leq (k' - | \mathfrak{E}'_{max2} |)$.

Or $| \mathfrak{E}'_{max2} | = | \mathfrak{E}_{min}/\mathfrak{E}'_{min} | = | \mathfrak{E}_{min} | - | \mathfrak{E}'_{min} \cap \mathfrak{E}_{min} |$.

Donc $k" \leq k' - | \mathfrak{E}'_{max2} | \Leftrightarrow k" \leq (k' - | \mathfrak{E}_{min} | + | \mathfrak{E}'_{min} \cap \mathfrak{E}_{min} |)$.

Ce qui donne comme condition nécessaire

$$| \mathfrak{E}'_{min} \cap \mathfrak{E}_{min} | \geq k" - k' + | \mathfrak{E}_{min} |$$

3.2.2.3. Condition sur la possibilité de compléter le cube

Si les conditions précédentes sont satisfaites, on est sûr de pouvoir construire l'intersection. Une fois cela fait, il se peut que le cube construit à ce point ne soit pas un k'-cube; il faut donc constituer $\mathfrak{E}'_{max4}$ avec $k' - |\mathfrak{E}'_{max2}| - k''$ indices pris de $\overline{\mathfrak{E}_{max}}$. Pour que cela soit possible, il faut que $k' - |\mathfrak{E}'_{max2}| - k'' \leq |\overline{\mathfrak{E}_{max}}|$.

D'une part, $|\overline{\mathfrak{E}_{max}}| = N - |\mathfrak{E}_{max}|$, et d'autre part $|\mathfrak{E}'_{max2}| = |\mathfrak{E}_{min}| - |\mathfrak{E}'_{min} \cap \mathfrak{E}_{min}|$ amènent à la condition $k' - |\mathfrak{E}_{min}| + |\mathfrak{E}'_{min} \cap \mathfrak{E}_{min}| - k'' \leq N - |\mathfrak{E}_{max}|$, c'est-à-dire, $|\mathfrak{E}'_{min} \cap \mathfrak{E}_{min}| \leq N - k' + k'' + |\mathfrak{E}_{min}| - |\mathfrak{E}_{max}|$. En utilisant le fait que $\mathfrak{E}_{max} = \mathfrak{E}_{min} \cup \mathfrak{E}_C$, on obtient $k = |\mathfrak{E}_C| = |\mathfrak{E}_{max}| - |\mathfrak{E}_{min}|$.

On en déduit la condition nécessaire suivante

$$|\mathfrak{E}'_{min} \cap \mathfrak{E}_{min}| \leq N - k' + k'' - k$$

Les deux conditions précédentes donnent l'intervalle des valeurs pour les nombres d'indices de $\mathfrak{E}'_{min}$ qui peuvent être choisis dans $\mathfrak{E}_{min}$.

3.2.2.4. Construction du cube recherché

Constituer $\mathfrak{E}'_{max1}$ en choisissant $\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max}$ vérifiant les trois conditions précitées.

Construire $\mathfrak{E}'_{max2}$ en calculant $\mathfrak{E}_{min}/\mathfrak{E}'_{min}$.

Choisir k'' indices de $\mathfrak{E}_C/\mathfrak{E}'_{min}$ pour effectuer l'intersection.

Compléter le k'-cube avec des indices pris dans $\overline{\mathfrak{E}_{max}}$.

Les conditions nécessaires précédentes garantissent de pouvoir construire le cube; ce sont donc des conditions suffisantes d'existence de solutions.

3.2.3. Cas particulier de l'inclusion

On va montrer ici que la recherche d'un k'-cube K' inclus dans un k-cube K donné se ramène à chercher un k'-cube intersectant avec le k-cube K générant un cube intersection de dimension k'.

Ceci permettra de ramener le cas de l'inclusion au cas plus général de l'intersection

Preuve

Supposons $K' \subseteq K$.

$\subseteq K \Leftrightarrow K \cap K' = K' \Rightarrow$ dimension($K \cap K'$) = dimension(K')

est donc bien un cube admettant une intersection de dimension k' avec K.

Inversement, si K et K' ont un k'-cube pour intersection, alors $| \mathfrak{E}_C \cap \mathfrak{E}'_C | = k'$

$| \mathfrak{E}'_C | = k' \Leftrightarrow | \mathfrak{E}_C \cap \mathfrak{E}'_C | = | \mathfrak{E}'_C | \Leftrightarrow \mathfrak{E}'_C \subseteq \mathfrak{E}_C$

Montrons $\mathfrak{E}_{min} \subseteq \mathfrak{E}'_{min}$; pour des raisons d'intersection on a $\mathfrak{E}_{min} \subseteq \mathfrak{E}'_{max} = \mathfrak{E}'_{min} \cup \mathfrak{E}'_C$

$\mathfrak{E}_{min} \subseteq \mathfrak{E}'_{min} \cup \mathfrak{E}'_C \Rightarrow \underline{\mathfrak{E}_{min}}/\underline{\mathfrak{E}'_C} \subseteq \mathfrak{E}'_{min}$

Comme ($\mathfrak{E}'_C \subseteq \mathfrak{E}_C \Leftrightarrow \underline{\mathfrak{E}_C} \subseteq \underline{\mathfrak{E}'_C}$), $\mathfrak{E}_{min}/\mathfrak{E}_C \subseteq \mathfrak{E}_{min}/\mathfrak{E}'_C$ et donc $\mathfrak{E}_{min}/\mathfrak{E}_C \subseteq \mathfrak{E}'_{min}$

Mais $\mathfrak{E}_{min}/\mathfrak{E}_C = \mathfrak{E}_{min}$, ce qui amène à $\mathfrak{E}_{min} \subseteq \mathfrak{E}'_{min}$

On a donc K et K' qui intersectent et $\mathfrak{E}_{min} \subseteq \mathfrak{E}'_{min}$ et $\mathfrak{E}'_C \subseteq \mathfrak{E}_C$, ceci entraîne $K' \subseteq K$

cqfd

3.2.4. Recherche d'un cube de dimension fixée ayant des intersections de dimensions fixées avec un ensemble de cubes donnés

Le problème posé ici est de trouver un k-cube $K = [\mathfrak{E}_{min}, \mathfrak{E}_{max}]$ intersectant avec un ensemble de cubes $\{K_1, ..., K_n\}$ de dimensions $\{k_1, ..., k_n\}$ fixées, et tel que les dimensions $\{k'_1, ..., k'_n\}$ des intersections avec ces cubes soient fixées.

On ne propose pas de conditions nécessaires et suffisantes pour trouver un tel cube, mais on donnera un ensemble de conditions nécessaires permettant de filtrer au maximum l'ensemble des solutions possibles.

Pour bien comprendre ces conditions nécessaires, on prendra l'exemple suivant et on montrera la conséquence de l'application de chacune de ces conditions.

exemple

Soient K_1= [123, 12345] et K_2 = [146, 13456]. Dans un hypercube de dimension 7, on cherche un 3-cube intersectant avec K_1 et K_2, de telle sorte que les intersections soient respectivement de dimensions $k_1' = 0$ et $k_2' = 1$.

3.2.4.1. Condition nécessaire et suffisante pour qu'un cube admette des intersections avec un ensemble de cubes donnés

Soit $\{K_1, ..., K_n\}$ un ensemble de cubes donnés avec $K_i = [\mathfrak{E}_{min_i}, \mathfrak{E}_{max_i}]$ pour i = 1 à n. Alors, $K = [\mathfrak{E}_{min}, \mathfrak{E}_{max}]$ intersecte avec chacun des cubes si et seulement si $\mathfrak{E}_{min} \subseteq \bigcap_{i=1}^{n} \mathfrak{E}_{max_i}$ et $\mathfrak{E}_{max} \supseteq \bigcup_{i=1}^{n} \mathfrak{E}_{min_i}$. On notera BSm la borne supérieure $\bigcap_{i=1}^{n} \mathfrak{E}_{max_i}$ (de l'ensemble des valeurs possibles) de $\mathfrak{E}_{min}$ et BIM la borne inférieure $\bigcup_{i=1}^{n} \mathfrak{E}_{min_i}$ (de l'ensemble des valeurs possibles) de $\mathfrak{E}_{max}$.

Preuve

Pour que K intersecte avec $\{K_1, ..., K_n\}$, il faut et il suffit que la condition d'intersection avec chaque K_i soit vérifiée, c'est-à-dire,

$\forall i=1 \text{ à } n \ (\mathfrak{E}_{min} \subseteq \mathfrak{E}_{max_i} \text{ et } \mathfrak{E}_{max} \supseteq \mathfrak{E}_{min_i})$

Or $(\forall i=1 \text{ à } n \ \mathfrak{E}_{min} \subseteq \mathfrak{E}_{max_i} \Leftrightarrow \mathfrak{E}_{min} \subseteq \bigcap_{i=1}^{n})$

et $(\forall i=1 \text{ à } n \ \mathfrak{E}_{max} \supseteq \mathfrak{E}_{min_i} \Leftrightarrow \mathfrak{E}_{max} \supseteq \bigcup_{i=1}^{n} \mathfrak{E}_{min_i})$.

cqfd

Un cube $[\mathfrak{E}_{min}, \mathfrak{E}_{max}]$ solution du problème posé vérifie donc $\mathfrak{E}_{min} \subseteq$ BSm = 1345 et $\mathfrak{E}_{max} \supseteq$ BIM = 12346.

3.2.4.2. Corollaire: Dimension minimale d'un tel cube

La dimension minimale du cube K = $[\mathfrak{E}_{min}, \mathfrak{E}_{max}]$ intersectant avec un ensemble de cubes $\{K_1, ..., K_n\}$ est égale à | BIM/BSm |.

Preuve

$\mathfrak{E}_C$, par définition, est égale à $\mathfrak{E}_{max}/\mathfrak{E}_{min}$.

Or $\mathfrak{E}_{min} \subseteq$ BSm $\Rightarrow$ $\overline{\mathfrak{E}_{min}} \supseteq \overline{BSm}$. Donc, comme $\mathfrak{E}_{max} \supseteq$ BIM, $\mathfrak{E}_C$ vérifie $\mathfrak{E}_C \supseteq$ BIM/BSm, d'où la dimension du cube $| \mathfrak{E}_C | \geq |$ BIM/BSm |.

cqfd

3.2.4.3. Condition nécessaire concernant la dimension minimale du cube recherché

Le corollaire précédent nous précise qu'une dimension minimale est imposée au cube recherché. Comme ce cube doit être un k-cube, il faut que | BIM/BSm | $\leq$ k.

exemple

La dimension minimale imposée est égale à | 12346/1345 | = | 26 | = 2. On peut continuer la recherche d'un 3-cube.

Pour bien suivre l'exemple, on indiquera l'ensemble des valeurs possibles de $\mathfrak{E}_{min}$, noté EO, après application de chaque condition nécessaire. Notons que tout ensemble possible pour $\mathfrak{E}_{min}$ est un sous-ensemble de 1345.

Au départ, EO = {Ø, 1, 3, 4, 5, 13, 14, 15, 34, 35, 45, 134, 135, 145, 345, 1345}.

3.2.4.4. Condition nécessaire dûe à l'héritage de l'ensemble des cubes

Pour construire le cube recherché, il faudra choisir $\mathfrak{E}_{min} \subseteq$ BSm. L'intersection de ce cube avec les autres cubes imposant que $\mathfrak{E}_{max} \supseteq$ BIM, un ensemble d'indices $\mathfrak{E}_H$ égal à BIM/$\mathfrak{E}_{min}$ sont hérités des origines de l'ensemble

des cubes. Pour que la construction du k-cube soit possible, il faut que $| BIM/\mathfrak{E}_{min} | \leq k$. La condition nécessaire est donc

$$| BIM \cap \mathfrak{E}_{min} | \geq | BIM | - k$$

Notons aussi que, comme $\mathfrak{E}_{min} \subseteq BSm$, on a $\mathfrak{E}_{min} \cap BSm = \mathfrak{E}_{min}$. On peut donc réécrire la condition précédente par

$$| (BIM \cap BSm) \cap \mathfrak{E}_{min} | \geq | BIM | - k$$

exemple

La condition impose de trouver $\mathfrak{E}_{min}$ vérifiant $| 134 \cap \mathfrak{E}_{min} | \geq 5 - 3 = 2$. EO devient {13, 14, 34, 134, 135, 145, 345, 1345}. Voyons, par exemple, pourquoi $\mathfrak{E}_{min} = 15$ ne convient pas. Le cube partiel obtenu est [15, 123456] qui est un 4-cube, ce qui est trop grand. On entend par cube partiel le cube dont la coordonnée relative maximale est égale à $\mathfrak{E}_{min} \cup \mathfrak{E}_H$, c'est-à-dire, $[\mathfrak{E}_{min}, \mathfrak{E}_{min} \cup \mathfrak{E}_H]$.

3.2.4.5. Condition nécessaire dûe aux intersections héritées

L'ensemble d'indices hérités des autres cubes $BIM/\mathfrak{E}_{min}$ est un sous-ensemble de $\mathfrak{E}_C$. Ce sous-ensemble impose donc une intersection partielle avec chaque cube K_i, égal à $(BIM/\mathfrak{E}_{min}) \cap \mathfrak{E}_{C_i}$. Or l'intersection voulue avec chaque cube K_i est de dimension k_i'. Pour que ces dimensions soient respectées, il faut que la condition suivante soit respectée: $\forall i=1$ à n $| (BIM/\mathfrak{E}_{min}) \cap \mathfrak{E}_{C_i} | \leq k_i'$. En d'autres termes, il faut que $\forall i=1$ à n $| (BIM \cap \mathfrak{E}_{C_i}) \cap \mathfrak{E}_{min} | \geq | BIM \cap \mathfrak{E}_{C_i} | - k_i'$. Cette condition peut être aussi exprimée par

$$\forall i=1 \text{ à } n \; | (BIM \cap BSm \cap \mathfrak{E}_{C_i}) \cap \mathfrak{E}_{min} | \geq | BIM \cap \mathfrak{E}_{C_i} | - k_i'$$

exemple

La condition impose de trouver $\mathfrak{E}_{min}$ vérifiant

$| 134 \cap 45 \cap \mathfrak{E}_{min} | \geq | 134 \cap 45 | - 0$ et $| 134 \cap 35 \cap \mathfrak{E}_{min} | \geq | 134 \cap 35 | - 1$, ou encore $| 4 \cap \mathfrak{E}_{min} | \geq 1$ et $| 3 \cap \mathfrak{E}_{min} | \geq 0$,

EO devient {14, 34, 134, 145, 345, 1345}. Voyons pourquoi $\mathfrak{E}_{min} = 13$ ne convient pas. Le cube partiel obtenu est [13, 12346]. L'ensemble d'indices hérités est égal à 246. Ceci impose les indices d'intersection $246 \cap 45 = 4$ et

$246 \cap 35 = \emptyset$ pour K_1 et K_2 respectivement. On note que l'intersection partielle obtenue avec K_1 est de dimension 1 et donc trop grande.

3.2.4.6. Condition nécessaire sur l'existence d'un nombre suffisant d'indices pour compléter individuellement chaque intersection

Si la condition précédente est vérifiée, alors les intersections partielles sont plus petites que celles voulues. Considérons un cube K_i; l'ensemble des indices composant son intersection partielle est $(BIM \cap \mathfrak{E}_{C_i})/\mathfrak{E}_{min}$. Il faut donc la compléter cette intersection avec $k_i' - | (BIM \cap \mathfrak{E}_{C_i})/\mathfrak{E}_{min} |$ indices. Ces indices doivent être pris dans $\mathfrak{E}_{C_i}$ (car ils forment l'intersection), sont différents de ceux de $BIM \cap \mathfrak{E}_{C_i}$, et n'appartiennent pas à $\mathfrak{E}_{min}$ car ces indices d'intersection appartiennent forcément à $\mathfrak{E}_C$. L'ensemble d'indices disponibles est donc égal à $(\mathfrak{E}_{C_i}/BIM)/\mathfrak{E}_{min}$. Il faut donc qu'il reste suffisamment d'indices pour compléter chaque intersection, d'où $k_i' - | (BIM \cap \mathfrak{E}_{C_i})/\mathfrak{E}_{min} | \leq | (\mathfrak{E}_{C_i}/BIM)/\mathfrak{E}_{min} |$, ou encore $k_i' \leq | (BIM \cap \mathfrak{E}_{C_i})/\mathfrak{E}_{min} | + | (\mathfrak{E}_{C_i}/BIM)/\mathfrak{E}_{min} |$. Or $BIM \cap \mathfrak{E}_{C_i}$ et $BIM/\mathfrak{E}_{C_i}$ sont deux ensembles disjoints, ce qui implique $(BIM \cap \mathfrak{E}_{C_i})/\mathfrak{E}_{min}$ et $(\mathfrak{E}_{C_i}/BIM)/\mathfrak{E}_{min}$ sont encore disjoints, ce qui entraîne que $| (BIM \cap \mathfrak{E}_{C_i})/\mathfrak{E}_{min} | + | (\mathfrak{E}_{C_i}/BIM)/\mathfrak{E}_{min} | = | \mathfrak{E}_{C_i}/\mathfrak{E}_{min} |$. La condition nécessaire devient donc $k_i' \leq | \mathfrak{E}_{C_i}/\mathfrak{E}_{min} |$, ou encore $| \mathfrak{E}_{C_i} \cap \mathfrak{E}_{min} | \leq | \mathfrak{E}_{C_i} | - k_i'$. Par définition, on a aussi $| \mathfrak{E}_{C_i} | = k_i$.

Finalement, pour tous les cubes, en utilisant le fait que $\mathfrak{E}_{min} \subseteq BSm$, on obtient la condition nécessaire suivante:

$$\forall i=1 \text{ à } n \; | (BSm \cap \mathfrak{E}_{C_i}) \cap \mathfrak{E}_{min} | \leq k_i - k_i'.$$

exemple

La condition impose de trouver $\mathfrak{E}_{min}$ vérifiant

$| 1345 \cap 45 \cap \mathfrak{E}_{min} | \leq 2 - 0$ et $| 1345 \cap 35 \cap \mathfrak{E}_{min} | \leq 2 - 1$, ou encore $| 45 \cap \mathfrak{E}_{min} | \leq 2$ et $| 35 \cap \mathfrak{E}_{min} | \leq 1$.

EO devient {14, 34, 134, 145}. Voyons pourquoi $\mathfrak{E}_{min} = 345$ ne convient pas. Le cube partiel obtenu est [345, 123456]. L'ensemble d'indices hérités est égal à 126. Ceci impose les indices d'intersection $126 \cap 45 = \emptyset$ et $126 \cap 35 = \emptyset$ pour K_1 et K_2 respectivement. Il faut fournir un indice pour l'intersection avec K_2 et celui-ci devrait être pris dans $\mathfrak{E}_{C2} = 35$. Or ces deux indices ont déjà été utilisés dans $\mathfrak{E}_{min}$.

3.2.4.7. Condition nécessaire sur la place restante pour compléter individuellement chaque intersection

Dans la section précédente, on a cherché s'il existait suffisamment d'indices pour compléter les intersections. Néanmoins on n'a pas vérifié s'il existait suffisamment de place dans le k-cube recherché pour rajouter les indices nécessaires pour compléter ces intersections. Une fois l'héritage $\mathfrak{E}_H = BIM/\mathfrak{E}_{min}$ obtenu, la place restante dans le k-cube pour d'autres indices est égale à
$k - |\mathfrak{E}_H|$. L'intersection imposée pour chaque cube K_i étant égale à $\mathfrak{E}_H \cap \mathfrak{E}_{C_i}$, la place restante doit pouvoir contenir les $k_i' - |\mathfrak{E}_H \cap \mathfrak{E}_{C_i}|$ indices à fournir pour compléter l'intersection.La place restante $k - |\mathfrak{E}_H|$ doit donc être inférieure ou égale à $k_i' - |\mathfrak{E}_H \cap \mathfrak{E}_{C_i}|$, ce qui peut aussi s'écrire $k - k_i' \leq |\mathfrak{E}_H| - |\mathfrak{E}_H \cap \mathfrak{E}_{C_i}|$. Comme $\mathfrak{E}_H \cap \mathfrak{E}_{C_i} \subseteq \mathfrak{E}_H$, on a $|\mathfrak{E}_H| - |\mathfrak{E}_H \cap \mathfrak{E}_{C_i}| = |\mathfrak{E}_H/\mathfrak{E}_{C_i}|$ et ainsi $|\mathfrak{E}_H/\mathfrak{E}_{C_i}| \leq k - k_i'$. Remplaçons $\mathfrak{E}_H$ par $BIM/\mathfrak{E}_{min}$, on obtient $|(BIM/\mathfrak{E}_{C_i})/\mathfrak{E}_{min}| \leq k - k_i'$, ce qui amène à la condition nécessaire suivante

$$\forall i=1 \text{ à } n \; |((BSm \cap BIM)/\mathfrak{E}_{C_i}) \cap \mathfrak{E}_{min}| \geq |(BIM/\mathfrak{E}_{C_i})| - k + k_i'$$

exemple

La condition impose de trouver $\mathfrak{E}_{min}$ vérifiant
$|(134/45) \cap \mathfrak{E}_{min}| \geq |12346/45| - 3 + 0$
et $|(134/35) \cap \mathfrak{E}_{min}| \geq |12346/35| - 3 + 1$, ou encore
$|13 \cap \mathfrak{E}_{min}| \geq 1$ et $|14 \cap \mathfrak{E}_{min}| \geq 2$.

EO devient {14, 134, 145}. Voyons pourquoi $\mathfrak{E}_{min} = 34$ ne convient pas. Le cube partiel obtenu est [34, 12346]. L'ensemble d'indices hérités est égal à

126. Ceci impose les indices d'intersection $126 \cap 45 = \emptyset$ et $126 \cap 35 = \emptyset$ pour K_1 et K_2 respectivement. Il faut fournir un indice pour l'intersection avec K_2 mais il ne reste plus de place pour introduire cet indice dans $\mathfrak{E}_{max}$.

3.2.4.8. Condition nécessaire sur la possibilité de compléter le cube une fois chaque intersection construite individuellement

Considérons le cube recherché et faisons le bilan de sa construction partielle effectuée jusqu'à maintenant. $\mathfrak{E}_{min} \subseteq$ BSm a été choisi comme origine; cela a imposé un héritage $\mathfrak{E}_H$ = BIM/$\mathfrak{E}_{min}$ pour la partie caractéristique $\mathfrak{E}_C$, dû aux origines des autres cubes. Pour chacun des cubes K_i, cet héritage a imposé une intersection partielle $\mathfrak{E}_H \cap \mathfrak{E}_{C_i}$. Celle-ci a été complétée par des indices de $\mathfrak{E}_{C_i}$, autres que ceux de BIM ou de $\mathfrak{E}_{min}$, pour obtenir une intersection à k_i' indices. Si le k-cube n'est pas constitué, c'est-à-dire si l'ensemble $\mathfrak{E}_C$ construit jusqu'à présent ne comporte pas encore k indices, alors il faudra compléter le cube partiel par un nombre nécessaire d'indices.

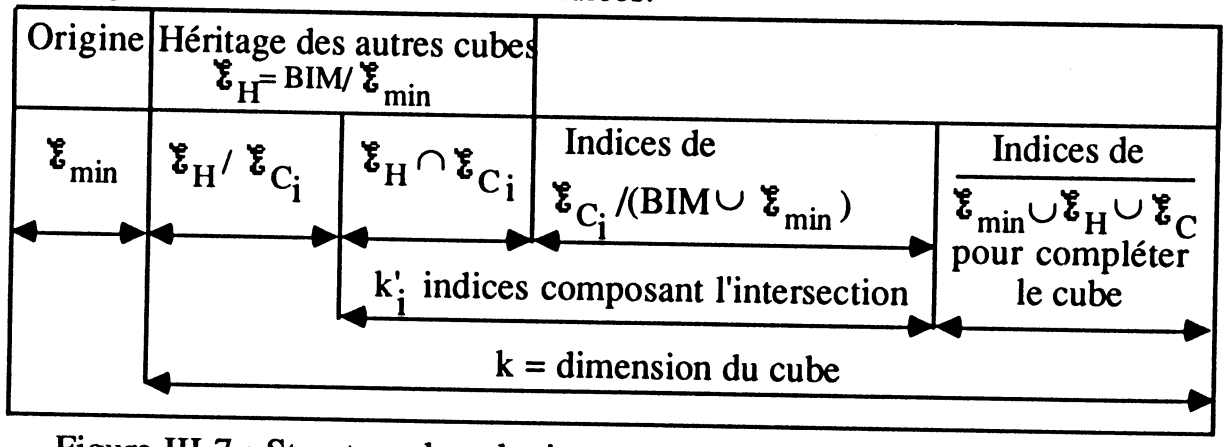

Figure III-7 : Structure du cube intersectant au vu de chaque intersection

On voit d'après le schéma précédent que le sous-ensemble de $\mathfrak{E}_{max}$ obtenu contient $\mathfrak{E}_{min}$, $\mathfrak{E}_H$ et $k_i' - | \mathfrak{E}_H \cap \mathfrak{E}_{C_i} |$ indices de $(\mathfrak{E}_{C_i}/\text{BIM})/\mathfrak{E}_{min}$ utiles à compléter l'intersection avec le cube K_i. $\mathfrak{E}_H$ peut aussi s'écrire comme union disjointe de $\mathfrak{E}_H \cap \mathfrak{E}_{C_i}$ et $\mathfrak{E}_H/\mathfrak{E}_{C_i}$; $\mathfrak{E}_H/\mathfrak{E}_{C_i}$ est l'ensemble d'indices ne prenant pas part à l'intersection. Le nombre d'indices qui composent partiellement $\mathfrak{E}_C$ est donc égal à $| \mathfrak{E}_H/\mathfrak{E}_{C_i} | + k_i'$. Le nombre d'indices à fournir pour compléter le cube au vu du cube K_i est ainsi égal à $k - | \mathfrak{E}_H/\mathfrak{E}_{C_i} | - k_i'$.

Pour compléter le cube, les indices nécessaires ne peuvent être pris ni dans $\mathfrak{E}_{min}$ ni dans $\mathfrak{E}_H/\mathfrak{E}_{C_i}$. Ils ne peuvent pas non plus être pris parmi les indices de $\mathfrak{E}_{C_i}$ ne composant pas l'intersection, car l'intersection a été construite avec la dimension voulue. $\overline{\mathfrak{E}_{min} \cup \mathfrak{E}_H/\mathfrak{E}_{C_i} \cup \mathfrak{E}_{C_i}}$ est donc l'ensemble d'indices disponibles. Le nombre d'indices disponibles est égal à $| \overline{\mathfrak{E}_{min} \cup \mathfrak{E}_H/\mathfrak{E}_{C_i} \cup \mathfrak{E}_{C_i}} |$, c'est-à-dire $N - | \mathfrak{E}_{min} \cup \mathfrak{E}_H/\mathfrak{E}_{C_i} \cup \mathfrak{E}_{C_i} |$. Or $\mathfrak{E}_H/\mathfrak{E}_{C_i}$ est disjoint de $\mathfrak{E}_{min}$ et de $\mathfrak{E}_{C_i}$, ce qui donne $| \mathfrak{E}_{min} \cup \mathfrak{E}_H/\mathfrak{E}_{C_i} \cup \mathfrak{E}_{C_i} | = | \mathfrak{E}_{min} \cup \mathfrak{E}_{C_i} | + | \mathfrak{E}_H/\mathfrak{E}_{C_i} |$. En réécrivant $\mathfrak{E}_{C_i} \cup \mathfrak{E}_{min}$ par $\mathfrak{E}_{C_i} \cup \mathfrak{E}_{min}/\mathfrak{E}_{C_i}$, le nombre d'indices disponibles est alors égal à $N - | \mathfrak{E}_H/\mathfrak{E}_{C_i} | - | \mathfrak{E}_{C_i} | - | \mathfrak{E}_{min}/\mathfrak{E}_{C_i} |$. La condition pour pouvoir compléter le cube au vu du cube K_i est donc la suivante:

$$k - | \mathfrak{E}_H/\mathfrak{E}_{C_i} | - k_i' \leq N - | \mathfrak{E}_H/\mathfrak{E}_{C_i} | - | \mathfrak{E}_{C_i} | - | \mathfrak{E}_{min}/\mathfrak{E}_{C_i} |$$

ce qui, en simplifiant, se réécrit pour tous les cubes:

$$\forall i=1 \text{ à } n \; | (BSm/\mathfrak{E}_{C_i}) \cap \mathfrak{E}_{min}) | \leq N - k_i - k + k_i'$$

exemple

La condition impose de trouver $\mathfrak{E}_{min}$ vérifiant

$| (1345/45) \cap \mathfrak{E}_{min} | \leq 7 - 2 - 3 + 0$ et $| (1345/35) \cap \mathfrak{E}_{min} | \leq 7 - 2 - 3 + 1$, ou encore $| 13 \cap \mathfrak{E}_{min} | \leq 2$ et $| 14 \cap \mathfrak{E}_{min} | \leq 3$.

EO reste égal à {14, 134, 145}.

3.2.4.9. Méthode de recherche d'une origine valide

Dans l'exemple que l'on a développé pour illustrer les conditions nécessaires, on note que l'ensemble EO de tous les minorants de BSm a été construit, puis réduit à l'application de chacune des conditions nécessaires. Dans la réalité, une méthode visant à accélérer la recherche d'un ensemble d'origines potentielles est utilisée. Pour cela, on génère un ensemble d'inéquations à partir des conditions nécessaires, auquel on applique certaines règles de simplification.

Simplification d'un ensemble d'inéquations

On a vu dans ce qui précède que chaque condition nécessaire impose une restriction sur la valeur que peut prendre $\mathfrak{E}_{min}$. Cette restriction s'exprime sous la forme d'une inéquation de la forme $| \mathfrak{E}_{min} \cap \mathfrak{E} | \lesseqgtr c$, où $\lesseqgtr$ représente une des inégalités $\leq$ ou $\geq$, et $\mathfrak{E}$ représente un ensemble connu et c un nombre connu d'indices. Après application des différentes conditions nécessaires, un ensemble d'inéquations de la forme $\{ | \mathfrak{E}_{min} \cap \mathfrak{E}_k | \lesseqgtr c_k \}$ est obtenu.

On donne par la suite un ensemble de règles permettant de simplifier ces inéquations.

Inéquations inutiles

Deux cas d'inéquations inutiles se présentent.

Le premier cas concerne une inéquation de la forme $| \mathfrak{E}_{min} \cap \mathfrak{E} | \geq 0$. Cette inéquation est inutile car quelle que soit la valeur de $\mathfrak{E}_{min}$, cette inégalité est vérifiée.

Le deuxième cas concerne une inéquation de la forme $| \mathfrak{E}_{min} \cap \mathfrak{E} | \leq c$, avec la condition $| \mathfrak{E} | \leq c$. Cette inéquation est alors inutile car quelle que soit la valeur de $\mathfrak{E}_{min}$, on a $| \mathfrak{E}_{min} \cap \mathfrak{E} | \leq | \mathfrak{E} | \leq c$.

Inéquations sans solution

On énonce deux cas d'inéquations sans solution.

Le premier cas concerne une inéquation de la forme $| \mathfrak{E}_{min} \cap \mathfrak{E} | \geq c$, avec la condition $| \mathfrak{E} | < c$. Il n'existe pas de solution car, sous la condition précédente, l'inéquation $| \mathfrak{E}_{min} \cap \mathfrak{E} | \geq c$ impliquerait $| \mathfrak{E}_{min} \cap \mathfrak{E} | > | \mathfrak{E} |$. Ceci est impossible car quelle que soit la valeur de $\mathfrak{E}_{min}$, $| \mathfrak{E}_{min} \cap \mathfrak{E} | \leq | \mathfrak{E} |$.

Le deuxième cas concerne une inéquation de la forme $| \mathfrak{E}_{min} \cap \mathfrak{E} | \leq c$, avec la condition $c < 0$. Il n'existe pas de solution car, sous cette condition, on aurait $| \mathfrak{E}_{min} \cap \mathfrak{E} | < 0$, ce qui est impossible.

Réduction du problème

Deux cas intéressants peuvent permettre de réduire le problème de la recherche de $\mathfrak{E}_{min}$. Le premier cas permet de trouver des indices dont on est sûr qu'ils appartiennent à $\mathfrak{E}_{min}$. Par contre, le deuxième cas permet de trouver ceux qui n'appartiennent pas à $\mathfrak{E}_{min}$.

Le premier cas concerne une inéquation de la forme $| \mathfrak{E}_{min} \cap \mathfrak{E}_m | \geqslant | \mathfrak{E}_m |$ pour un m donné. On montre dans ce cas que $\mathfrak{E}_{min} \supseteq \mathfrak{E}_m$ et $\mathfrak{E}_{min}$ est solution de l'ensemble d'inéquations réduit $\{ | \mathfrak{E}_{min} \cap (\mathfrak{E}_k / \mathfrak{E}_m) | \leqslant c_k - | \mathfrak{E}_k \cap \mathfrak{E}_m |; k \neq m \}$.

Preuve

Notons d'abord que

$$| \mathfrak{E}_{min} \cap \mathfrak{E}_k | \leqslant c_k \Leftrightarrow | \mathfrak{E}_{min} \cap (\mathfrak{E}_k / \mathfrak{E}_m) | \leqslant c_k - | \mathfrak{E}_{min} \cap \mathfrak{E}_k \cap \mathfrak{E}_m |$$

On a aussi $| \mathfrak{E}_{min} \cap \mathfrak{E}_m | \geq | \mathfrak{E}_m |$

$$\Leftrightarrow | \mathfrak{E}_{min} \cap \mathfrak{E}_m | = | \mathfrak{E}_m |$$
$$\Leftrightarrow \mathfrak{E}_{min} \supseteq \mathfrak{E}_m$$
$$\Leftrightarrow \mathfrak{E}_{min} \cap \mathfrak{E}_m = \mathfrak{E}_m$$

Sous la condition $\mathfrak{E}_{min} \cap \mathfrak{E}_m = \mathfrak{E}_m$,

$$| \mathfrak{E}_{min} \cap (\mathfrak{E}_k / \mathfrak{E}_m) | \leqslant c_k - | \mathfrak{E}_{min} \cap \mathfrak{E}_k \cap \mathfrak{E}_m |$$
$$\Leftrightarrow$$
$$| \mathfrak{E}_{min} \cap (\mathfrak{E}_k / \mathfrak{E}_m) | \leqslant c_k - | \mathfrak{E}_k \cap \mathfrak{E}_m |$$

cqfd

Le deuxième cas concerne une inéquation de la forme $| \mathfrak{E}_{min} \cap \mathfrak{E}_m | \leqslant 0$ pour un m donné. On montre dans ce cas que $\mathfrak{E}_{min} \cap \mathfrak{E}_m = \emptyset$ et $\mathfrak{E}_{min}$ est solution de l'ensemble d'inéquations réduit $\{ | \mathfrak{E}_{min} \cap (\mathfrak{E}_k / \mathfrak{E}_m) | \leqslant c_k; k \neq m \}$.

Preuve

Notons d'abord les équivalences suivantes:

$$| \mathfrak{E}_{min} \cap \mathfrak{E}_m | \leq 0 \Leftrightarrow | \mathfrak{E}_{min} \cap \mathfrak{E}_m | = 0 \Leftrightarrow \mathfrak{E}_{min} \cap \mathfrak{E}_m = \emptyset$$

Sous la condition $\mathfrak{E}_{min} \cap \mathfrak{E}_m = \emptyset$,

$$| \mathfrak{E}_{min} \cap (\mathfrak{E}_k / \mathfrak{E}_m) | \leqslant c_k - | \mathfrak{E}_{min} \cap \mathfrak{E}_k \cap \mathfrak{E}_m | \Leftrightarrow | \mathfrak{E}_{min} \cap (\mathfrak{E}_k / \mathfrak{E}_m) | \leqslant c_k$$

cqfd

exemple

Soit le système d'inéquations suivant obtenu en appliquant les conditions nécessaires à l'exemple précédent:

$| \mathfrak{E}_{min} \cap 134 | \geq 2$

$| \mathfrak{E}_{min} \cap 4 | \geq 1$

$| \mathfrak{E}_{min} \cap 3 | \geq 0$

$| \mathfrak{E}_{min} \cap 13 | \geq 1$

$| \mathfrak{E}_{min} \cap 14 | \geq 2$

$| \mathfrak{E}_{min} \cap 45 | \leq 2$

$| \mathfrak{E}_{min} \cap 35 | \leq 1$

$| \mathfrak{E}_{min} \cap 13 | \leq 2$

$| \mathfrak{E}_{min} \cap 14 | \leq 3$

On note que les inéquations 3, 6, 8 et 9 sont inutiles. Des inéquations restantes, on note que les inéquations 2 et 5 entraînent que $\mathfrak{E}_{min} \supseteq 14$. En appliquant les règles de réduction, on aboutit à:

$| \mathfrak{E}_{min} \cap 3 | \geq 0$

$| \mathfrak{E}_{min} \cap 3 | \geq 0$

$| \mathfrak{E}_{min} \cap 35 | \leq 1$

Les inéquations 1 et 4 deviennent inutiles. Il ne subsiste que l'inéquation 7. A partir de là, il suffit de rechercher les minorants de 35 vérifiant cette inégalité à savoir Ø, 3 et 5. Les solutions pour $\mathfrak{E}_{min}$ sont donc 14, 134 et 145. On retrouve bien l'ensemble EO des origines retenues.

3.2.4.10. Contre-exemple montrant que les conditions nécessaires précédentes ne sont pas suffisantes

Après application de toutes les conditions nécessaires précédentes on aboutit à un ensemble d'origines possibles pour le cube recherché. On veut montrer ici que tous les éléments de cet ensemble ne permettent pas de construire le cube recherché. Considérons pour cela la solution $\mathfrak{E}_{min} = 134$; alors on ne peut pas trouver de cube respectant à la fois les dimensions voulues pour toutes les intersections. Construisons le cube; le 2-cube partiel obtenu est égal à [134, 12346]. L'ensemble d'indices hérités est égal à 26. Ceci impose les indices d'intersection $26 \cap 45 = Ø$ et $26 \cap 35 = Ø$ pour K_1 et K_2 respectivement. Il faut

alors fournir un indice pour l'intersection avec K_2, et celui-ci ne peut qu'être égal à 5 ($\mathfrak{E}_{C2}/\mathfrak{E}_{min}$ = 35/134). Ce qui donnerait le 3-cube [134, 123456]. Or ce cube admet une intersection de dimension 1 avec K_1, ce qui est trop grand. Donc avec $\mathfrak{E}_{min}$ = 134, on ne peut pas construire le cube recherché.

Par contre pour $\mathfrak{E}_{min}$ = 14 et $\mathfrak{E}_{min}$ = 145, les cubes obtenus sont respectivement [14, 12346] et [145, 123456]; ce sont deux 3-cubes qui respectent les dimensions des intersections.

3.2.4.11. Construction du cube recherché au vu de l'ensemble des origines possibles

Problème général

Soit EO l'ensemble des origines vérifiant les conditions nécessaires; considérons une origine $\mathfrak{E}_{min} \in$ EO et résumons la construction du cube recherché à partir de l'origine choisie.

L'héritage imposé pour $\mathfrak{E}_C$ par le choix de $\mathfrak{E}_{min}$ est BIM/$\mathfrak{E}_{min}$. La dimension du cube partiel est donc | BIM/$\mathfrak{E}_{min}$ |.

2) Il faut donc compléter le cube partiel obtenu par un ensemble E à k - | BIM/$\mathfrak{E}_{min}$ | indices pris dans BIM $\cup$ $\mathfrak{E}_{min}$.

L'intersection imposée par l'héritage pour le cube K_i est égale à BIM$\cap\mathfrak{E}_{C_i}/\mathfrak{E}_{min}$, de dimension | BIM$\cap\mathfrak{E}_{C_i}/\mathfrak{E}_{min}$ |.

Il faut fournir k_i' - | BIM$\cap\mathfrak{E}_{C_i}/\mathfrak{E}_{min}$ | indices pris dans $\mathfrak{E}_{C_i}/(\mathfrak{E}_{min} \cup$ BIM) pour compléter chacune des intersections.

Au vu de toutes ces remarques, on doit chercher un ensemble E vérifiant

$$| E \cap \text{BIM} \cup \mathfrak{E}_{min} | = k - | \text{BIM}/\mathfrak{E}_{min} |$$

et

$$\forall i=1 \text{ à } n \; | E \cap (\mathfrak{E}_{C_i}/(\text{BIM} \cup \mathfrak{E}_{min})) | = k_i' - | (\text{BIM} \cap \mathfrak{E}_{C_i})/\mathfrak{E}_{min} |$$

exemple

Pour $\mathfrak{E}_{min}$ = 134 dans l'exemple précédent, on doit trouver E vérifiant | E$\cap$57 | = 1 et | E$\cap$5 | = 0 et | E$\cap$5 | =1

$| E \cap 5 | = 0$ et $| E \cap 5 | = 1$ étant contradictoires, il n'existe pas de solution.

Pour $\mathfrak{E}_{min} = 14$ dans l'exemple précédent, on doit trouver E vérifiant $| E \cap 57 | = 0$ et $| E \cap 5 | = 0$ et $| E \cap 5 | = 0$.

La solution est $E = \emptyset$, d'où le cube [14, 14 236].

Résolution d'un ensemble d'équations de la forme $| E_k \cap E | = e_k$

Comme annoncé dans le paragraphe précédent, la construction des cubes solutions du problème posé se fera donc par la recherche d'un ensemble E vérifiant les égalités précédentes dès qu'une des origines $\mathfrak{E}_{min}$ possibles est fixée. On se retrouve donc avec un ensemble d'équations de la forme $\{ | E_k \cap E | = e_k ; k = 1 \text{ à } P+1 \}$ où E_k est un ensemble d'indices, e_k est un nombre d'indices à fournir et P dénote le nombre de cubes du problème. On peut montrer que ce problème se ramène à la résolution d'un système linéaire à solutions dans $\{0, 1\}^N$.

Equivalence à un système linéaire d'équations

Définition: Vecteur indicateur

Soit E une partie de {1, ..., N}. On définit le vecteur indicateur VE de E par un vecteur de $\{0, 1\}^N$, vérifiant $VE_k = 1$ si et seulement si $k \in E$.

Vecteur indicateur d'une intersection

Soient X, A et B des parties de {1, ..., N}.
Alors $X = A \cap B \Leftrightarrow VX_k = VA_k . VB_k$.

Cardinal d'un ensemble

Le cardinal d'un ensemble E peut s'écrire $\sum_{i=1}^{N} VE_k$.

Ceci permet d'écrire $| E' \cap E | = \sum_{i=1}^{N} VE_k' . VE_k$. En terme de matrice on peut écrire $| E' \cap E | = VE' . {}^T VE$ où ${}^T VE$ est la matrice transposée de VE.

Construction de la matrice

Chaque équation de la forme $| E_k \cap E | = e_k$ peut donc s'écrire $VE_k . {}^T VE = e_k$. On construit ainsi une $(P+1) \times N$-matrice M dont la $k^{ième}$ ligne M_k

est égale à VE_k. On obtient donc l'équation $M.^TVE = {}^Te$, où e est le vecteur $(e_1, ..., e_p)$. Cette matrice n'est en général pas carrée, ce qui entraîne que son déterminant est nul et donc le système admet une solution non unique. De plus, ce problème s'apparente à un programme linéaire en nombres entiers car, pour ce système, on recherche des solutions qui appartiennent à l'espace B^N. La NP-complétude de ce problème et la complexité des algorithmes mis en oeuvre pour sa résolution font que l'on optera pour une méthode qui consiste à affecter les valeurs 0 et 1 à chaque variable jusqu'à obtention d'une solution.

Néanmoins, pour accélérer cette recherche, on utilisera quatre règles de simplification qui permettent de réduire le système.

Règles de simplification

Règle 1:

On est en présence d'une équation de la forme $| E_m \cap E | = | E_m |$. On montre dans ce cas que $E \supseteq E_m$ et que E est solution de l'ensemble réduit d'équations $\{ | (E_k/E_m) \cap E | = e_k - | E_k \cap E_m |$ pour $k \neq m\}$)

Preuve

Notons les équivalences suivantes:

$$(| E_m \cap E | = | E_m |) \Leftrightarrow (E_m \subseteq E) \Leftrightarrow (E_m \cap E = E_m).$$

De plus, pour $k \neq m$, en réécrivant $| E_k \cap E |$, on a

$$e_k = | E_k \cap E | = | (E_k \cap E_m) \cap E | + | (E_k/E_m) \cap E |.$$

Or $E_k \cap (E_m \cap E) = E_k \cap E_m$, d'où $e_k = | E_k \cap E_m | + | (E_k/E_m) \cap E |$

Ainsi $(| E_k \cap E | = e_k) \Leftrightarrow (| (E_k/E_m) \cap E | = e_k - | E_k \cap E_m |)$

cqfd

Règle 2:

Deux cas d'impossibilité se présentent ici.

Pour le premier cas, on est en présence d'une équation de la forme $| E_m \cap E | = e_m$ avec la condition $| E_m | < e_m$. On montre dans ce cas qu'il n'y

a pas de solution. En effet, chercher E vérifiant $| E_m \cap E | = e_m$ avec $e_m > | E_m |$ revient à chercher E vérifiant $| E_m \cap E | > | E_m |$, ce qui est impossible.

Pour le deuxième cas, on est en présence d'une équation de la forme $| E_m \cap E | = e_m$ avec la condition $e_m < 0$. Il n'y a pas de solution à cette équation car quelle que soit la valeur de E, $| E_m \cap E | > 0$.

Règle 3:

On est en présence d'une équation de la forme $| E_m \cap E | = 0$. On montre dans ce cas que $E \cap E_m = \emptyset$ et que E est solution de l'ensemble réduit d'équations $\{ | (E_k / E_m) \cap E | = e_k$ pour $k \neq m \}$)

Preuve

$(| E_m \cap E | = 0) \Leftrightarrow (E_m \cap E = \emptyset)$.

On a vu que $| E_k \cap E | = | (E_k \cap E_m) \cap E | + | (E_k / E_m) \cap E |$.

Or $E_k \cap (E_m \cap E) = \emptyset$, d'où $e_k = | E_k \cap E | = | (E_k / E_m) \cap E |$

Ainsi $(| E_k \cap E | = e_k) \Leftrightarrow (| (E_k / E_m) \cap E | = e_k)$

cqfd

Règle 4

On est en présence de deux équations de la forme $| E_m \cap E | = e_m$ et $| E_{m'} \cap E | = e_{m'}$ avec la condition $E_m \subseteq E_{m'}$. On montre alors que l'équation $| E_{m'} \cap E | = e_{m'}$ peut être remplacée par $| (E_{m'} / E_m) \cap E | = e_{m'} - e_m$.

Preuve

$(E_m \subseteq E_{m'}) \Leftrightarrow (E_{m'} = E_m \cup E_{m'} / E_m)$

Donc $(| E_{m'} \cap E | = e_{m'}) \Leftrightarrow (| E_m \cap E | + | (E_{m'} / E_m) \cap E) | = e_{m'})$

Or $| E_m \cap E | = e_m$.

Donc $(| E_{m'} \cap E | = e_{m'}) \Leftrightarrow (| (E_{m'} / E_m) \cap E) | = e_{m'} - e_m)$

cqfd

Recherche exhaustive de solution

On est dans le cas où aucune des quatre règles précédentes n'est applicable. Dans ce cas, on ne peut éviter la recherche exhaustive de solution. On montre alors qu'il suffit de considérer une équation quelconque $| E_m \cap E| = e_m$, de choisir un sous-ensemble E' de E_m à e_m indices et ensuite de résoudre l'ensemble réduit d'équations $\{ | (E_k/E_m) \cap E | = e_k - | E' \cap E_k |; k \neq m\}$. Cette opération est répétée avec d'autres sous-ensembles E' de E_m.

Le problème se ramène donc à démontrer l'équivalence suivante:

(E est solution de $\{ | E_k \cap E | = e_k \}$)

$\Leftrightarrow$

$\forall m \; \exists E'$ vérifiant ($| E' | = | E_m \cap E |$ et $E' \subseteq E$ et $E \subseteq E_m$)
tel que E est solution de $\{ | (E_k/E_m) \cap E | = e_k - | E_k \cap E' |; k \neq m\}$

Preuve

Notons tout d'abord l'égalité suivante:

$\forall E_k \; \forall E_m | (E_k/E_m) \cap E | = e_k - | E_k \cap E_m \cap E |$.

$\Rightarrow$:

Comme E est solution de $\{ | E_k \cap E | = e_k \}$, E est solution de $| E_m \cap E | = e_m$ pour tout m.

Pour tout m, si on pose $E' = E_m \cap E$, on obtient un ensemble E' vérifiant $| E' | = | (E_m \cap E |$ et $E' \subseteq E$ et $E \subseteq E_m$. De plus, pour tout $k \neq m$, on obtient $| (E_k/E_m) \cap E | = e_k - | E_k \cap E' |$.

$\Leftarrow$:

Fixons un m arbitraire

Soit E' vérifiant ($| E' | = | (E_m \cap E |$ et $E' \subseteq E$ et $E \subseteq E_m$)

($| E' | = | (E_m \cap E |$ et $E' \subseteq E$ et $E \subseteq E_m$)
$\Leftrightarrow$ ($| E' | = | (E_m \cap E |$ et $E' \subseteq E \cap E_m$)
$\Leftrightarrow$ ($E' = E_m \cap E$)

Donc $| (E_k/E_m) \cap E | = e_k - | E_k \cap E' |$

$\Leftrightarrow | (E_k/E_m) \cap E | = e_k - | E \cap E_k \cap E_m |$

cqfd

Cette dernière propriété nous permet en fait de choisir une équation et de chercher toutes les solutions pour cette équation. Pour chacune des solutions de cette équation, on pourra remettre à jour l'ensemble des équations restantes et ainsi, récursivement, réappliquer une des quatre règles de simplification, si cela est possible, sinon d'effectuer un nouveau choix selon cette dernière propriété.

exemple

Soit le système d'équations suivant:

$| E \cap 12346 | = 3$

(2) $| E \cap 245 | = 1$

(3) $| E \cap 1346 | = 2$

(4) $| E \cap 12357 | = 2$

Les équations 1 et 3 permettent d'appliquer la règle 4, à savoir remplacer l'équation 1 par $| E \cap 2 | = 1$.

Par cette transformation, l'équation 1 permet d'appliquer la règle 1: donc $E \supseteq 2$

Le système se réduit ainsi à

$| E \cap 45 | = 0$

(3) $| E \cap 1346 | = 2$

(4) $| E \cap 1357 | = 1$

L'équation 2 permet maintenant d'appliquer la règle 2: donc $E \cap 45 = \emptyset$

Le système se réduit donc à

$| E \cap 136 | = 2$

(4) $| E \cap 137 | = 1$

A ce point, aucune règle de simplification ne peut être appliquée. On effectue donc la recherche exhaustive.

Cherchons, par exemple, les solutions de l'équation 3, ce qui donne 13, 16 et 36.

Choisissons, par exemple, 13 comme solution; ceci nous amène, après remise à jour du système restant, à $| E \cap 7 | = -1$; donc il n'y a pas de solution (règle 3).

Par contre prenons 16 ou 36 comme solution; dans les deux cas, ceci amène à $| E \cap 7 | = 0$; donc $E \cap 7 = \emptyset$. Finalement, les solutions obtenues sont E = 126 et E = 236.

3.2.5. Impossibilité de généraliser les conditions nécessaires et suffisantes d'intersection entre deux cubes

D'autres conditions nécessaires basées sur les conditions nécessaires et suffisantes données pour l'intersection entre deux cubes auraient pu être utilisées. Celles-ci peuvent être énoncées de la manière suivante:

Le k-cube $K = [\mathfrak{E}_{min}, \mathfrak{E}_{max}]$ admet des intersections avec un ensemble de cubes
$\{K_1, ..., K_n\}$ de dimensions respectives $\{k_1, ..., k_n\}$ pour des intersections de dimensions respectives $\{k_1', ..., k_n'\}$

entraîne

$\mathfrak{E}_{min} \subseteq BSm$
et $\forall i \; (k_i' - k + | \mathfrak{E}_{min_i} | \leq | \mathfrak{E}_{min_i} \cap \mathfrak{E}_{min} | \leq N - k + k_i' - k_i)$
et $\forall i \; (| \mathfrak{E}_{C_i} \cap \mathfrak{E}_{min} | \leq k_i - k_i')$

Montrons que ces conditions ne sont pas suffisantes, et moins efficaces que les conditions nécessaires précédentes. Cherchons les origines $\mathfrak{E}_{min}$ vérifiant ces conditions nécessaires pour l'exemple précédent; on obtient alors le système d'inéquations suivant:

$$| 13 \cap \mathfrak{E}_{min} | \leq 2$$
$$| 13 \cap \mathfrak{E}_{min} | \geq 0$$
$$| 45 \cap \mathfrak{E}_{min} | \leq 2$$
$$| 14 \cap \mathfrak{E}_{min} | \leq 3$$
$$| 14 \cap \mathfrak{E}_{min} | \geq 1$$
$$| 35 \cap \mathfrak{E}_{min} | \leq 1$$

Les seules inéquations intéressantes sont (5) et (6); toutes les autres sont inutiles. Les origines vérifiant ces deux inégalités sont 1, 4, 14, 13, 34, 134, 15, 45 et 145. Au vu de ces solutions, deux remarques s'imposent. Premièrement, ces conditions nécessaires ne sont pas suffisantes, car l'ensemble d'origines trouvé est plus grand que l'ensemble des origines solutions. Deuxièmement, on note aisément que le "pouvoir filtrant" de ces conditions nécessaires est nettement plus faible que celles données précédemment. La raison en est que ces conditions nécessaires garantissent de pouvoir construire un cube ayant une intersection avec chaque cube K_i avec la dimension voulue, mais ne garantissent pas que le cube obtenu soit le même pour tous les cubes, ce qui est une condition plus forte.

A titre d'exemple, considérons l'origine $\mathfrak{E}_{min} = 45$. Au vu du cube $K_1 = [123, 12345]$, le seul 3-cube ayant une intersection de dimension 0 avec K_1 est [45, 12345]; au vu du cube $K_2 = [146, 13456]$, le cube solution est [45, 13456]. Les solutions retenues pour les 2 cubes sont différentes. Par contre, si $\mathfrak{E}_{min} = 14$, pour K_1, les cubes solutions sont [14, 12346] et [14, 12347] et pour K_2, les cubes solutions sont [14, 13467], [14, 14567], [14, 12346] et [14, 12456]. Le cube solution commun est donc [14, 12346].

3.2.6. Recherche d'un cube de dimension connue disjoint d'un cube donné

On propose ici de rechercher un k'-cube $K' = [\mathfrak{E}'_{min}, \mathfrak{E}'_{max}]$ disjoint d'un k-cube $K = [\mathfrak{E}_{min}, \mathfrak{E}_{max}]$. On rappelle que K et K' sont disjoints si et seulement si $\mathfrak{E}'_{min} \not\subseteq \mathfrak{E}_{max}$ ou $\mathfrak{E}_{min} \not\subseteq \mathfrak{E}'_{max}$. On effectuera la construction de K' en considérant les deux cas $\mathfrak{E}'_{min} \not\subseteq \mathfrak{E}_{max}$ et $\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max}$, et on donnera les conditions nécessaires et suffisantes de cette construction dans chacun des cas.

3.2.6.1. Choix de $\mathfrak{E}'_{min}$ tel que $\mathfrak{E}'_{min} \not\subseteq \mathfrak{E}_{max}$

Condition nécessaire et suffisante d'existence de $\mathfrak{E}'_{min}$

Dans ce cas-ci, le fait que $\mathfrak{E}'_{min} \not\subseteq \mathfrak{E}_{max}$ garantit la disjonction entre les deux cubes. On va montrer dans ce cas-là que la condition nécessaire et suffisante est la suivante: $(|\mathfrak{E}'_{min} \cap \mathfrak{E}_{max}| \geq 1)$ et $(|\mathfrak{E}'_{min}| \leq N - k')$.

Preuve

D'une part, $(\mathfrak{E}'_{min} \not\subseteq \mathfrak{E}_{max}) \Leftrightarrow (\mathfrak{E}'_{min} \cap \overline{\mathfrak{E}_{max}} \neq \emptyset)$

$\Leftrightarrow (| \mathfrak{E}'_{min} \cap \overline{\mathfrak{E}_{max}} | > 0) \Leftrightarrow (| \mathfrak{E}'_{min} \cap \overline{\mathfrak{E}_{max}} | \geq 1)$

D'autre part, pour construire le cube après avoir choisi $\mathfrak{E}'_{min}$ vérifiant la condition précédente, il faut et il suffit que le nombre d'indices disponibles pour constituer $\mathfrak{E}'_C$ soit supérieur à k', c'est-à-dire $N - | \mathfrak{E}'_{min} | \geq k'$, d'où la condition nécessaire: $| \mathfrak{E}'_{min} | \leq N - k'$.

La condition nécessaire et suffisante pour construire le cube recherché est donc

$(| \mathfrak{E}'_{min} \cap \overline{\mathfrak{E}_{max}} | \geq 1)$ et $(| \mathfrak{E}'_{min} | \leq N - k')$

cqfd

Construction de K'

Choisir $\mathfrak{E}'_{min} \not\subseteq \mathfrak{E}_{max}$ vérifiant les conditions nécessaires et suffisantes

Choisir k' indices de $\overline{\mathfrak{E}'_{min}}$ pour constituer $\mathfrak{E}'_C$.

3.2.6.2. Choix de $\mathfrak{E}'_{min}$ tel que $\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max}$

Condition nécessaire et suffisante d'existence de $\mathfrak{E}'_{min}$

Ici, il faudra à tout prix éviter de construire K' de telle sorte que $\mathfrak{E}'_{max} \supseteq \mathfrak{E}_{min}$. On va montrer que la condition nécessaire et suffisante ici est la suivante:

$$| \mathfrak{E}_{min} \cap \mathfrak{E}'_{min} | \leq | \mathfrak{E}_{min} | - 1 \text{ et } | \mathfrak{E}'_{min} | \leq N - k' - 1.$$

Preuve

Remarquons d'abord que $(\mathfrak{E}_{min} \subseteq \mathfrak{E}'_{min}) \Rightarrow (\mathfrak{E}_{min} \subseteq \mathfrak{E}'_{max})$, ce qu'on veut éviter.

Donc il faudra choisir $\mathfrak{E}'_{min}$ tel que $\mathfrak{E}'_{min} \not\supseteq \mathfrak{E}_{min}$

Or $(\mathfrak{E}'_{min} \not\supseteq \mathfrak{E}_{min}) \Leftrightarrow |\mathfrak{E}_{min} \cap \mathfrak{E}'_{min}| < |\mathfrak{E}_{min}|$

$\Leftrightarrow |\mathfrak{E}_{min} \cap \mathfrak{E}'_{min}| \leq |\mathfrak{E}_{min}| - 1$

Une fois cette condition garantie, pour que $\mathfrak{E}'_{max} \not\supseteq \mathfrak{E}_{min}$, il faut aussi s'assurer que $\mathfrak{E}'_C \not\supseteq \mathfrak{E}_{min}/\mathfrak{E}'_{min}$; cela provient de l'implication suivante:

$(\mathfrak{E}'_{max} \supseteq \mathfrak{E}_{min}) \Leftrightarrow (\mathfrak{E}'_C = \mathfrak{E}'_{max}/\mathfrak{E}'_{min} \supseteq \mathfrak{E}_{min}/\mathfrak{E}'_{min})$.

En dernier lieu, il faudra aussi s'assurer qu'il reste suffisamment d'indices pour pouvoir compléter $\mathfrak{E}'_C$.

Décomposons $\mathfrak{E}'_C$: $\mathfrak{E}'_C = \mathfrak{E}'_C \cap \mathfrak{E}_{min} \cup \mathfrak{E}'_C/\mathfrak{E}_{min}$.

Comme $\mathfrak{E}'_C \cap \mathfrak{E}'_{min} = \emptyset$, on a $\mathfrak{E}'_C/\mathfrak{E}'_{min} = \mathfrak{E}'_C$.

Ce qui nous permet de réécrire $\mathfrak{E}'_C$:

$$\mathfrak{E}'_C = \mathfrak{E}'_C \cap (\mathfrak{E}_{min}/\mathfrak{E}'_{min}) \cup \mathfrak{E}'_C/(\mathfrak{E}_{min} \cup \mathfrak{E}'_{min})$$

et donc

$$= |\mathfrak{E}'_C| = |\mathfrak{E}'_C \cap (\mathfrak{E}_{min}/\mathfrak{E}'_{min})| + |\mathfrak{E}'_C/(\mathfrak{E}_{min} \cup \mathfrak{E}'_{min})|$$

D'une part, on assure le fait que $(\mathfrak{E}'_C \not\supseteq \mathfrak{E}_{min}/\mathfrak{E}'_{min})$, par l'équivalence suivante:

$$(\mathfrak{E}'_C \not\supseteq \mathfrak{E}_{min}/\mathfrak{E}'_{min}) \Leftrightarrow |\mathfrak{E}'_C \cap (\mathfrak{E}_{min}/\mathfrak{E}'_{min})| \leq |(\mathfrak{E}_{min}/\mathfrak{E}'_{min})| - 1$$

et d'autre part, on garantit qu'il y ait suffisamment d'indices disponibles pour constituer $\mathfrak{E}'_C$ par le fait que:

$$|\mathfrak{E}'_C/(\mathfrak{E}_{min} \cup \mathfrak{E}'_{min})| \leq |\overline{\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}}|$$

Ces deux conditions entraînent donc:

$$k' \leq (|(\mathfrak{E}_{min}/\mathfrak{E}'_{min})| - 1) + |\overline{\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}}|.$$

Or ($\overline{\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}} = \overline{\mathfrak{E}_{min}} \cap \overline{\mathfrak{E}'_{min}}$ et $\mathfrak{E}_{min}/\mathfrak{E}'_{min} = \mathfrak{E}_{min} \cap \overline{\mathfrak{E}'_{min}}$)

$\Rightarrow k' \leq | \mathfrak{E}_{min} \cap \overline{\mathfrak{E}'_{min}} | - 1 + | \overline{\mathfrak{E}_{min}} \cap \overline{\mathfrak{E}'_{min}} |$ en remplaçant

Mais $| \mathfrak{E}_{min} \cap \overline{\mathfrak{E}'_{min}} | + | \overline{\mathfrak{E}_{min}} \cap \overline{\mathfrak{E}'_{min}} |$

$= | \mathfrak{E}_{min} \cap \overline{\mathfrak{E}'_{min}} \cup \overline{\mathfrak{E}_{min}} \cap \overline{\mathfrak{E}'_{min}} | = | \overline{\mathfrak{E}'_{min}} | = N - | \mathfrak{E}'_{min} |$

On en déduit $k' \leq N - | \mathfrak{E}'_{min} | - 1$ et ainsi $| \mathfrak{E}'_{min} | \leq N - k' - 1$.

Inversement, si cette condition est vérifiée, alors on pourra choisir des indices de $\mathfrak{E}_{min}/\mathfrak{E}'_{min}$ pour constituer $\mathfrak{E}'_C$ tout en respectant la condition $\mathfrak{E}'_C \not\supseteq \mathfrak{E}_{min}/\mathfrak{E}'_{min}$, et il restera suffisamment d'indices dans $\overline{\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}}$ pour compléter le cube.

cqfd

Construction du cube recherché

Choisir $\mathfrak{E}'_{min} \subseteq \mathfrak{E}_{max}$ vérifiant les conditions nécessaires et suffisantes $| \mathfrak{E}'_{min} | \leq N - k' - 1$ et $| \mathfrak{E}_{min} \cap \mathfrak{E}'_{min} | \leq | \mathfrak{E}_{min} | - 1$.

Choisir un nombre n d'indices de $| \mathfrak{E}_{min}/\mathfrak{E}'_{min} |$ vérifiant

$(k' - | \overline{\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}} | \leq n \leq | \mathfrak{E}_{min}/\mathfrak{E}'_{min} | - 1)$.

Choisir $(k' - n)$ indices de $| \overline{\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}} |$ pour compléter le cube.

Justifions l'encadrement de n. En construisant $\mathfrak{E}'_C$, on choisira certains indices dans $\mathfrak{E}_{min}/\mathfrak{E}'_{min}$ et les autres dans $\overline{\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}}$. On a vu que le nombre maximal d'indices qu'on peut prendre dans $\mathfrak{E}_{min}/\mathfrak{E}'_{min}$ est $| \mathfrak{E}_{min}/\mathfrak{E}'_{min} | - 1$, c'est-à-dire, $n \leq | \mathfrak{E}_{min}/\mathfrak{E}'_{min} | - 1$, d'où $k' - | \overline{\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}} | \leq n$.

Une fois, ces n indices de $\mathfrak{E}_{min}/\mathfrak{E}'_{min}$ choisis, il faut alors prendre $(k' - n)$ indices de $\overline{\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}}$. Cela n'est possible que si $k' - n \leq | \overline{\mathfrak{E}_{min} \cup \mathfrak{E}'_{min}} |$.

exemple

On cherche tous les 2-cubes disjoints de K = [1, 12] dans un hypercube de dimension 3.

Cherchons les solutions dans le cas 1; il faut trouver les ensembles $\mathfrak{E}'_{min}$ vérifiant

$$(|\ \mathfrak{E}'_{min} \cap 3\ | \geq 1) \text{ et } (|\ \mathfrak{E}'_{min}\ | \leq 3 - 2 = 1).$$

$\mathfrak{E}'_{min} = 3$ est la seule solution. Construisons le cube. On choisit 2 indices différents de 3, c'est-à-dire, 12. Le cube obtenu est donc [3, 123].

Cherchons les solutions dans le cas 2; il faut trouver les $\mathfrak{E}'_{min}$ vérifiant

$$|\ 1 \cap \mathfrak{E}'_{min}\ | \leq |\ 1\ | - 1 = 0 \text{ et } |\ \mathfrak{E}'_{min}\ | \leq 3 - 2 - 1 = 0$$

$\mathfrak{E}'_{min} = \emptyset$ est la seule solution. Construisons le cube. On choisit n indices de $\mathfrak{E}_{min}/\mathfrak{E}'_{min} = 1$ avec $(2 - |\ 23\ | = 0 \leq n \leq |\ 1\ | - 1 = 0)$. Puis, pour compléter le cube, on choisit 2 indices de 23. On obtient ainsi le cube [Ø, 23].

Ces deux solutions sont illustrées dans la figure suivante.

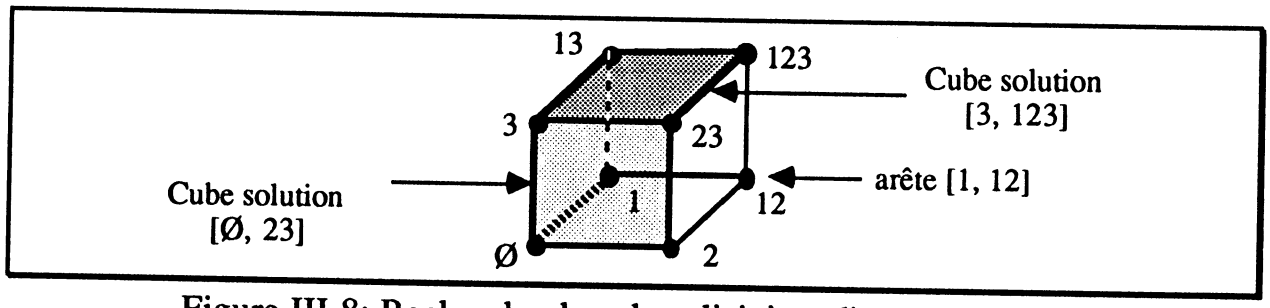

Figure III-8: Recherche de cubes disjoints d'un cube donné

CHAPITRE IV

ALGORITHME DE CODAGE

Ce chapitre est divisé en deux parties: la première traite de l'immersion cubique des groupes d'adjacence obtenus après la recherche de situations à fusion de monômes dans l'hypercube, et la deuxième montre comment les contraintes de codage induites par les situations à factorisation sont satisfaites.

4.1. IMMERSION DES GROUPES D'ADJACENCE DANS L'HYPERCUBE

A la fin de la recherche des situations à fusion potentielle de monômes, on obtient une liste de groupes d'adjacence pondérés à immerger dans l'hypercube; il faut alors affecter chaque groupe d'adjacence à un cube de l'hypercube.

Avant de décrire l'algorithme d'immersion, certaines notations seront introduites, et certaines propriétés sur les groupes d'adjacence et leur extension par ajout d'états fantômes seront énoncées.

4.1.1. Contraintes sur les états fantômes

Notations

On rappelle que l'ensemble de tous les états fantômes rajoutés à l'ensemble des états de l'automate considéré est noté F. Cet ensemble est non-vide si le nombre d'états de l'automate est strictement inférieur à 2^N, où N est le nombre de bits de codage choisi.

L'ensemble des états du $i^{ème}$ groupe d'adjacence sera noté L_i.

Comme tout groupe d'adjacence est affecté à un cube, le nombre d'états qu'il contient doit être égal à une puissance de 2. Néanmoins, pour certains groupes d'adjacence, cette condition n'est pas vérifiée. Pour remédier à cela, ils doivent être complétés avec des états fantômes. Ainsi, l'ensemble (éventuellement vide) des états fantômes rajoutés au $i^{ème}$ groupe d'adjacence sera noté F_i. Un groupe d'adjacence sera alors représenté par le couple $G_i = (L_i, F_i)$. Le cube auquel est affecté le groupe d'adjacence G_i est noté K_i.

Le groupe d'adjacence intersection de deux groupes d'adjacence G_i et G_j, noté G_{ij}, est égal à (L_{ij}, F_{ij}) où L_{ij} et F_{ij} représentent $L_i \cap L_j$ et $F_i \cap F_j$ respectivement.

Dans la suite, $\lceil X \rceil$ représentera le plus petit entier supérieur ou égal à X. De manière analogue, $\lfloor X \rfloor$ représentera le plus grand entier inférieur ou égal à X.

4.1.2. Contraintes sur l'ensemble des états fantômes d'un groupe d'adjacence

4.1.2.1. Extension d'un groupe d'adjacence

Soit G_i un groupe d'adjacence tel que $| L_i |$ soit différent d'une puissance de 2. Ce groupe d'adjacence doit alors être étendu par un ensemble d'états fantômes F_i pour pouvoir être affecté à un k_i-cube vérifiant $2^{k_i} = | L_i | + | F_i |$.

On en déduit que $k_i \geq \lceil \mathrm{Log}_2(| L_i |) \rceil$ et $| F_i | = 2^{k_i} - | L_i |$.

D'autre part, comme $| F_i | \leq | F |$, on a aussi $2^{k_i} \leq | L_i | + | F |$. Ceci entraîne que $k_i \leq \lceil \mathrm{Log}_2(| L_i | + | F |) \rceil$.

Les valeurs minimale et maximale de k_i, notées $(k_i)_{min}$ et $(k_i)_{max}$ respectivement, sont égales à $\lceil \mathrm{Log}_2(| L_i |) \rceil$ et $\lceil \mathrm{Log}_2(| L_i | + | F |) \rceil$.

Notons que dans certains cas, un groupe d'adjacence ne peut pas être étendu car il n'y a pas suffisamment d'états fantômes (ceci se traduit par $(k_i)_{min} > (k_i)_{max}$).

exemple

Considérons un automate pour lequel l'ensemble F des états fantômes disponibles admet 5 éléments, soit F = {f1, ..., f5} et soit $G_1=(L_1, F_1)$ le groupe d'adjacence pour lequel L_1 = {s1, s2, s3}.

Alors $(k_1)_{min} = 2$ et $(k_1)_{max} = 3$. Ainsi, on peut faire l'extension de G_1 de deux manières, soit avec $| F_1 | = 1$, ce qui entraîne que F_1 = {f1}, par exemple, soit avec $| F_1 | = 5$, ce qui entraîne que $F_1 = F$.

Considérons maintenant un autre groupe d'adjacence dont le nombre d'états est égal à 9. Pour pouvoir étendre le groupe d'adjacence, il faudrait que l'ensemble F des états fantômes comporte au moins 7 éléments, ce qui n'est pas le cas. Notons que $k_{min} = 4$ et $k_{max} = 3$.

4.1.2.2. Contraintes sur l'ensemble des états fantômes d'un groupe d'adjacence au vu des intersections avec d'autres groupes d'adjacence

Soit G_j un groupe d'adjacence déjà étendu et soit G_i un groupe d'adjacence dont l'extension doit être effectuée.

Deux cas peuvent se présenter, soit $| L_{ij} | \neq 0$, soit $| L_{ij} | = 0$, selon si l'intersection des ensembles d'états des deux groupes est vide ou non.

Cas $| L_{ij} | \neq 0$

Dans ce premier cas, les groupes d'adjacence sont affectés à deux cubes admettant une intersection. Soit k_{ij} la dimension du cube intersection; les $2^{k_{ij}}$ noeuds de ce cube contiennent alors les états en commun entre les deux groupes d'adjacence, ce qui impose $| L_{ij} | + | F_{ij} | = 2^{k_{ij}}$. La condition sur l'ensemble des états fantômes de G_i est alors $| F_j \cap F_i | = 2^{k_{ij}} - | L_{ij} |$, une fois k_{ij} déterminée.

Valeur minimale de k_{ij}

Au vu de l'équation précédente, on a $| L_{ij} | \leq 2^{k_{ij}}$, ce qui entraîne $k_{ij} \geq \lceil \mathrm{Log}_2(| L_{ij} |) \rceil$. D'autre part, ce cube est l'intersection des deux cubes K_i et K_j de dimensions respectives k_i et k_j, auxquels sont affectés les deux groupes d'adjacence G_i et G_j respectivement. Or une des conditions nécessaires d'existence d'une intersection entre les deux cubes $K_i = [\mathfrak{E}_{min_i}, \mathfrak{E}_{max_i}]$ et $K_j = [\mathfrak{E}_{min_j}, \mathfrak{E}_{max_j}]$ est la suivante: $| \mathfrak{E}_{min_i} \cap \mathfrak{E}_{min_j} | \leq N + k_{ij} - k_i - k_j$. On en déduit la condition nécessaire $k_{ij} \geq k_i + k_j - N$.

Ainsi la valeur minimale que peut prendre k_{ij}, notée $(k_{ij})_{min}$, est égale à

$$(k_{ij})_{min} = \max(\lceil \mathrm{Log}_2(| L_{ij} |) \rceil, k_i + k_j - N)$$

Valeur maximale de k_{ij}

$| F_i \cap F_j |$ vérifie $| F_i \cap F_j | \leq \min(| F_i | , | F_j |)$. Donc $2^{k_{ij}} = | L_{ij} | + | F_{ij} | \leq | L_{ij} | + \min(| F_i | , | Fj |)$. On en déduit que $k_{ij} \leq \lfloor \mathrm{Log}_2(| L_{ij} | + \min(| F_i | , | F_j |) \rfloor$.

Ainsi, la valeur maximale que peut prendre k_{ij}, notée $(k_{ij})_{max}$ est égale à

$$(k_{ij})_{max} = \lfloor Log_2(\mid L_{ij} \mid + \min(\mid F_i \mid, \mid F_j \mid)) \rfloor.$$

Cas $\mid L_{ij} \mid = 0$

Ici, il n'y a pas d'état commun entre les ensembles d'états des deux groupes d'adjacence. Il y a alors deux façons possibles d'effectuer l'extension du groupe d'adjacence G_i, selon si on choisit d'avoir $F_i \cap F_j = \emptyset$ ou $F_i \cap F_j \neq \emptyset$.

Dans le premier cas, on choisit d'affecter les deux groupes d'adjacence à deux cubes disjoints. Ceci impose donc à chercher F_i tel que $\mid F_{ij} \mid = 0$.

Dans le deuxième cas, on crée une intersection entre les deux groupes d'adjacence. Les valeurs minimale et maximale pour k_{ij} sont calculées de la même manière que dans le cas $\mid L_{ij} \mid \neq 0$.

Comme une intersection est recherchée, k_{ij} vérifie $k_{ij} \geq 0$. On obtient ainsi

$$(k_{ij})_{min} = \max(0, k_i + k_j - N).$$

et $$(k_{ij})_{max} = \lfloor Log_2(\min(\mid F_i \mid, \mid F_j \mid)) \rfloor.$$

Notons que, dans certains cas, il peut ne pas y avoir de solution à ce problème si $(k_{ij})_{min} > (k_{ij})_{max}$.

exemple

Plaçons-nous dans un hypercube de dimension 4 et considérons un automate à 12 états; donc l'ensemble F des états fantômes est égal à {f1, f2, f3, f4}. Supposons qu'un groupe d'adjacence G_1 soit égal à ({s1, s2, s3, s4, s5}, {f1, f2, f3}). Soit G_2 un groupe d'adjacence tel que L_2 = {s1, s2, s3, s6, s7, s8}. L'extension de G_2 se fait en utilisant un ensemble F_2 de 2 états fantômes. G_1 et G_2 peuvent alors être affectés à des 3-cubes. On est dans le cas d'une intersection non vide entre les deux groupes d'adjacence.

Les conditions sur le cube intersection imposent que $(k_{12})_{min} = 2$ et $(k_{12})_{max} = 2$. Comme $\mid L_{12} \mid = 3$, ceci entraîne que $\mid F_{12} \mid = 1$.

On recherche donc F_2 vérifiant $\mid F_2 \mid = 2$ et $\mid \{f1, f2, f3\} \cap F_2 \mid = 1$. Une solution est alors F_2 = {f1, f4}.

Ainsi G_{12} = ({s1, s2, s3, s6, s7, s8}, {f1, f4}) est une extension.

Considérons un deuxième exemple. Plaçons-nous dans un hypercube de dimension 3 et considérons un automate à 7 états. F est alors réduit à {f1}. Soit un groupe d'adjacence étendu G_1 = ({s1, s2, s3}, {f1}) et un groupe d'adjacence G_2 non encore étendu pour lequel L_2 = {s4, s5, s6}. Montrons que ces deux groupes ne peuvent avoir aucune intersection. En effet, pour une intersection, on aurait $(k_{12})_{min} = 1$ et $(k_{12})_{max} = 0$. Pour une disjonction, il faudrait trouver un état fantôme différent de f1 pour compléter G_2, ce qui est impossible.

4.1.3. Technique de retour-arrière et extension des groupes

Considérons l'ensemble des groupes d'adjacence suivants (on ne distinguera pas les états fantômes):

G_1 = {s1, s2, s3, s4}, G_2 = {s1, s2, s5, s6}, G_3 = {s1, s3}, G_4 = {s5, s7} et G_5 = {s3, s7}.

Sans entrer dans les détails, on donne les cubes affectés aux trois premiers groupes d'adjacence: K_1 = [Ø, 12], K_2 = [Ø, 13], K_3 = [Ø, 2].

Deux solutions possibles pour G_4 sont alors K_4 = [13, 123] et K_4' = [3, 23]. Les deux possibilités d'affectation des noeuds dans l'hypercube après immersion des quatre premiers groupes d'adjacence sont illustrées dans la figure suivante.

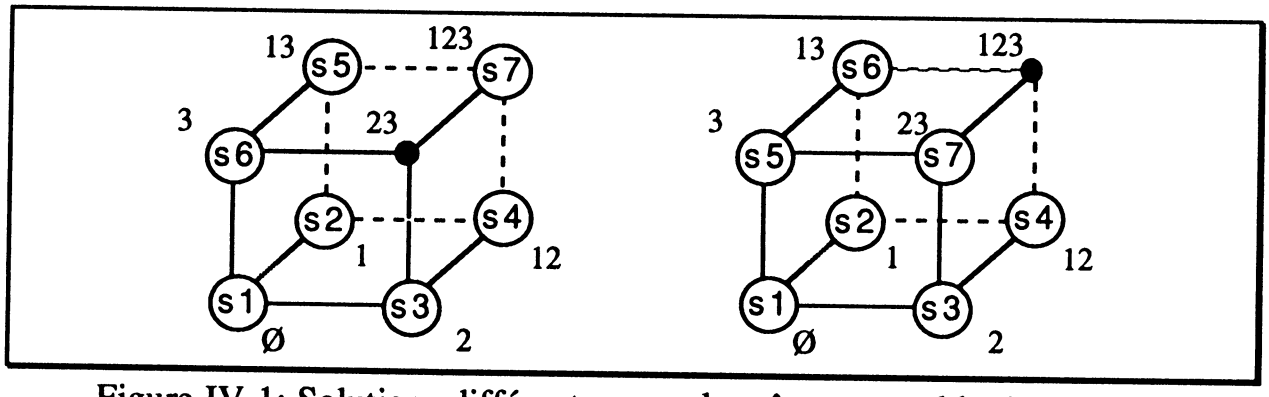

Figure IV-1: Solutions différentes pour le même ensemble de groupes d'adjacence

Si on choisit K_4 comme étant le cube auquel est affecté G_4, alors on ne trouvera pas de solution pour G_5. On sera obligé de remettre en cause la solution de G_4 et choisir K_4' comme alternative. Avec ce choix, l'immersion de G_5 devient possible et le cube associé K_5 est égal à [2, 23].

4.1.4. Nécessité de générer les intersections intermédiaires entre groupes d'adjacence

L'immersion des groupes d'adjacence requiert de les placer sur des cubes vérifiant des relations d'intersection et de disjonction induites par les parties communes entre ces groupes. Ainsi, si p groupes d'adjacence ont été affectés à p cubes, alors, pour le prochain groupe, on recherche les relations entre ce groupe et les p groupes précédents. On est alors tenté d'effectuer la recherche d'un cube vérifiant les relations d'intersection et de disjonction avec les p cubes. Or, on va montrer sur un exemple que cela n'est pas suffisant. En effet, lors de la recherche du cube pour le dernier groupe, on devra tenir compte de l'affectation des intersections intermédiaires entre groupes d'adjacence.

Considérons les trois groupes d'adjacence suivants qu'on voudrait immerger dans un 3-cube: G_1 = {s1, s2}, G_2 = {s1, s3} et G_3 = {s2, s3}.

G_1 est affecté au 1-cube K_1 = [Ø, 1]. Pour G_2, on note qu'il y a un état en commun avec G_1, ce qui impose de chercher un 1-cube ayant une intersection de dimension 0 avec K_1. Une solution pour K_2 est [Ø, 2]. On considère ensuite G_3 qui admet 1 sommet en commun avec G_1 et un sommet en commun avec G_2. Les états {s2, s3} devront donc être placés sur un 1-cube. La figure suivante illustre le placement des sommets s1, s2 et s3 après immersion des deux premiers groupes d'adjacence et montre qu'il n'est pas possible de placer les états {s2, s3} sur une arête, ou 1-cube.

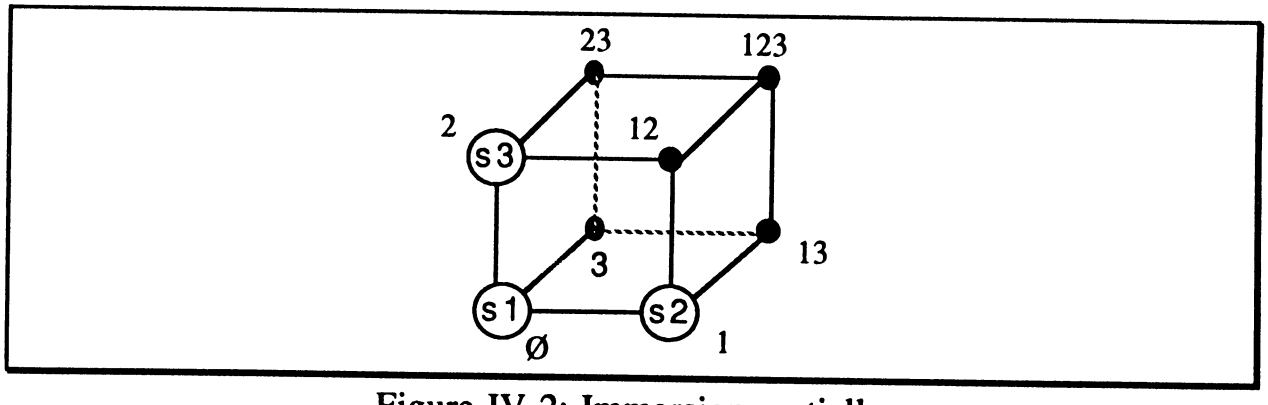

Figure IV-2: Immersion partielle

Pourtant, pour l'immersion du troisième groupe, on voudrait obtenir un 1-cube ayant pour intersections un 0-cube avec [Ø, 1] et un 0-cube avec [Ø, 2]. Un tel cube existe; c'est le cube [Ø, 3]. Ceci est en contradiction avec la conclusion précédente. Le problème provient du fait que l'on a spécifié les dimensions des cubes intersections au vu du nombre d'états en commun entre G_1

et G_3 d'une part, et entre G_2 et G_3 d'autre part, sans tenir compte des états qui étaient impliqués dans ces intersections.

En effet, on a imposé que les cubes intersections soient de dimension 0 car les intersections G_{13} et G_{23} comportent chacune un état. Par contre, on n'a pas spécifié que ces états étaient différents. Au vu de la solution [Ø, 3], on voit bien que c'est l'état s1 qui est considéré comme étant l'état en commun entre G_3 et G_1 d'une part, et entre G_3 et G_2 d'autre part, et non pas s2 et s3 respectivement comme c'est le cas.

Donc, pour éviter ce genre de problème, il s'avèrera nécessaire à chaque étape de calculer les intersections imposées par l'immersion avant de passer au traitement du prochain groupe d'adjacence. Ainsi, pour l'exemple utilisé on obtiendra

$$G_1 = \{ s1, s2\} \quad \mapsto \quad K_1 = [Ø, 1]$$
$$G_2 = \{ s1, s3\} \quad \mapsto \quad K_2 = [Ø, 2]$$
$$G_{12} = \{s1\} \quad \mapsto \quad K_{12} = K_1.K_2 = [Ø, Ø]$$

Au vu de cette solution, il faudra maintenant rechercher pour G_3 un 1-cube ayant pour intersections un 0-cube avec K_1 et un 0-cube avec K_2, et tel qu'il soit disjoint de K_{12}. Dans ce cas-là, on ne trouve pas de solution.

4.1.5. Extension des groupes d'adjacence aux cubes supérieurs

Dans l'exemple développé au-dessus, on note qu'il n'y a pas de solution au problème si les groupes d'adjacence sont affectés à des 1-cubes. Supposons que seuls ces trois états soient impliqués; on peut alors utiliser un ensemble de 5 états fantômes. Ceci entraîne alors que chaque groupe d'adjacence peut être affecté à un k-cube avec $1 \leq k \leq 2$. Dans le cas où $k = 2$, chaque groupe d'adjacence peut être étendu avec 2 états fantômes chacun. On montre dans ce qui suit la solution de l'immersion:

$$G_1 = (\{ s1, s2\}, \{f1, f2\}) \quad \mapsto \quad K_1 = [Ø, 12]$$
$$G_2 = (\{s1, s3\}, \{f1, f3\}) \quad \mapsto \quad K_2 = [Ø, 13]$$
$$G_{12} = (\{s1\}, \{f1\}) \quad \mapsto \quad K_{12} = [Ø, 1]$$
$$G_3 = (\{s2, s3\}, \{f1, f4\}) \quad \mapsto \quad K_3 = [Ø, 23]$$
$$G_{13} = (\{s2\}, \{f1\}) \quad \mapsto \quad K_{13} = [Ø, 2]$$
$$G_{23} = (\{s3\}, \{f3\}) \quad \mapsto \quad K_{23} = [Ø, 3]$$
$$G_{123} = (\{\}, \{f1\}) \quad \mapsto \quad K_{123} = [Ø, Ø]$$

Les codes de s1, s2 et s3 au vu de cette immersion sont donc [1], [2] et [3], ce qui correspond à un codage en 1 parmi N. En pratique, l'algorithme qui essaie d'immerger les groupes d'adjacence dans des cubes de dimension supérieure à la dimension minimale s'est avéré trop coûteux en temps CPU. C'est pourquoi on a choisi de se restreindre au cube de dimension minimale dans la réalisation de l'algorithme final.

4.1.6. Algorithme d'immersion

4.1.6.1. Présentation générale de l'algorithme

L'algorithme d'immersion des groupes d'adjacence est décrite sommairement dans la procédure suivante. Celle-ci reçoit en entrée l'ensemble trié des groupes d'adjacence G à immerger. En retour, elle construit G_IMM, l'ensemble des groupes d'adjacence pour lesquels l'immersion a réussi, et les intersections entre ces groupes d'adjacence. Elle construit de même l'ensemble des cubes solutions pour chacun des groupes d'adjacence de G_IMM.

```
Immersion (G, G_IMM, CUBES)
|  G_IMM ← {}
|  CUBES ← {}
|  pour chaque groupe d'adjacence Gi ε G faire
|  |  déterminer (ki)min et (ki)max en fonction de | Li | et de | F |
|  |  si (ki)min > (ki)max alors
|  |     TROUVE ← FAUX
|  |  sinon
|  |     ki = (ki)min
|  |  finsi
1 2
```

```
1 2
| | pour tout j tel que G_j ∈ G_IMM faire
| |   déterminer {L_ij = L_i∩L_j}
| |   déterminer {(k_ij)_min} en fonction de L_ij, k_i et k_j
| | finpour
| |
| | répéter
| |   pour tout j tel que G_j ∈ G_IMM
| |     déterminer {(k_ij)_max} en fonction de L_ij et F_j
| |   si ∃j tel que (k_ij)_min > (k_ij)_max alors
| |     TROUVE ← FAUX
| |   sinon
| |     répéter
| |       choisir (k_i1, ..., k_ip) t.q (k_ij)_min ≤ k_ij ≤ (k_ij)_max
| |       déterminer { | F_ij | } en fonction de k_i, k_j et k_ij,
| |       résoudre le système | F_i | = 2^ki - | L_i | et | F_ij | = 2^kij - | L_ij |
| |       si une solution pour F_i est trouvée alors
| |         trouver un cube K_i tel que dim(K_i) = k_i et dim(K_i.K_j) = k_ij
| |         si {K_i solutions} ≠ Ø alors
| |           stocker {K_i solutions} et choisir un cube K_i
| |           TROUVE ← VRAI
| |         finsi
| |       finsi
| |     jusqu'à (TROUVE
| |                 ou toute combinaison de (k_i1, ..., k_ip) a été utilisée)
| |   finsi
| |   si non TROUVE alors
| |         remettre en cause les solutions pour les G_j de G_IMM
| | jusqu'à TROUVE ou toutes les solutions ont été remises en cause
| |
| | si TROUVE alors
| |   G_IMM ← G_IMM ∪ {G_ij} ∪ {G_i}
| |   CUBES ← CUBES ∪ {K_ij} ∪ {K_i}
| | finsi
| finpour
fin Immersion
```

4.1.6.2. Explication détaillée de l'algorithme

L'algorithme précédent considère les groupes d'adjacence successivement par ordre de coût décroissant.

Un premier test vérifie qu'il y a suffisamment d'états fantômes pour l'extension d'un groupe d'adjacence incomplet. Par exemple, un groupe à 5 états ne pourrait pas être complété s'il y a seulement 2 états fantômes ($(k_i)_{min} = 3 > (k_i)_{max} = 2$). Si le test est correct, la dimension du cube est affectée à la dimension minimale.

Ensuite, les dimensions minimales et maximales des cubes associés aux intersections du groupe courant et des groupes déjà immergés sont calculées. Les dimensions minimales dépendent du nombre d'états en commun entre les groupes, et des dimensions des cubes associés aux groupes. Les dimensions maximales dépendent aussi du nombre d'états fantômes des groupes.

Un deuxième test vérifie que la construction des intersections est possible. Par exemple, un groupe d'adjacence {s1, s2, s3, s4} admet un seul état en commun avec un autre groupe {s1, s5, s6, s7}. L'intersection doit donc être affecté à un cube de dimension 0. Si on effectue l'immersion dans un hypercube de dimension 3, on ne trouvera pas de solution. La raison en est que, dans un 3-cube, toute intersection entre deux 2-cubes est au moins de dimension 1.

Une fois ces tests effectués, on choisit une combinaison valide de dimensions pour les intersections recherchées. Au vu de cette combinaison, un système d'équations est généré pour la recherche de l'ensemble des états fantômes du groupe considéré. Ce système est résolu. S'il n'y a pas de solution, une autre combinaison est essayée jusqu'à l'obtention d'une solution. On peut alors tenter une immersion dans l'hypercube. Si l'immersion réussit, l'ensemble des cubes solutions obtenus est stocké. Notons que cette immersion peut trouver par recherche-arrière, un autre ensemble de cubes pour l'ensemble des groupes d'adjacence déjà immergés. Si on ne trouve pas de cube, on essaie encore une autre combinaison jusqu'à ce que toutes les combinaisons aient été essayées.

Quand on sort de la boucle interne, si aucune solution n'a été trouvée, on remet en cause les solutions pour les groupes d'adjacence précédents. En effet, ces groupes ont été immergés pour un certain choix d'ensembles d'états fantômes. On recherche donc d'autres ensembles d'états fantômes pour lesquels une immersion est encore possible. Pour chacune des nouvelles solutions

obtenues, on peut chercher si l'immersion du groupe d'adjacence courant n'est toujours pas possible.

Si, malgré la remise en cause des solutions précédentes, aucune solution n'est obtenue pour le groupe considéré, alors celui-ci est "abandonné". Au cas contraire, les intersections de ce groupe avec les autres groupes sont calculées et stockées comme nouveaux groupes d'adjacence. De même, les cubes associés aux intersections sont calculés et rajoutés à la liste des cubes.

L'algorithme se termine quand il n'y a plus de groupe d'adjacence à considérer.

Notons que pour simplifier l'explication de l'algorithme, nous n'avons parlé que des intersections entre groupes d'adjacence et entre cubes, tout en sachant que pour certains groupes d'adjacence, une disjonction est possible. Dans ce cas, certaines parties de l'algorithme devront être adaptées. La détection d'une possibilité de disjonction se fera en vérifiant si $| L_i | = 0$. Dans le cas d'une disjonction entre les groupes G_i et G_j, il suffira d'imposer que $| F_{ij} | = 0$ et d'ignorer la valeur de k_{ij}. De même, on ne recherchera pas un cube K_j tel que $\dim(K_i.K_j) = k_{ij}$, mais tel qu'il soit disjoint de K_i.

4.1.6.3. Illustration de l'algorithme d'immersion

Considérons trois groupes d'adjacence constitués des ensembles d'états respectifs $L_1 = \{s1, s2, s3, s4, s5\}$, $L_2 = \{s1, s2, s6, s7, s8\}$ et $L_3 = \{s3, s6, s9\}$. On veut effectuer l'immersion de ces groupes d'adjacence dans un 5-cube. On supposera que l'ensemble des états fantômes est $F = \{f1, f2, f3, f4, f5\}$.

Considérons le premier groupe; on calcule k_1 qui vérifie $3 \leq k_1 \leq 3$. Donc, $k_1 = 3$, ce qui entraîne $| F_1 | = 3$. On prendra comme solution $F_1 = \{f1, f2, f3\}$. Il faut trouver un 3-cube pour ce groupe d'adjacence. Une solution pour K_1 est [Ø, 123].

On passe au deuxième groupe. La dimension k_2 du cube est égale à 3. Ceci entraîne $| F_2 | = 3$. On calcule $L_{12} = \{s1, s2\}$ de cardinal 2. On cherche la dimension de l'intersection: k_{12} vérifie $1 \leq k_{12} \leq 2$. Prenons $k_{12} = 1$, ce qui entraîne $| F_{12} | = 0$, c'est-à-dire $| F_2 \cap \{f1, f2, f3\} | = 0$. Une solution pour F_2 est $\{f4, f5, f6\}$. Il faut donc trouver un 3-cube qui a une intersection de dimension 1 avec [Ø, 123] . Une solution pour K_2 est [Ø, 145].

Par intersection, le groupe d'adjacence G_{12} = ({s1, s2}, {}) se retrouve sur le cube [Ø, 1].

On considère ensuite L_3; L_3 doit être complété par F_3 vérifiant $| F_3 | = 1$. On recherche les relations d'intersection. L_{13} = {s3}, L_{23} = {s6} et $L_{12,3}$ = {}. Ainsi, k_{13} vérifie $0 \leq k_{13} \leq 1$, k_{23} vérifie $0 \leq k_{23} \leq 1$. Au vu de $L_{12,3}$, on ne peut avoir qu'une disjonction entre G_3 et G_{12} (car F_{12} = {}). Pour cela, on effectuera la recherche d'un cube K_3 intersectant avec les deux cubes K_1 et K_2 et disjoint de leur intersection K_{12}; cela n'est pas possible. Il n'y a donc pas de solution pour L_3.

Dans ce cas, on essaie de remettre en cause la solution pour G_2; on cherche une autre solution pour F_2 en prenant $k_{12} = 2$; ceci entraîne $| F_{12} | = 2$. Comme $| F_2 | = 3$ et F_1 = {f1, f2, f3}, une solution pour F_2 est {f1, f2, f4}. Le cube K_2 obtenu pour G_2 est alors égal à [Ø, 124].

On remet à jour G_{12} qui devient ({s1, s2}, {f1, f2}); ce groupe est donc affecté au cube [Ø, 12].

On revient au groupe G_3. Les conditions sur k_{12} et k_{13} ne sont pas altérées. Par contre, au vu de G_{12}, on peut maintenant choisir une disjonction (qui n'amènera à aucune solution pour les mêmes raisons que précédemment), ou une intersection de dimension $k_{12,3} = 0$. Cette dernière solution sera prise. On recherche une combinaison valide; la seule qui marche est $k_{12} = 1$, $k_{13} = 1$ et $k_{12,3} = 0$, ce qui se traduit par la recherche de F3 vérifiant $| F_3 | = 1$, $| \{f1, f2, f3\} \cap F_3 | = 1$, $| \{f1, f2, f4\} \cap F_3 | = 1$ et $| \{f1, f2\} \cap F_3 | = 1$. Une solution existe; c'est F_3 = {f1}. Il convient maintenant de trouver un 2-cube ayant des intersections de dimension 1, 1 et 0 respectivement avec [Ø, 123], [Ø, 124] et [Ø, 12]; c'est le cube [Ø, 34].

La remise en cause de la solution pour G_2 a donc permis de réussir l'immersion de G_3.

4.1.6.4. Détermination du code de chaque état

Une fois l'immersion terminée, on obtient la liste des groupes d'adjacence pour lesquels l'immersion a réussi de même que le cube associé à chacun de ces groupes. Il convient maintenant au vu de cette solution de retrouver le code de chaque état. Ceci est fait trivialement en considérant chaque état comme étant un

groupe d'adjacence. Il suffira alors de calculer les relations d'intersection ou de disjonction existant entre ce nouveau groupe d'adjacence et les groupes d'adjacence immergés. Ceci est effectué simplement en cherchant, pour chaque groupe d'adjacence, si cet état lui appartient ou pas. Il reste donc à rechercher un 0-cube ayant une intersection de dimension 0 avec le cube associé à tout groupe d'adjacence contenant cet état, et disjoint du cube associé à tout groupe d'adjacence où il n'apparaît pas. Deux cas peuvent alors se produire, soit la solution est unique, soit il existe plusieurs codes possibles pour cet état. Dans ce dernier cas, les codes obtenus sont soit restreints au cube associé à un des groupes d'adjacence, soit disjoints de tous les cubes associés aux groupes d'adjacence immergés. Cela dépendra du fait que l'état appartient à un des groupes d'adjacence immergés ou pas.

exemple

Considérons les trois groupes d'adjacence suivants: $G_1 = \{s1, s2, s3, s4\}$, $G_2 = \{s1, s5\}$ et $G_3 = \{s2, s6\}$. Supposons de plus que l'ensemble des états de l'automate comporte aussi les états s7 et s8.

Sans entrer dans les détails, on donne les cubes associés aux trois premiers groupes d'adjacence: $K_1 = [\emptyset, 12]$, $K_2 = [\emptyset, 3]$, $K_3 = [12, 123]$. On obtient ainsi par intersection les groupes d'adjacence intermédiaires G_{12} et G_{13}. La solution complète de l'immersion est décrite ci-après.

$G_1 = \{s1, s2, s3, s4\} \mapsto [\emptyset, 12]$
$G_2 = \{s1, s5\} \mapsto [\emptyset, 3]$
$G_{12} = \{s1\} \mapsto [\emptyset, \emptyset]$
$G_3 = \{s2, s6\} \mapsto [12, 123]$
$G_{13} = \{s2\} \mapsto [12, 12]$

Pour trouver le code de s5, par exemple, il faut rechercher un 0-cube ayant une intersection de dimension 0 avec [Ø, 3], mais disjoint de [Ø, 12], [Ø, Ø], [12, 123] et [12, 12]. L'unique solution est le 0-cube [3, 3], qu'on note [3] par simplicité. On procède de la même façon pour les autres états; on obtient ainsi le résultat suivant:

s1 → [Ø], s2 → [12], s3 → [1] ou [2], s4 → [1] ou [2],
s5 → [3], s6 → [123], s7 → [13] ou [23], s8 → [13] ou [23]

On note que les états s1, s2, s5 et s6 ont des positions bien déterminées sur l'hypercube. Les états s5 et s6 sont restreints à deux positions au sein du cube [Ø, 12]; ce cube est associé au groupe d'adjacence G_1 auquel ils appartiennent tous deux. Par contre, les états s7 et s8 se retrouvent sur des sommets disjoints de tous les cubes obtenus après immersion, à savoir [13] et [23].

Lorsqu'on immerge un groupe d'adjacence, il importe seulement que les états de ce groupe se retrouvent sur un sous-cube de l'hypercube. Une fois cette condition vérifiée, la position des états à l'intérieur du sous-cube est sans importance, hormis le respect des intersections qui peuvent exister avec d'autres groupes d'adjacence. Ainsi, s3, par exemple, peut se retrouver indifféremment sur le sommet [1] ou [2] pour l'immersion de {s1, s2, s3, s4}. Néanmoins, l'affectation de ce sommet à la position [1], par exemple, peut favoriser de meilleurs simplifications qu'à la position [2]. La position unique d'un état dans l'hypercube sera ainsi déterminée dans un deuxième temps au vu des attractions entre états calculées lors de la détection des situations à factorisation potentielle de monômes.

4.1.6.5. Choix de l'état de code 0

Le noeud associé à l'état de code 0 est le seul noeud qui n'est actif pour aucune variable interne. Cela veut dire que les monômes apparaissant sur les arcs entrants de ce noeud ne sont utilisés dans aucune équation des variables internes. Néanmoins, certains de ces arcs peuvent être actifs pour une sortie dans le cas d'un automate de Mealy. Dans ce cas, seuls les monômes apparaissant sur les arcs non étiquetés par une sortie n'apparaissent dans aucune équation des variables internes et de sortie. Ainsi, pour un automate de Moore, l'état qui est codé 0 est celui qui correspond au noeud avec le plus grand nombre d'arcs entrants. Pour un automate de Mealy, c'est l'état correspondant au noeud ayant le plus d'arcs entrants non étiquetés par une sortie qui aura le code 0.

4.2. IMMERSION DES ENSEMBLES D'ÉTATS À FACTORISATION POTENTIELLE DE MONOMES

Après l'immersion des groupes d'adjacence, certains états n'ont pas un code unique possible. Ce problème sera résolu au vu des situations à factorisation. Pour cela, on a défini des attractions entre états qui permettront d'affecter un code unique à chaque état; pour ce faire, des états ayant une forte attraction entre eux seront donné des codes proches au sens de la distance de Hamming. Cette partie montrera comment calculer ces attractions, et comment au vu de ces attractions, les codes seront affectés aux états n'ayant pas encore un code unique.

4.2.1. Calcul des attractions

On rappelle les situations considérées: ce sont (1) les arcs à entrées intersectantes, (2) les noeuds à sortance multiple, (3) les noeuds à entrance multiple et (4) les arcs actifs pour les sorties d'un automate de Mealy avec entrées intersectantes. L'attraction entre deux états s_i et s_j au sein de la $k^{ième}$ situation est notée $Gain_k(s_i, s_j)$. L'attraction globale entre deux états s_i et s_j est notée $ATT(s_i, s_j)$. L'algorithme de calcul des attractions est décrite sommairement dans ce qui suit; il reçoit en entrée les ensembles S et O des états et des sorties de l'automate, et calcule l'ensemble ATT des attractions pour tout couple d'états. L'algorithme cumule les attractions de tout couple d'état à chaque fois qu'ils sont impliqués dans une même situation.

```
procédure Calcul_Attraction(S, O, ATT)
|  pour tout état s ∈ S faire
|  |  pour tout couple d'états (s_i, s_j) successeurs de s faire
|  |  |  ATT(s_i, s_j) ← ATT(s_i, s_j) + Gain_2(s_i, s_j)
|  |  finpour
|  finpour
|
|  pour tout état s ∈ S faire
|  |  pour tout couple d'états (s_i, s_j) prédécesseurs de s faire
|  |  |  ATT(s_i, s_j) ← ATT(s_i, s_j) + Gain_3(s_i, s_j)
|  |  finpour
|  finpour
|
|  pour tout o ∈ O faire
|  |  pour tout couple d'états (s_i, s_j) en amont d'arcs actifs pour o faire
|  |  |  ATT(s_i, s_j) ← ATT(s_i, s_j) + Gain_4(s_i, s_j)
|  |  finpour
|  finpour
|
|  pour tout couple d'états (s_i, s_j) successeurs de s faire
|  |  pour tout couple d'états (s_k, s_l) resp. successeurs de s_i et de s_j faire
|  |  |  ATT(s_i, s_j) ← ATT(s_i, s_j) + Gain_1(s_i, s_j)
|  |  |  ATT(s_k, s_l) ← ATT(s_k, s_l) + Gain_1(s_k, s_l)
|  |  finpour
|  finpour
fin Calcul_Attraction
```

4.2.2. Obtention d'un code unique pour chaque état

Une fois, les attractions calculées, on peut déterminer le code de chaque état. On souhaiterait que les codes de deux états soient d'autant plus proches que l'attraction entre ces deux états est forte. Cela reviendra à maximiser la fonction objective $\Sigma_{\{i \neq j\}}$ ATT(s_i, s_j)*Proximité(C(s_i), C(s_j)), où Proximité(c, c') est le nombre de bits identiques entre deux codes c et c'. Or, Proximité(c, c') est égal à N - H(c, c'), où H est la distance de Hamming. Le problème se ramène donc à minimiser la fonction objective $\Sigma_{\{i \neq j\}}$ ATT(s_i, s_j)*H(C(s_i), C(s_j)). La

résolution de ce problème est connue comme étant NP-complète [ARM62a]. C'est pour cette raison que nous utiliserons une heuristique de type glouton. L'algorithme qui permet de figer le code de chaque état est décrit ci-après. Il reçoit comme entrées l'ensemble ECU des états à code unique, l'ensemble ECM des états à codes multiples et CP l'ensemble des ensembles de codes possibles pour chaque état. Il met à jour l'ensemble C des codes des états.

```
procédure Figer_Code(ECU, ECM, CP, C)
  tant que ECM ≠ Ø faire
    choisir s ∈ ECM maximisant Σ{s' ∈ ECU} ATT(s, s')
    choisir c ∈ CP(s) minimisant Σ{s' ∈ ECU} ATT(s, s')*H(c, C(s')
    ECM ← ECM/{s}; ECU ← ECU ∪ {s}; C(s) ← c;
    pour tout s' ∈ ECM faire
      CP(s') ← CP(s')/{c}
    finpour
    tant que (∃s" tel que | CP(s") | = 1) faire
      ECM ← ECM/{s"}; ECU ← ECU ∪ {s"}; C(s") ← CP(s")
      pour tout s' ∈ ECM faire
        CP(s') ← CP(s')/{C(s")}
      finpour
    fintant
  fintant
fin Figer_Code
```

4.2.3. Explication de l'algorithme

Supposons que certains états aient déjà des codes uniques. On recherche l'état s qui est le plus fortement attiré par ces états fixés. Ensuite, on choisit parmi les codes possibles de s celui qui permettra de minimiser la fonction objective au vu des états déjà fixés. L'état s est enlevé de ECM et rajouté à ECU. Le code de s, C(s), est mis à jour. Ce code est ensuite enlevé de l'ensemble des codes possibles de chaque état de ECM. Ceci peut avoir pour effet de fixer le code d'un autre état. En effet, supposons que les codes possibles pour un état s" donné de ECM soient c1 et c2, et que c1 soit affecté à un autre état. Alors, le

seul code possible pour s" devient c2, ce qui se traduit par | CP(s") | = 1. Le choix de c2 pour s" peut avoir le même effet pour un autre état; on continuera la mise-à-jour tant que des états se retrouveront avec des codes uniques.

On réitère alors la boucle tant qu'il y a des états qui n'ont pas de code unique.

Si, au départ, il n'y a pas d'état à code unique, alors on choisira l'état pour lequel la somme des attractions avec les autres états est la plus grande. On fixera son code arbitrairement à un de ses codes possibles.

exemple

Supposons que les ensembles de codes associés aux états soient les suivants:

CP(s1) ↦ {[Ø]}
CP(s2) ↦ {[2], [12]}
CP(s3) ↦ {[1]}
CP(s4) ↦ {[2], [12]}
CP(s5) ↦ {[3], [13]}
CP(s6) ↦ {[3], [13], [23], [123]}
CP(s7) ↦ {[3], [13], [23], [123]}

Au départ ECU = {s1, s3} et ECM = {s2, s4, s5, s6, s7}. Les attractions entre les différents couples d'états sont données dans le tableau suivant.

	s1	s2	s3	s4	s5	s6	s7
s1	-	10	8	9	12	11	7
s2	10	-	12	8	9	12	4
s3	8	12	-	7	6	10	8
s4	9	8	7	-	8	8	8
s5	12	9	6	8	-	9	12
s6	11	12	10	8	9	-	11
s7	7	4	8	8	12	11	-

Figure IV-3: Tableau des attractions entre états

Les attractions avec les états de ECU sont 22, 16, 18, 21 et 15 respectivement pour s2, s4, s5, s6 et s7.

s2 est donc choisi; les codes possibles de s2 sont [2] et [12]. La valeur de la fonction objective partielle au vu de ECU pour chacun des codes est donnée par

code [2] : 10*H([Ø],[2]) + 12*H([1],[2]) = 34
code [12] : 10*H([Ø],[12]) + 12*H([1],[12]) = 32

Le code choisi pour s2 est [12]. ECU devient {s1, s2, s3} et ECM devient {s4, s5, s6, s7}. CP est mis à jour, ce qui donne

CP(s1) ↦ {[Ø]}
CP(s2) ↦ {[12]}
CP(s3) ↦ {[1]}
CP(s4) ↦ {[2]}
CP(s5) ↦ {[3], [13]}
CP(s6) ↦ {[3], [13], [23], [123]}
CP(s7) ↦ {[3], [13], [23], [123]}

On note que le code de s4 devient unique. ECU devient alors {s1, s2, s3, s4}, et ECM devient {s5, s6, s7}. CP n'est pas modifié.

Les attractions avec les états de ECU sont 35, 41 et 27 respectivement pour s5, s6 et s7.

s6 est choisi. On calcule de la même manière la valeur de la fonction objective partielle pour chacun des codes de s6; ceci donne respectivement 83, 80, 84 et 81 pour [3], [13], [23] et [123].

Le code de s6 est donc pris égal à [13]. Ceci impose le code [3] pour s5. Après mise-à-jour, on obtient ECU = {s1, s2, s3, s4, s5, s6}, ECM = {s7}. CP devient

CP(s1) ↦ {[Ø]}
CP(s2) ↦ {[2]}
CP(s3) ↦ {[1]}
CP(s4) ↦ {[12]}
CP(s5) ↦ {[3]}
CP(s6) ↦ {[13]}
CP(s7) ↦ {[23], [123]}

Finalement, on calcule la valeur de la fonction objective partielle pour chacun des codes de s7; ceci donne respectivement 88 et 92 pour [23] et [123]. [23] est donc choisi comme code de s7.

4.3. CONCLUSION

Nous avons présenté dans ce chapitre deux algorithmes pour la génération des codes des états. Le premier concerne l'immersion des groupes d'adjacence dans des sous-cubes de l'hypercube. Il est montré comment chaque groupe

d'adjacence est étendu au vu des groupes déjà traités. Ce groupe est alors affecté à un sous-cube dont la recherche est effectuée en considérant les relations d'intersection et de disjonction avec les groupes déjà traités. Les différentes extensions possibles de chaque groupe et les divers cubes générés pour celui-ci constituent deux types de recherche-arrière. Ces recherches sont accélérées par l'utilisation de conditions nécessaires et/ou suffisantes de la théorie des cubes intersectants. Le deuxième algorithme calcule les attractions entre couples d'états. Puis il en effectue l'immersion en considérant pour chaque état l'ensemble de ses codes possibles; le code minimisant une fonction objective liée à l'attraction entre les états est choisi pour cet état. Le processus est répété jusqu'à que tous les états soient codés. Finalement, l'état qui aura pour code 0 est choisi, et les codes des autres états sont remis à jour.

CHAPITRE V

RÉSULTATS ET ÉVALUATIONS

Ce chapitre a pour objet de montrer les résultats obtenus en utilisant le logiciel de codage optimisé qui a été intégré dans le système ASYL. Les évaluations ont été faites sur des exemples d'automates provenant en majorité des benchmarks proposés par le Centre Micro-électronique de Caroline du Nord (MCNC) [MCNC87]. Tous les tableaux présentés rappelleront les caractéristiques de ces automates dans les quatre premières colonnes; I donne le nombre d'entrées, O le nombre de sorties, S le nombre d'états et P le nombre de transitions dans ces automates. Trois types de comparaison seront effectués; le premier concerne la comparaison d'un codage optimisé par rapport à un codage aléatoire pour une réalisation sur cellules standards, le deuxième concerne la comparaison du codage d'ASYL à celui de JEDI, et le troisième évalue l'effet d'un codage optimisé pour des cibles autres que des cellules standards.

5.1. ETUDE DE L'EFFICACITÉ DU CODAGE OPTIMISÉ PAR RAPPORT À UN CODAGE ALÉATOIRE

Pour cette comparaison, nous avons voulu évaluer l'efficacité du codage optimisé d'ASYL pour quatre stratégies différentes.

5.1.1. Etude de l'apport des situations à fusion de monômes

Nous avons d'abord recherché uniquement les situations à fusion de monômes (L1) et évalué le gain obtenu par rapport à un codage aléatoire (moyenne de cinq codages aléatoires).

5.1.2. Etude de l'apport des situations à factorisation

Pour ces situations, nous avons différencié trois cas:

α) Seules les situations créant des expressions algébriques communes à plusieurs fonctions (les noeuds à entrance multiple, les arcs actifs pour une sortie avec entrées intersectantes) sont considérées après détection des situations de type L1. Dans ce cas, on parlera de stratégie L2a.

β) Seules les situations créant des expressions algébriques avec sous-expressions communes à plusieurs fonctions (les arcs à entrées intersectantes, les noeuds à sortance multiple) sont considérées après détection des situations de type L1. Dans ce cas, on parlera de stratégie L2b.

γ) toutes les situations à factorisation sont considérées après détection des situations de type L1. Dans ce cas, on parlera de stratégie L2c.

Cette distinction est faite toujours dans l'idée de ne pas chercher à mettre immédiatement en place trop de facteurs communs pouvant nuire à la connectique.

Les tableaux 1 et 2 montrent les résultats obtenus pour une réalisation sur cellules standards après codage optimisé en considérant les quatre stratégies précitées L1, L2a, L2b et L2c. Ils donnent respectivement les surfaces globales (en mils^2) et les facteurs de routage (rapport surface des connexions/surface des cellules) des circuits finaux après synthèse par le logiciel de VLSI Technology Inc. en utilisant la bibliothèque de cellules standards VSC320 1,2μ. Les résultats d'un codage aléatoire sont utilisés comme base de comparaison; ils ont été obtenus en faisant la moyenne des résultats obtenus pour cinq codages aléatoires différents. Le tableau 3 récapitule les pourcentages de gain obtenus pour la surface globale et le facteur de routage par rapport au codage aléatoire; le gain est évalué selon la formule:

$$\text{Gain} = (\text{Surface}_{\text{Aléatoire}} - \text{Surface}_{\text{Optimisé}})/\text{Surface}_{\text{Aléatoire}} * 100\%$$

Au vu de ces tableaux, on note un gain global en surface de 31,4%, 37,9%, 37,8% et 35,3% respectivement pour les stratégies L1, L2a, L2b et L2c. Le gain global est obtenu en calculant la surface totale pour une stratégie et celle pour le codage aléatoire, et en évaluant le gain selon la formule précédente. Le gain moyen en surface est égal à 27,9%, 32,2%, 31,2% et 31,3% pour les mêmes stratégies respectivement. Pour le facteur de routage, le gain moyen est de 12,5%, 15,2%, 14,6% et 14,1%.

Exemple	Caractéristiques				Surface				
	I	O	S	P	aléa	L1	L2a	L2b	L2c
arbiter	9	9	20	46	281,2	126,0	126,8	126,2	161,3
bbara	4	2	10	60	131,1	86,6	83,1	93,8	83,1
bbsse	7	7	16	56	255,3	192,8	187,7	179,9	190,3
bbtas	2	2	6	24	38,7	30,0	27,0	29,9	28,9
cse	7	7	16	91	430,0	342,8	324,8	324,8	308,8
dk14	8	5	7	56	205,3	205,7	145,8	140,4	143,8
dk16	4	3	27	108	683,5	522,8	441,7	441,7	441,5
dk17	4	3	8	32	118,8	90,2	93,8	93,8	92,1
dk27	2	2	7	14	40,6	31,1	35,3	35,3	32,1
donfile	2	1	24	96	424,6	206,2	174,3	150,4	173,3
esd	14	7	21	154	704,0	362,3	483,9	502,7	530,7
ex1	9	19	20	138	534,3	380,3	316,9	388,0	330,7
ex4	6	9	14	21	132,7	102,8	84,8	111,7	90,2
ex6	5	8	8	34	190,5	176,2	148,5	148,5	143,1
jay	4	33	58	181	1403,2	856,8	666,7	682,7	847,7
keyb	7	2	19	170	589,8	341,3	380,5	320,6	317,0
kirkman	12	6	17	370	358,1	259,8	228,8	228,8	228,8
lion9	2	1	9	25	80,5	45,7	53,7	53,7	53,7
mark1	5	16	15	22	154,6	131,9	102,4	130,6	110,9
modulo12	1	1	12	24	48,1	42,2	42,2	42,2	42,2
opus	5	6	11	22	161,4	99,2	97,0	97,0	96,9
planet	7	19	48	115	1681,1	1604,0	1431,6	1258,4	1338,2
s1	8	6	20	107	1060,2	594,4	602,8	613,0	603,4
s1a	8	6	20	107	743,9	341,8	412,5	347,6	408,9
s8	4	1	5	20	47,7	41,1	41,1	41,1	41,1
sand	11	9	32	184	1496,8	1371,8	1295,4	1204,1	1328,3
scf	27	56	121	166	3314,0	2866,0	2187,2	2518,6	-
sse	7	7	16	56	246,2	200,8	187,7	198,2	190,3
styr	9	10	30	166	1570,1	924,9	914,9	871,9	939,0
tav	4	4	4	49	35,3	33,2	33,2	33,2	33,2
tbk	6	3	32	1569	2012,7	538,1	542,9	500,6	542,1
train11	2	1	11	25	66,1	57,0	52,9	52,9	52,9

Tableau 1: Comparaison de surfaces globales (différentes options pour ASYL)

Exemple	Caractéristiques				Facteur de Routage				
	I	O	S	P	aléa	L1	L2a	L2b	L2c
arbiter	9	9	20	46	1,05	0,82	0,75	0,72	0,92
bbara	4	2	10	60	0,84	0,64	0,68	0,70	0,68
bbsse	7	7	16	56	0,99	0,82	0,84	0,78	0,86
bbtas	2	2	6	24	0,56	0,47	0,43	0,59	0,54
cse	7	7	16	91	1,18	1,17	1,10	1,10	1,16
dk14	8	5	7	56	0,92	0,83	0,84	0,77	0,84
dk16	4	3	27	108	1,45	1,33	1,23	1,23	1,20
dk17	4	3	8	32	0,82	0,74	0,70	0,70	0,71
dk27	2	2	7	14	0,59	0,52	0,51	0,51	0,47
donfile	2	1	24	96	1,22	0,96	0,94	0,87	0,91
esd	14	7	21	154	1,42	1,01	1,11	1,19	1,31
ex1	9	19	20	138	1,23	1,03	0,86	1,08	0,92
ex4	6	9	14	21	0,81	0,71	0,61	0,71	0,70
ex6	5	8	8	34	0,88	1,00	0,87	0,87	0,80
jay	4	33	58	181	1,90	1,57	1,35	1,25	1,44
keyb	7	2	19	170	1,26	0,99	1,08	0,93	0,89
kirkman	12	6	17	370	1,02	0,91	0,82	0,82	0,82
lion9	2	1	9	25	0,75	0,62	0,76	0,76	0,76
mark1	5	16	15	22	1,05	0,93	0,76	1,02	0,88
modulo12	1	1	12	24	0,64	0,64	0,64	0,64	0,64
opus	5	6	11	22	0,88	0,71	0,64	0,64	0,64
planet	7	19	48	115	1,91	1,81	1,82	1,72	1,67
s1	8	6	20	107	1,58	1,23	1,48	1,28	1,28
s1a	8	6	20	107	1,57	1,15	1,36	1,22	1,34
s8	4	1	5	20	0,67	0,61	0,61	0,61	0,61
sand	11	9	32	184	1,71	1,71	1,52	1,49	1,60
scf	27	56	121	166	2,81	2,76	2,64	2,33	-
sse	7	7	16	56	1,00	0,95	0,84	0,92	0,86
styr	9	10	30	166	1,73	1,53	1,48	1,54	1,48
tav	4	4	4	49	0,61	0,53	0,53	0,53	0,56
tbk	6	3	32	1569	1,81	1,31	1,34	1,24	1,31
train11	2	1	11	25	0,69	0,73	0,67	0,67	0,67

Tableau 2: Comparaison de facteurs de routage (différentes options pour ASYL)

Exemple	Caractéristiques				Surface				Facteur de Routage			
	I	O	S	P	L1	L2a	L2b	L2c	L1	L2a	L2b	L2c
arbiter	9	9	20	46	55,2	54,9	55,1	42,6	21,9	28,6	31,4	12,4
bbara	4	2	10	60	33,9	36,6	28,5	36,6	23,8	19,0	16,7	19,0
bbsse	7	7	16	56	24,5	26,5	29,5	25,5	17,2	15,2	21,2	13,1
bbtas	2	2	6	24	22,5	30,2	22,7	25,3	16,1	23,2	-5,40	3,6
cse	7	7	16	91	20,3	24,5	24,5	28,2	0,84	6,8	6,80	1,7
dk14	8	5	7	56	-0,1	29,0	31,6	30,0	9,80	8,7	16,3	8,7
dk16	4	3	27	108	23,5	35,4	35,4	35,4	8,30	15,2	15,2	17,2
dk17	4	3	8	32	24,1	21,0	21,0	22,5	9,80	14,6	14,6	13,4
dk27	2	2	7	14	23,4	13,1	13,1	20,9	11,9	13,6	13,6	20,3
donfile	2	1	24	96	51,4	58,9	64,6	59,2	21,3	23,0	28,7	25,4
esd	14	7	21	154	48,5	31,3	28,6	24,6	28,9	21,8	16,2	7,7
ex1	9	19	20	138	28,8	40,7	27,4	38,1	16,3	30,1	12,2	25,2
ex4	6	9	14	21	22,5	36,1	15,8	32,0	12,3	24,7	12,3	13,6
ex6	5	8	8	34	7,5	22,0	22,0	24,9	-13,6	1,1	1,1	9,1
jay	4	33	58	181	38,9	52,5	51,3	39,6	17,4	28,9	34,2	24,2
keyb	7	2	19	170	42,1	35,5	45,6	46,3	21,4	14,3	26,2	29,4
kirkman	12	6	17	370	27,5	36,1	36,1	36,1	10,8	19,6	19,6	19,6
lion9	2	1	9	25	43,2	33,3	33,3	33,3	17,3	-1,3	-1,3	-1,3
mark1	5	16	15	22	14,7	33,8	15,5	28,3	11,4	27,6	2,9	16,2
modulo12	1	1	12	24	12,3	12,3	12,3	12,3	0,0	0,0	0,0	0,0
opus	5	6	11	22	38,5	39,9	39,9	40,0	19,3	27,3	27,3	27,3
planet	7	19	48	115	4,6	14,8	25,1	20,4	5,2	4,7	9,90	12,6
s1	8	6	20	107	43,9	43,1	42,2	43,1	22,2	6,3	19,0	19,0
s1a	8	6	20	107	54,1	44,5	53,3	45,0	26,8	13,4	22,3	14,6
s8	4	1	5	20	13,8	13,8	13,8	13,8	9,0	9,0	9,0	9,0
sand	11	9	32	184	18,4	13,5	19,6	11,3	0,0	11,1	12,9	6,4
scf	27	56	121	166	13,5	34,0	24,0	24,0	1,8	6,0	17,1	17,1
sse	7	7	16	56	18,4	23,8	19,5	22,7	5,0	16,0	8,00	14,0
styr	9	10	30	166	41,1	41,7	44,5	40,2	11,6	14,5	11,0	14,5
tav	4	4	4	49	5,9	5,90	5,90	5,9	13,1	13,1	13,1	8,2
tbk	6	3	32	1569	73,3	73,0	75,1	73,1	27,6	26,0	31,5	27,6
train11	2	1	11	25	13,8	20,0	20,0	20,0	-5,8	2,9	2,90	2,9
Moyenne					**27,9**	**32,2**	**31,2**	**31,3**	**12,5**	**15,2**	**14,6**	**14,1**

Tableau 3: Pourcentage des gains en surface et en facteur de routage

Deux remarques principales s'imposent au vu de ces chiffres:

1) La stratégie L1 est responsable de la plus grande partie de l'optimisation apportée, mais n'est pas suffisante. En effet, si nous considérons par exemple la stratégie L2a, nous notons qu'un gain de 27,9% est obtenu pour la phase d'optimisation L1 avec un gain total de 32,2% après la phase d'optimisation L2a.

Cette deuxième phase est donc nécessaire pour compléter le gain de la phase L1 mais n'apporte qu'une contribution minime. Le tableau suivant (tableau 4) compare les meilleurs résultats obtenus pour l'une des stratégies L2 aux résultats de la stratégie L1, confirmant le faible apport, mais néanmoins utile, des stratégies L2.

Exemple	Caractéristiques				ASYL L1			ASYL (meilleur L2)		
arbiter	9	9	20	46	126,0	0,82	10,9	126,2	0,72	9,9
bbara	4	2	10	60	86,6	0,64	8,7	83,1	0,68	8,7
bbsse	7	7	16	56	192,8	0,82	10,3	179,9	0,78	9,9
bbtas	2	2	6	24	30,0	0,47	7,3	27,0	0,43	7,3
cse	7	7	16	91	342,8	1,17	6,4	308,8	1,16	6,4
dk14	8	5	7	56	205,7	0,83	10,0	140,3	0,77	10,0
dk16	4	3	27	108	522,8	1,33	13,3	441,5	1,20	13,3
dk17	4	3	8	32	90,2	0,74	7,3	92,1	0,71	7,3
dk27	2	2	7	14	31,1	0,52	5,3	32,1	0,47	5,3
donfile	2	1	24	96	206,2	0,96	10,4	150,4	0,87	8,5
esd	14	7	21	154	362,3	1,01	17,3	483,9	1,11	16,3
ex1	9	19	20	138	380,3	1,03	14,2	316,9	0,86	12,3
ex4	6	9	14	21	102,8	0,71	7,6	84,8	0,61	6,1
ex6	5	8	8	34	176,2	1,00	9,2	143,1	0,80	9,2
jay	4	33	58	181	856,8	1,57	18,8	666,7	1,35	19,7
keyb	7	2	19	170	341,3	0,99	17,9	317,0	0,89	17,9
kirkman	12	6	17	370	259,8	0,91	11,1	228,8	0,82	11,1
lion9	2	1	9	25	45,7	0,62	5,9	53,7	0,76	4,3
mark1	5	16	15	22	131,9	0,93	9,5	102,4	0,76	7,0
modulo12	1	1	12	24	42,2	0,64	5,5	42,2	0,64	5,5
opus	5	6	11	22	99,2	0,71	9,2	96,9	0,64	9,2
planet	7	19	48	115	1604,0	1,81	18,5	1258,4	1,72	18,3
s1	8	6	20	107	594,4	1,23	14,5	602,8	1,48	14,5
s1a	8	6	20	107	341,8	1,15	11,9	347,6	1,22	11,9
s8	4	1	5	20	41,1	0,61	5,1	41,1	0,61	5,1
sand	11	9	32	184	1371,8	1,71	20,9	1204,1	1,49	18,7
scf	27	56	121	166	2850,0	2,75	22,3	2187,2	2,64	18,4
sse	7	7	16	56	200,8	0,95	10,3	187,7	0,84	9,9
styr	9	10	30	166	924,9	1,53	20,0	871,9	1,54	16,1
tav	4	4	4	49	33,2	0,53	4,6	33,2	0,53	4,6
tbk	6	3	32	1569	538,1	1,31	14,9	500,6	1,24	14,9
train11	2	1	11	25	57,0	0,73	5,0	52,9	0,67	5,0

Tableau 4: Etude de l'apport des situations à factorisation

2) Pour toutes les stratégies, on note non seulement un gain important en surface mais aussi un gain important en facteur de routage. Ce gain est d'autant plus important qu'il signifie en fait un plus grand gain en surface de routage qu'en surface active (surface des cellules), nous confortant dans notre idée que la prise en compte des situations à fusion de monômes permettrait un meilleur gain en surface de routage. Pour mieux illustrer ceci, considérons l'exemple "arbiter". La surface obtenue pour un codage aléatoire est de 281,2 pour un facteur de routage de 1,05. On en déduit donc une surface active de 137,2 et une surface de routage 144,0. Après le codage de type L1, la surface totale obtenue est de 126,0 comprenant une surface active de 69,2 et une surface de routage de 56,8. On obtient ainsi un gain de 49,6% en surface active et de 60,6% en surface de routage. On note qu'un plus grand gain en surface de routage est obtenu au détriment de la surface active; ceci est dû au fait que les situations de type L1 favorisent une simplification maximale de la logique au sein d'une même fonction au détriment d'une mise en commun de logique entre différentes fonctions.

Les gains en surface active et en surface de routage sont présentés dans le tableau 5. Le gain global en surface active est respectivement de 27,3%, 32,0%, 30,5% et 28,8% pour les stratégies L1, L2a, L2b et L2c, et le gain moyen est respectivement de 23,5%, 26,8%, 26,8% et 26,0%. Le gain global en surface de routage est respectivement de 33,9%, 41,5%, 42,4% et 39,3%, et le gain moyen de 34,0%, 37,4%, 36,8% et 35,9%. On note les plus grands pourcentages de gain pour la surface de routage dans les différents cas.

Exemple	Caractéristiques				Surface Active				Surface de Routage			
	I	O	S	P	L1	L2a	L2b	L2c	L1	L2a	L2b	L2c
arbiter	9	9	20	46	49,5	47,2	46,3	38,8	60,6	62,3	63,1	46,3
bbara	4	2	10	60	25,9	30,6	31,4	30,6	43,5	43,8	42,8	43,8
bbsse	7	7	16	56	17,4	20,5	17,8	20,3	31,6	32,5	35,2	30,7
bbtas	2	2	6	24	17,7	23,9	31,6	24,4	31,0	41,6	27,9	27,1
cse	7	7	16	91	19,9	21,6	21,6	27,5	20,6	26,9	26,9	28,7
dk14	8	5	7	56	-5,1	25,9	23,0	26,9	5,20	32,3	35,5	33,3
dk16	4	3	27	108	19,6	29,0	29,0	28,1	26,2	39,8	39,8	40,5
dk17	4	3	8	32	20,6	15,5	15,5	17,5	28,3	27,8	27,8	28,6
dk27	2	2	7	14	19,9	8,5	8,50	14,5	29,4	20,9	20,9	31,9
donfile	2	1	24	96	45,0	53,0	51,3	52,6	56,7	63,8	65,2	64,6
esd	14	7	21	154	38,0	21,2	24,0	21,0	55,9	38,4	36,3	27,1
ex1	9	19	20	138	21,8	28,9	36,4	28,1	34,5	50,3	44,2	46,2
ex4	6	9	14	21	18,0	28,2	32,4	27,6	28,1	45,9	40,7	37,5
ex6	5	8	8	34	13,1	21,6	21,6	21,5	1,20	22,5	22,5	28,7
jay	4	33	58	181	31,1	41,4	38,8	28,2	43,1	58,3	59,7	45,6
keyb	7	2	19	170	34,3	29,9	24,5	35,7	48,4	39,9	44,2	54,6
kirkman	12	6	17	370	23,3	29,1	29,1	29,1	31,5	43,0	43,0	43,0
lion9	2	1	9	25	38,7	33,7	33,7	33,7	49,3	32,8	32,8	32,8
mark1	5	16	15	22	9,4	22,9	32,8	21,8	19,7	44,2	34,7	34,4
modulo12	1	1	12	24	12,3	12,3	12,3	12,3	12,3	12,3	12,3	12,3
opus	5	6	11	22	32,4	31,1	31,1	31,2	45,5	49,9	49,9	49,9
planet	7	19	48	115	1,2	12,1	8,90	13,2	6,40	16,3	18,0	24,1
s1	8	6	20	107	35,1	40,8	35,7	35,6	49,5	44,6	47,9	47,8
s1a	8	6	20	107	45,1	39,6	35,8	39,6	59,8	47,7	50,1	48,5
s8	4	1	5	20	10,6	10,6	10,6	10,6	18,6	18,6	18,6	18,6
sand	11	9	32	184	8,4	6,93	5,80	7,5	8,40	17,3	17,9	13,5
scf	27	56	121	166	12,4	30,9	24,5	13,0	13,9	35,1	37,4	27,9
sse	7	7	16	56	16,3	17,1	20,6	16,9	20,5	30,4	26,9	28,5
styr	9	10	30	166	36,4	35,9	37,4	34,2	43,8	45,1	44,2	43,7
tav	4	4	4	49	1,0	1,0	1,0	2,9	14,0	14,0	14,0	10,9
tbk	6	3	32	1569	67,5	67,6	66,2	67,2	76,5	76,0	76,8	76,3
train11	2	1	11	25	15,8	19,0	19,0	19,0	10,9	21,4	21,4	21,4
Moyenne					**23,5**	**26,8**	**26,8**	**26,0**	**34,0**	**37,4**	**36,8**	**35,9**

Tableau 5: Pourcentage des gains en surface active et surface de routage

Le tableau 6 indique les chemins critiques correspondant aux quatre stratégies de codage. Le gain moyen obtenu est respectivement de 13,5%, 15,2%, 13,7% et 14,7%.

Exemple	Caractéristiques				Temps	Chemin Critique				
	I	O	S	P		rand	L1	L2a	L2b	L2c
arbiter	9	9	20	46	44,4	12,30	10,9	10,9	9,9	10,9
bbara	4	2	10	60	41,4	10,62	8,7	8,8	8,9	8,8
bbsse	7	7	16	56	29,2	12,42	10,3	9,9	11,7	9,9
bbtas	2	2	6	24	17,8	6,54	7,3	7,3	7,3	7,3
cse	7	7	16	91	83,1	15,70	12,8	14,1	14,1	12,8
dk14	8	5	7	56	28,2	10,32	10,0	10,0	10,0	10,0
dk16	4	3	27	108	162,6	14,52	13,3	13,3	13,3	13,3
dk17	4	3	8	32	3,3	8,02	7,3	7,3	7,3	7,3
dk27	2	2	7	14	3,2	5,64	5,3	5,3	5,3	5,3
donfile	2	1	24	96	229,8	14,14	10,4	8,5	9,8	10,4
esd	14	7	21	154	171,6	17,24	17,3	16,3	16,1	16,3
ex1	9	19	20	138	107,7	16,76	14,2	12,3	14,9	12,3
ex4	6	9	14	21	15,0	8,52	7,6	7,0	6,5	6,1
ex6	5	8	8	34	2,6	12,52	9,2	9,2	9,2	9,2
jay	4	33	58	181	606,1	19,82	18,8	19,7	17,8	17,3
keyb	7	2	19	170	127,9	18,52	17,9	17,9	17,9	17,9
kirkman	12	6	17	370	5,3	15,34	11,1	12,7	11,4	12,7
lion9	2	1	9	25	66,8	6,64	5,9	5,9	5,9	4,3
mark1	5	16	15	22	0,9	8,48	9,5	7,0	8,7	9,2
modulo12	1	1	12	24	21,5	6,78	5,5	5,5	5,5	5,5
opus	5	6	11	22	14,9	9,70	9,2	9,2	9,2	9,2
planet	7	19	48	115	223,5	23,34	18,5	19,8	18,3	22,5
s1	8	6	20	107	217,4	17,70	4,5	14,5	15,4	15,4
s1a	8	6	20	107	115,5	16,44	11,9	11,9	12,1	11,9
s8	4	1	5	20	11,4	5,74	5,1	5,1	5,1	5,1
sand	11	9	32	184	111,2	21,80	20,9	20,9	18,7	20,9
scf	27	56	121	166	24,22	24,2	20,9	22,3	18,4	-
sse	7	7	16	56	30,1	13,02	10,3	9,9	13,7	9,9
styr	9	10	30	166	323,2	21,68	20,0	16,1	18,7	16,6
tav	4	4	4	49	2,6	5,48	4,6	4,6	4,6	6,2
tbk	6	3	32	1569	820,1	31,20	14,9	15,1	16,8	15,1
train11	2	1	11	25	69,0	6,38	5,0	5,0	5,0	5,0
Moyenne						**13,5**	**15,2**	**13,7**	**14,7**	

Tableau 6: Comparaison des chemins critiques (différentes options pour ASYL)

5.2. COMPARAISON DU CODAGE D'ASYL AU CODAGE DE JEDI

Deux types de comparaison seront effectués ici, une comparaison en considérant la meilleure surface globale obtenue pour différentes options de chacun des logiciels, et une comparaison en considérant le meilleur chemin critique.

Pour cela, chaque automate a été codé, d'une part, par trois options de codage de JEDI ("i-dominated", "o-dominated", "io-dominated"), et d'autre part, par les quatre stratégies d'ASYL. Pour le premier type de comparaison, les résultats correspondant à la meilleure surface globale obtenue sont considérés. Pour le second type, ce sont ceux qui correspondent au meilleur chemin critique qui sont choisis. Donc, pour chaque automate, sept codages ont été effectués, et la synthèse pour chacun des codages réalisée.

Le premier tableau (tableau 7) concerne la comparaison en considérant la meilleure surface globale, et le facteur de routage et le chemin critique correspondants. On note la supériorité des résultats d'ASYL en ce qui concerne la surface globale et le facteur de routage. Par contre, les résultats d'ASYL concernant le chemin critique sont légèrement moins bons que ceux de JEDI. Notons quand même que le chemin critique considéré ici ne concerne que les cellules, sans prendre en compte la connectique. Une évaluation plus fine doit être effectuée pour obtenir le chemin critique réel au vu des capacités induites par les fils de connexions. Dans l'ensemble, un gain moyen de 10,6% en surface globale et de 20,1% en facteur de routage est obtenu pour ASYL, contre une perte de 6,3% en chemin critique.

Le deuxième tableau (tableau 8) concerne la comparaison en considérant le meilleur chemin critique, et la surface globale et le facteur de routage correspondants. Ici, on note une amélioration du chemin critique d'ASYL, une perte de seulement 2,7% étant notée. D'un autre côté, le gain moyen en surface globale augmente et devient 14,6% pour un gain moyen en facteur de routage de 21,5%.

Ces différents gains pour les deux cas précités sont représentés dans les graphes des figures V-1 et V-2. Tous les points situés au-dessus de la droite de gain nul de chaque graphe indiquent un meilleur résultat pour ASYL. Ces points sont indiqués en fonction d'une complexité croissante en terme de surface globale trouvée par ASYL.

Exemple	Caractéristiques				JEDI (M.S)			ASYL (M.S)		
arbiter	9	9	20	46	192,8	0,97	7,7	125,9	0,82	10,9
bbara	4	2	10	60	93,8	0,93	8,2	83,1	0,68	8,8
bbsse	7	7	16	56	224,1	1,00	10,8	179,9	0,78	11,7
bbtas	2	2	6	24	32,1	0,74	4,6	27,0	0,43	7,3
cse	7	7	16	91	391,8	1,33	13,1	308,8	1,16	12,8
dk14	8	5	7	56	180,0	0,97	7,5	140,3	0,77	10,0
dk16	4	3	27	108	463,3	1,30	14,4	441,5	1,20	13,3
dk17	4	3	8	32	90,2	0,86	6,6	90,2	0,74	7,3
dk27	2	2	7	14	33,2	0,68	3,8	31,1	0,52	5,3
donfile	2	1	24	96	101,1	0,89	7,3	150,4	0,87	9,8
esd	14	7	21	154	516,0	1,38	13,6	362,3	1,01	17,3
ex1	9	19	20	138	555,0	1,38	12,6	316,9	0,86	12,3
ex4	6	9	14	21	103,8	0,78	6,5	84,8	0,61	7,0
ex6	5	8	8	34	169,2	1,06	11,5	143,1	0,80	9,2
keyb	7	2	19	170	406,2	1,24	14,0	317,0	0,89	17,9
kirkman	12	6	17	370	449,3	1,21	14,3	228,8	0,82	11,1
lion9	2	1	9	25	68,6	0,92	5,1	45,7	0,62	5,9
mark1	5	16	15	22	145,5	1,13	7,9	102,4	0,76	7,0
modulo12	1	1	12	24	48,2	0,83	4,9	42,2	0,64	5,5
opus	5	6	11	22	113,5	0,85	8,0	96,9	0,64	9,2
planet	7	19	48	115	1109,2	1,77	18,6	1258,4	1,72	18,3
s1	8	6	20	107	205,3	0,99	13,1	594,4	1,23	14,5
s1a	8	6	20	107	300,3	1,37	14,6	341,8	1,15	11,9
s8	4	1	5	20	74,7	0,75	6,4	41,1	0,61	5,1
sand	11	9	32	184	1353,0	1,91	20,5	1204,1	1,49	18,7
scf	27	56	121	166	3540,2	3,37	19,6	2187,2	2,64	18,4
sse	7	7	16	56	227,2	1,02	10,8	187,7	0,84	9,9
styr	9	10	30	166	1010,9	1,62	18,0	871,9	1,54	18,7
tav	4	4	4	49	33,2	0,68	4,7	33,2	0,53	4,6
tbk	6	3	32	1569	587,3	1,56	15,8	500,6	1,24	16,8
train11	2	1	11	25	108,9	0,96	7,5	52,9	0,67	5,0

Tableau 7: Comparaison des meilleurs résultats en surface d'ASYL et de JEDI

Exemple	Caractéristiques				JEDI (M.C.C)			ASYL (M.C.C)		
arbiter	9	9	20	46	195,5	1,00	7,7	126,2	0,72	9,9
bbara	4	2	10	60	93,8	0,93	8,2	86,6	0,64	8,8
bbsse	7	7	16	56	250,6	1,13	10,0	187,7	0,84	9,9
bbtas	2	2	6	24	32,1	0,74	4,6	27,0	0,43	7,3
cse	7	7	16	91	391,8	1,33	13,1	308,8	1,16	6,4
dk14	8	5	7	56	180,0	0,97	7,5	140,4	0,77	10,0
dk16	4	3	27	108	482,0	1,45	12,2	441,5	1,20	13,3
dk17	4	3	8	32	90,2	0,86	6,6	90,2	0,74	7,3
dk27	2	2	7	14	33,2	0,68	3,8	31,1	0,52	5,3
donfile	2	1	24	96	101,1	0,89	7,3	174,3	0,94	8,5
esd	14	7	21	154	516,0	1,38	13,6	502,7	1,19	16,1
ex1	9	19	20	138	555,0	1,38	12,6	316,9	0,86	12,3
ex4	6	9	14	21	103,8	0,78	6,5	90,2	0,70	6,1
ex6	5	8	8	34	172,3	1,09	10,5	143,1	0,80	9,2
keyb	7	2	19	170	406,2	1,24	14,0	317,0	0,89	17,9
kirkman	12	6	17	370	499,9	1,21	13,3	259,8	0,91	11,1
lion9	2	1	9	25	68,6	0,92	5,1	53,7	0,76	4,3
mark1	5	16	15	22	145,5	1,13	7,9	102,4	0,76	7,0
modulo12	1	1	12	24	48,2	0,83	4,9	42,2	0,64	5,5
opus	5	6	11	22	113,5	0,85	8,0	96,9	0,64	9,2
planet	7	19	48	115	1109,2	1,77	18,6	1258,4	1,72	18,3
s1	8	6	20	107	417,6	1,26	12,9	594,4	1,23	14,5
s1a	8	6	20	107	385,1	1,48	11,2	341,8	1,15	11,9
s8	4	1	5	20	74,7	0,75	6,4	41,1	0,61	5,1
sand	11	9	32	184	1353,0	1,91	20,5	1204,1	1,49	18,7
scf	27	56	121	166	3540,2	3,37	19,6	2518,6	2,33	18,4
sse	7	7	16	56	253,6	1,16	10,8	187,7	0,84	9,9
styr	9	10	30	166	1378,3	1,96	17,6	914,9	1,48	16,1
tav	4	4	4	49	33,2	0,68	4,7	33,2	0,53	4,6
tbk	6	3	32	1569	587,3	1,56	15,8	538,1	1,31	14,9
train11	2	1	11	25	108,9	0,96	7,5	52,9	0,67	5,0

Tableau 8: Comparaison des meilleurs résultats en chemin critique d'ASYL et de JEDI

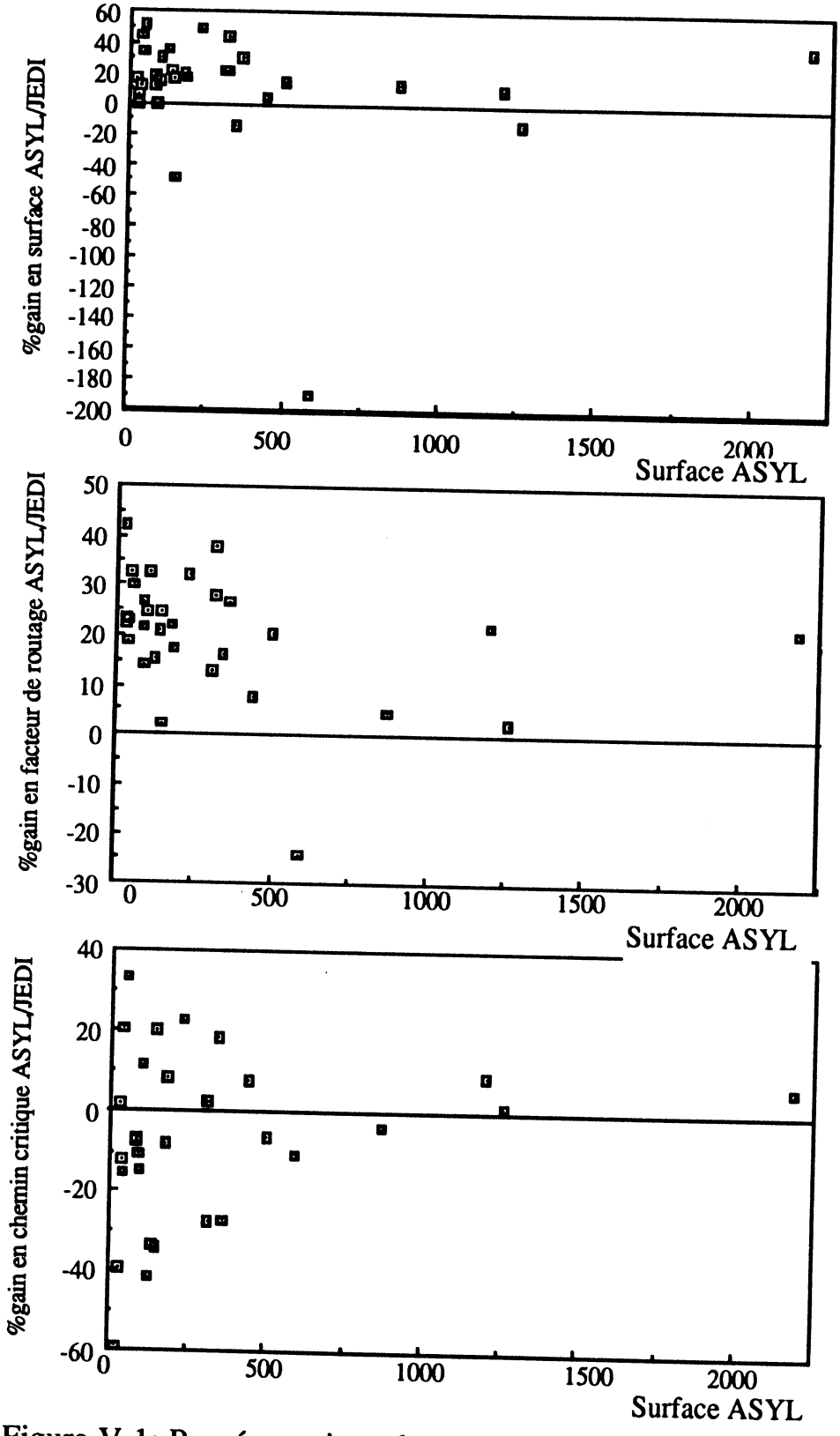

Figure V-1: Représentations des gains d'ASYL par rapport à JEDI en considérant la meilleure surface

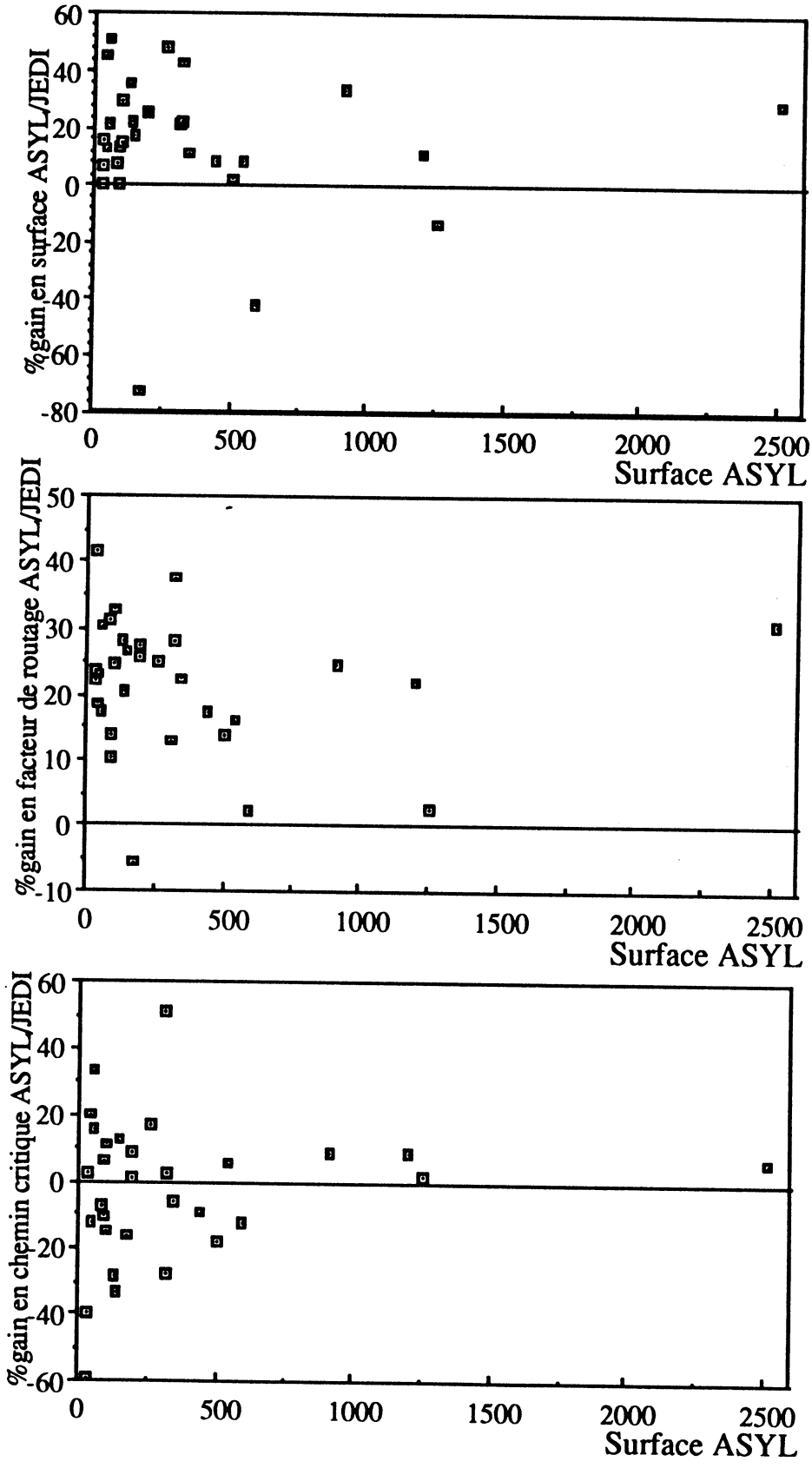

Figure V-2: Représentations des gains d'ASYL par rapport à JEDI en considérant le meilleur chemin critique

5.3. ÉVALUATION DU CODAGE D'ASYL SUR D'AUTRES CIBLES

5.3.1. Circuits Xilinx

La première cible considérée concerne les circuits de type Xilinx. Ces circuits se présentent comme des matrices de cellules identiques (CLB: Configurable Logic Block) pouvant réaliser une ou deux fonctions logiques élémentaires de 3 à 5 variables, et contenant des points de mémorisation. Une représentation d'un circuit Xilinx est donnée dans la figure V-3.

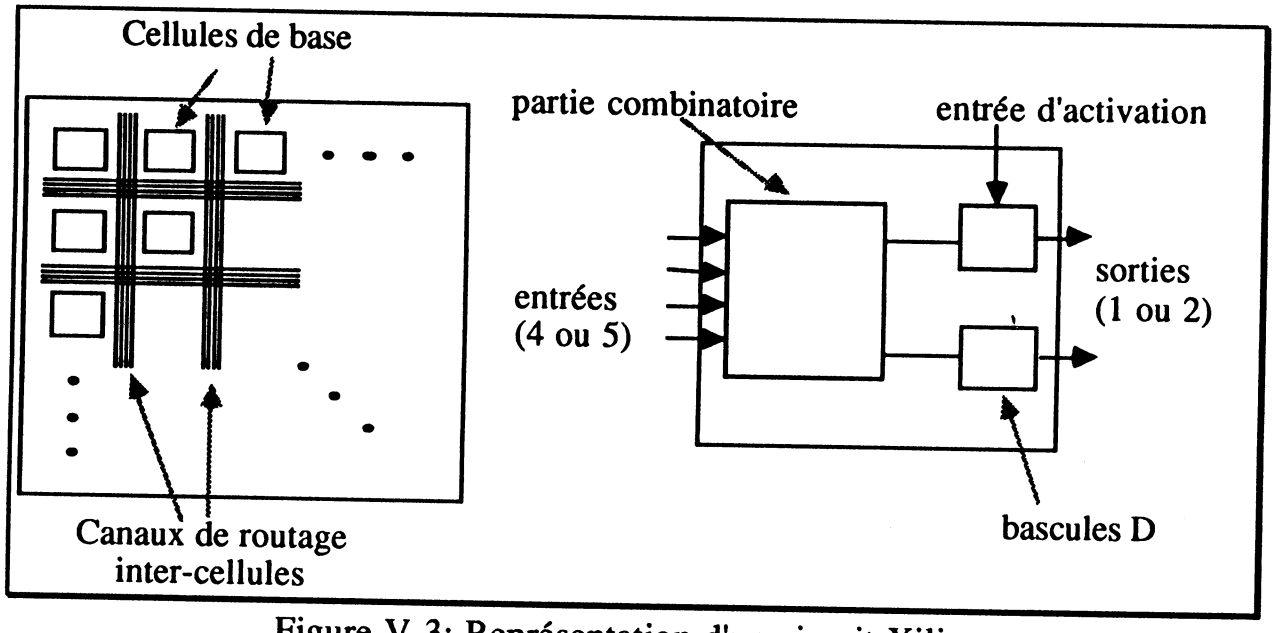

Figure V-3: Représentation d'un circuit Xilinx

L'évaluation du codage a été faite sur un certain nombre de benchmarks de tailles différentes, et les résultats obtenus sont présentés dans le tableau suivant. Le gain moyen en nombre de CLBs est de 24,6% et en niveaux de 5,8%. Ces pourcentages indiquent que le codage optimisé d'ASYL est aussi adapté pour la cible Xilinx.

Comme chaque CLB contient une bascule, on n'est pas limité au codage compact pour un automate, l'augmentation du nombre de bascules ne représentant pas un surcoût en logique. Nous avons voulu effectuer une synthèse de ces automates en utilisant un codage en 1 parmi N ("pire cas") pour voir si on obtenait un gain en logique. Les résultats sont donnés dans le même tableau. Le graphe des points exprimant la différence en nombre de CLBs pour un codage en 1 parmi N par rapport au codage d'ASYL est représentée dans la figure V-4. Les points au-dessus de la droite de différence nulle sont ceux pour lesquels l'utilisation d'un codage 1 parmi N est plus coûteuse. On remarque qu'à partir

d'une certaine taille du circuit, le codage en 1 parmi N fournit une meilleure solution.

Exemple	Caractéristiques				Aléatoire		ASYL		1/N	
	I	O	S	P	CLB	Niv.	CLB	Niv.	CLB	Niv.
arbiter	9	9	20	46	36	3	21	4	-	-
bbara	4	2	10	60	25	4	16	4	22	4
bbsse	7	7	16	56	37	5	27	4	32	5
bbtas	2	2	6	24	4	1	4	1	5	1
cse	7	7	16	91	63	5	44	6	46	6
dk14	8	5	7	56	19	2	21	2	30	3
dk16	4	3	27	108	68	5	52	4	43	4
dk17	4	3	8	32	6	1	6	1	15	3
dk27	2	2	7	14	3	1	3	1	5	1
donfile	2	1	24	96	49	5	19	4	38	3
esd	14	7	21	154	139	8	90	7	61	5
jay	4	33	58	181	135	6	82	5	-	-
keyb	7	2	19	170	106	8	92	8	45	7
kirkman	12	6	17	370	42	7	32	7	46	8
lion9	2	1	9	25	7	2	4	2	13	4
mark1	5	6	15	22	7	2	4	2	-	-
modulo12	1	1	12	24	4	1	4	1	6	1
opus	5	6	11	22	24	4	16	3	18	3
planet	7	19	48	115	178	8	130	7	88	8
s1	8	6	20	107	134	6	84	6	79	4
s1a	8	6	20	107	97	6	45	5	50	4
s8	4	1	5	20	4	2	5	2	17	3
sand	11	9	32	184	177	8	139	7	110	6
scf	27	56	121	166	283	8	200	6	-	-
sse	7	7	16	56	34	4	27	4	-	-
styr	9	10	30	166	178	8	117	7	-	-
tav	4	4	4	49	6	2	6	2	8	3
tbk	6	3	32	1569	226	11	86	8	194	11
train11	2	1	11	25	8	2	7	2	-	-

Tableau 9: Comparaison concernant la cible Xilinx

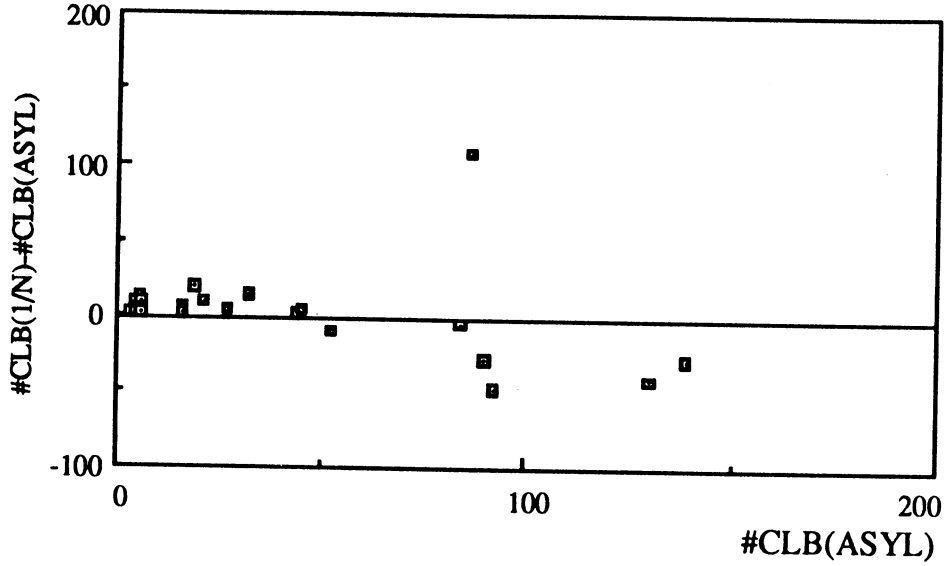

Figure V-4: Comparaison graphique du codage 1 parmi N au codage d'ASYL

L'explication est la suivante: lors de la synthèse d'un automate, on note, d'une manière générale, que la complexité de la logique réalisant les fonctions d'état suivant et de sortie décroît avec l'augmentation du nombre de bascules. Pour une synthèse sur cellules standards, le gain en logique obtenu par l'utilisation d'un plus grand nombre de bascules est contrecarré par la perte en surface due à ces bascules additionnelles. Néanmoins, pour les circuits de type Xilinx, une bascule est disponible pour chaque CLB utilisé. On aurait donc intérêt à utiliser plus de bascules pour diminuer la logique sans que l'utilisation d'une nouvelle bascule nous oblige à prendre un nouveau CLB. Pour de gros automates, même si un codage en 1 parmi N est effectué, la logique demeure importante et nécessite un certain nombre de CLBs. Ces CLBs fournissent néanmoins un nombre de bascules suffisants pour effectuer le codage en 1 parmi N. Par contre, pour les petits automates, utiliser un codage en 1 parmi N impose d'utiliser un nombre de CLBs nécessaire à obtenir les bascules. Néanmoins, la logique pour ces petits automates ne nécessitent pas un tel nombre de CLBs; certains CLBs sont seulement utilisés pour leurs bascules. Il devient alors évident que le nombre de bascules à utiliser pour les automates est "proportionnel" à leur complexité.

5.3.2. Circuits de type PAL

La deuxième cible considérée concerne les circuits de type PAL sous forme de boîtiers du commerce. Ce sont des dispositifs physiques programmables qui réalisent des fonctions sous forme de sommes de monômes, où aucun monôme

n'est partagé entre fonctions. Certains boîtiers peuvent contenir des points de mémorisation. Ils sont représentés dans la figure suivante.

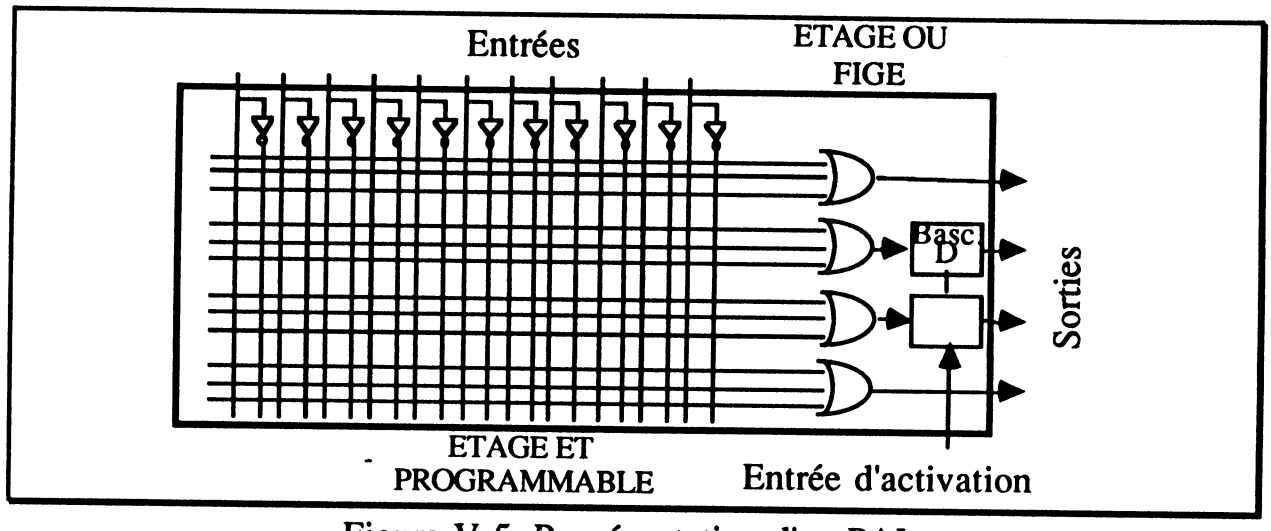

Figure V-5: Représentation d'un PAL

Les résultats obtenus pour la réalisation sur des circuits de ce type pour la bibliothèque MMI sont présentés dans le tableau suivant. Le gain moyen en nombre de boîtiers utilisés est de 20,8% et en niveaux de 6,3%. Le gain ici est du même ordre que dans le cas des Xilinx.

Exemple	Caractéristiques				Aléatoire		ASYL	
	I	O	S	P	B	N	B	N
bbsse	7	7	16	56	2	1	2	1
cse	7	7	16	91	3	2	2	1
dk16	4	3	27	108	3	2	1	2
donfile	2	1	24	96	2	2	1	1
esd	14	7	21	154	8	2	7	2
jay	4	33	58	181	6	2	5	2
kirkman	12	6	17	370	3	2	3	2
opus	5	6	11	22	1	1	1	1
planet	7	19	48	115	5	2	5	2
s1	8	6	20	107	4	2	3	2
s1a	8	6	20	107	3	2	2	2
sand	11	9	32	184	8	2	7	2
sse	7	7	16	56	2	1	2	1
styr	9	10	30	166	6	2	4	2
tbk	6	3	32	1569	6	2	3	2

Tableau 10: Comparaison concernant la cible PAL

5.3.3. Circuits de type PLA (comparaison avec NOVA)

La dernière cible concerne les PLAs (réalisation deux-couches). Nous avons voulu évaluer notre logiciel sur cette cible, même si celui-ci est plus adapté à une cible multicouche. Des comparaisons ont été effectués avec un logiciel de codage spécialisé pour la cible deux-couches, NOVA. Celui-ci est un outil développé à l'université de Berkeley [VIL89ab] qui, contrairement à ASYL, ne recherche pas des situations au sein du graphe d'un automate, mais effectue une minimisation symbolique de la liste des transitions. Les contraintes de codage sont alors déduites de la liste minimisée, et un codage est alors effectué par immersion de ces contraintes dans l'hypercube. Notons que NOVA se restreint à la cible PLA car la minimisation symbolique n'a donné des résultats probants à l'heure actuelle que sur ce type de circuits. Les résultats de cette comparaison sont donnés dans le tableau suivant. On note que NOVA obtient un gain de 6,3% par rapport à ASYL. Seule la stratégie L1 a été utilisée dans ASYL car, pour la cible deux-couches, on ne s'intéresse pas aux factorisations possibles. Ces résultats sont fortement encourageants car avec la minimisation symbolique on obtient exactement les groupes d'adjacence qu'il faut immerger, et ASYL s'en sort sans effectuer cette minimisation symbolique.

Exemple	Caractéristiques				NOVA	ASYL
	I	O	S	P	Monômes	Monômes
bbara	4	2	10	60	24	26
bbsse	7	7	16	56	29	33
bbtas	2	2	6	24	9	9
cse	7	7	16	91	45	47
dk14	8	5	7	56	25	27
dk16	4	3	27	108	57	59
dk17	4	3	8	32	19	19
dk27	2	2	7	14	8	8
donfile	2	1	24	96	28	38
ex1	9	19	20	138	37	44
ex4	6	9	14	21	18	19
ex6	5	8	8	34	25	27
keyb	7	2	19	170	48	49
kirkman	12	6	17	370	60	62
lion9	2	1	9	25	9	10
mark1	5	16	15	22	17	20
modulo12	1	1	12	24	11	11
opus	5	6	11	22	16	15
planet	7	19	48	115	86	94
s1	8	6	20	107	63	74
s1a	8	6	20	107	69	61
s8	4	1	5	20	8	10
sand	11	9	32	184	89	105
scf	27	56	121	166	137	137
sse	7	7	16	56	30	33
styr	9	10	30	166	94	89
tav	4	4	4	49	11	11
tbk	6	3	32	1569	57	58
train11	2	1	11	25	9	11

Tableau 11: Comparaison concernant la cible PLA entre ASYL et NOVA

5.4. CONCLUSION

L'évaluation des différentes stratégies de codage d'ASYL par rapport à une moyenne de codages aléatoires a démontré clairement que les meilleures optimisations étaient apportées par la considération de situations à fusion de monômes. Un apport additionnel des situations à factorisation n'est quand même pas à négliger. Des comparaisons faites avec un des meilleurs logiciels connus, JEDI, montrent la supériorité du codage d'ASYL sur celui-ci. Forts de ces résultats, nous avons voulu évaluer notre logiciel pour d'autres cibles

multicouches comme les circuits programmables Xilinx, et les PALs de la série MMI. Un gain moyen de l'ordre de 20-25% dans les deux cas nous rassure à cet effet. Néanmoins, pour la cible Xilinx, il serait intéressant de reconsidérer le codage en utilisant plus de bits. En dernier lieu, même si notre outil s'adresse plus à des cibles multicouches, nous avons voulu étudier son comportement pour la cible deux-couches. Des comparaisons effectuées avec NOVA montrent une légère avance de celui-ci sur ASYL. Cette lacune pourrait éventuellement être remédiée par l'utilisation du codage symbolique.

CHAPITRE VI

COMPARAISON AVEC D'AUTRES MÉTHODES

6.1.ETAT DE L'ART

Le problème du codage d'automates est apparemment un vieux problème et a fait l'objet de nombreuses publications depuis les années 50.

Au départ, les préoccupations étaient principalement d'éviter des courses critiques au sein d'automates asynchrones. En effet, les automates asynchrones sont des circuits qui évoluent en fonction des changements des entrées et qui passent par un certain nombre d'états instables avant d'atteindre un état stable. Les courses critiques proviennent de la différence de rapidité d'évolution des variables internes, et peuvent amener l'automate à un état non prévu ([KOH78]). Celles-ci arrivent principalement quand plusieurs variables internes changent en même temps. Pour pallier ce problème, il faut alors s'arranger pour que ces variables ne changent qu'une seule à la fois lors de l'évolution de l'automate. A cet effet, des méthodes de codage ont été proposées qui pouvaient assurer l'absence de telles courses critiques ([SAU68], [LIU63], [SAU72ab], [TRA66], [UNG63]).

D'autres auteurs se sont plutôt intéressés à obtenir des réalisations physiques peu coûteuses. Pour ce faire, Armstrong essaye de rapprocher les codes de certains états pouvant apparaître ensemble dans une même fonction logique. Il se base sur le fait que si plusieurs couples d'états permettent la création d'un unique monôme, alors ces monômes peuvent encore fusionner pour donner de plus gros monômes, et ainsi de suite ([ARM62a]). Puis il généralise l'idée dans [ARM62b] en considérant des paquets d'états courants et des paquets d'entrée. Pour ceux-ci, il calcule les états suivants et définit des règles de codage de ces différents paquets. Stearns et Hartmanis effectuent des codages en considérant des couples de partitions dans le graphe et proposent les théorèmes de dépendance réduites que nous avons vus dans un chapitre précédent ([HAR61ab], [HAR66]). Karp reprend leur théorie et propose une méthode de codage systématique dans [KAR64]. Dolotta dans [DOL64] procède autrement en considérant des "colonnes codables": ce sont des colonnes binaires de base qui quand elles sont juxtaposées constituent un ensemble de codes différents pour les états d'un automate. Un coût en logique utilisée est évaluée pour chacune des colonnes et un nombre nécessaire de colonnes les moins coûteuses sont choisies pour constituer les codes des états de l'automate. L'algorithme de [TOR68] semble être une amélioration de l'algorithme de Karp, où des filtres efficaces sont mis en oeuvre pour limiter encore plus le nombre de colonnes codables.

Plus récemment, avec l'apparition d'outils de minimisation hautement optimisés, de nouveaux algorithmes de codage sont apparus. Ces algorithmes supposent une bonne connaissance de ces outils et tendent tous à préparer le travail de ces minimiseurs. Les logiciels actuels les plus connus vont être décrits dans les paragraphes suivants. Quelques notions préliminaires seront avant tout introduites pour permettre une meilleure compréhension des algorithmes mis en oeuvre.

6.2. NOTIONS PRÉLIMINAIRES

6.2.1. Minimisation symbolique

La minimisation symbolique consiste à trouver une représentation de coût minimal d'un circuit indépendamment du codage binaire des entrées ou des sorties. D'une manière générale, dans des descriptions de haut niveau, des variables ne sont pas représentées sous forme binaire mais sous forme de symboles. La description d'un automate en est un cas typique: les variables "état courant" et "état suivant" prennent un certain nombre de valeurs symboliques qui sont les états de l'automate. On comprend alors aisément que le codage de ces variables symboliques influence fortement la complexité du circuit final. Cette méthode de codage repose sur deux étapes principales: 1°) la recherche d'une représentation minimale de la fonction qui soit indépendante du codage des entrées et des sorties, 2°) un codage de ces entrées et sorties compatible avec la représentation minimale trouvée.

De Micheli [DEM85a] énonce quatre types de codage symbolique: le codage des entrées uniquement, le codage des sorties uniquement, le codage d'entrées et de sorties, et le codage d'entrées et de sorties de telle sorte que le codage des entrées soit le même que celui des sorties (le codage des états est du dernier type). Il annonce ensuite un algorithme permettant de résoudre la première étape des quatre types de codage précédents. Il subsiste alors la deuxième étape qui consiste à satisfaire un certain nombre de contraintes sur les codes à trouver pour les symboles. Pour les symboles d'entrée, les contraintes sont d'affecter des sous-ensembles de symboles à des sous-cubes de l'hypercube des codes de ces symboles. Pour les symboles de sortie, une relation d'ordre est générée et les codes des symboles doivent être tels que la relation d'ordre induite sur ces codes est respectée.

6.2.2. Techniques de codage

Dans la phase d'immersion, il s'agit d'affecter des ensembles de symboles à des sous-cube de l'hypercube. Deux techniques peuvent être utilisées pour ce faire. La première consiste à considérer des codes de taille fixée à l'avance et de placer les symboles sur des sommets de l'hypercube en respectant les relations cubiques. Elle est connue sous le nom de codage par ligne ("row-encoding"). Une deuxième technique, le codage par colonne ("column-encoding"), consiste à constituer le code des symboles bit par bit. En fait, le bit ajouté, quand cela est nécessaire, permet de séparer les états d'un même groupe d'adjacence de ceux qui n'y sont pas. On parle alors de bissection de l'hypercube.

exemple

Considérons les groupes d'adjacence suivants {s1, s2, s3, s4}, {s2, s3, s5} et {s1, s2, s6}. L'ensemble des états est de cardinal 7.

Procédons au codage par ligne; la dimension est d'ores et déjà fixée à 3.

{s1, s2, s3, s4} est affecté au cube des codes {000, 001, 010, 011}. Puis {s2, s3, s5} est affecté au cube des codes {000, 001, 100, 101}. Finalement {s1, s2, s6} est affecté au cube des codes {000, 010, 100, 110}. Ceci a pour conséquence de définir pour les états {s1, s2, s3, s4, s5, s6, s7} les codes 010, 000, 001, 011, 101, 110 et 111 respectivement.

Procédons maintenant au codage par colonne.

Une première variable sépare {s1, s2, s3, s4} de {s5, s6, s7}. On choisit Y1 de telle sorte que Y1({s1, s2, s3, s4}) = 0 et Y1({s5, s6, s7}) =1. On déduit directement que le monôme associé au premier groupe est $\overline{Y1}$.

Une deuxième variable sépare ensuite {s2, s3, s5} de {s1, s4, s6, s7}. On obtient alors Y1,Y2({s2, s3}) = 00, Y1,Y2({s1, s4}) = 01, et Y1,Y2({s5}) = 10 et Y1,Y2({s6, s7})= 11. Le monôme associé au deuxième groupe est alors $\overline{Y2}$.

Une troisième variable sépare {s1, s2, s6} de {s3, s4, s5, s7}. On obtient finalement Y1,Y2,Y3({s2})=000, Y1,Y2,Y3({s3})=001, Y1,Y2,Y3({s1})=010, Y1,Y2({s4})=011, Y1,Y2,Y3({s5})=101, Y1,Y2,Y3({s6})=110 et Y1,Y2,Y3({s7})=111. Le monôme associé au troisième groupe est $\overline{Y3}$.

6.3. PRINCIPAUX OUTILS DE CODAGE

6.3.1. KISS

KISS est un logiciel développé par de De Micheli à l'université de Californie ([DEM84a]) et s'adresse à la cible deux-couches.

Dans une première approche [DEM83], l'auteur n'utilise pas de minimisation symbolique, mais analyse chaque couple de transitions de l'automate et en déduit des contraintes. Selon certaines conditions sur les entrées et les sorties, il déduit des contraintes sur les codes des états courants et suivants de ces couples de transition. Ces contraintes peuvent être des contraintes d'adjacence, de couverture (inclusion binaire) ou de distance minimale à respecter entre codes. Deux cas se présentent: soit les deux transitions peuvent être fusionnées permettant le gain d'un monôme, soit des variables peuvent être enlevées du monôme d'une des deux transitions. La partie de l'immersion consiste alors à respecter ces contraintes.

Plus tard, l'auteur crée KISS où la minimisation symbolique est effectuée avant la phase d'immersion.

6.3.1.1. Minimisation symbolique de KISS

L'auteur, même s'il est conscient que le codage d'états est un problème de codage d'entrées et de sorties symboliques, ne s'attaque qu'au problème du codage de ces états en tant que symboles d'entrée uniquement. Ceci peut se faire facilement en effectuant un codage dit "1 parmi N" des états. On remarque alors que les codes constitués de plus d'un "1" de même que le code tout à "0" n'interviennent pas dans l'automate. Ces codes peuvent alors être utilisés comme des entrées pour lesquelles les fonctions ont une valeur indéterminée (on parle alors de couverture à phi [KUN68] d'une fonction incomplètement spécifiée). Après minimisation, un système minimal en termes de monômes est obtenu, définissant ainsi les groupes d'adjacence pour les états. Les cardinaux de certains groupes d'adjacence peuvent encore être réduits car dans certains cas, certains états sont redondants au vu d'autres groupes d'adjacence.

L'auteur démontre que, si toutes les contraintes sont satisfaites, alors on ne peut pas trouver plus de monômes que dans le système obtenu. Par contre, après immersion et obtention des codes, la minimisation peut donner de meilleurs résultats, car les contraintes sur les sorties n'ont pas été considérées.

6.3.1.2. Immersion

KISS utilise un codage par colonne pour sa phase d'immersion. Néanmoins, au lieu de travailler directement sur les groupes d'adjacence, une matrice de contraintes est définie. Un certain nombre de règles de simplification sont alors appliquées sur cette matrice pour en diminuer le nombre de contraintes et ainsi accélérer l'immersion. Par exemple, une règle détecte de lignes identiques, et une autre détecte les lignes inverses. Dans les deux cas, une des lignes est supprimée. Après la phase de simplification, la matrice réduite est utilisée pour effectuer l'immersion. Chaque contrainte est considérée à son tour, et il est vérifié s'il est nécessaire de rajouter une variable, ou certains codes peuvent être déterminés pour des états évitant l'utilisation d'une nouvelle variable. KISS procède ainsi jusqu'à satisfaction de toutes les contraintes.

Plus tard, l'auteur comble les lacunes de KISS après avoir mis au point le minimiseur symbolique CAPPUCCINO. Ce logiciel considère à la fois les variables symboliques d'entrée et de sortie, et gère les deux types de contraintes obtenu pour coder les symboles associés.

6.3.2. JEDI

JEDI [LIN89] est un outil de codage symbolique développé par B. Lin à l'université de Californie à Berkeley qui vise la cible multicouche. Il peut ainsi s'attaquer au problème du codage d'automates où les symboles considérés sont les états d'un automate. Une première phase consiste en un calcul d'attractions entre symboles d'une même variable symbolique et une deuxième phase affecte des codes proches à des symboles ayant une forte attraction entre eux.

6.3.2.1. Calcul d'attractions

Deux stratégies similaires sont utilisées pour le codage des variables symboliques d'entrée et des variables symboliques de sortie.

Considérons une variable symbolique de sortie VS et notons $\{vs_1, \ldots, vs_n\}$ l'ensemble de ses valeurs possibles. Dans un premier temps, JEDI recherche l'ensemble I_j des monômes pour lesquels VS vaut vs_j, ceci pour chaque valeur vs_j. Ensuite, l'attraction m_{jk} entre deux symboles vs_j et vs_k est calculé selon la formule $\sum_{b=1}^{|I_j|} \sum_{a=1}^{|I_k|} P(i_{ja}, i_{kb})$, où $P(i_{ja}, i_{kb})$ est la proximité entre les monômes

i_{ja} et i_{kb}. Cette formule évalue en fait le nombre de littéraux communs à tout couple de monômes de I_j et I_k, et essaye de prédire les factorisations qui peuvent avoir lieu quand les deux symboles vs_j et vs_k ont des codes proches.

Le calcul des attractions dans le cas des entrées se fait de manière similaire. Pour une variable d'entrée VE admettant l'ensemble des valeurs $\{ve_1, ..., ve_n\}$, JEDI calcule l'ensemble O_j des vecteurs de sortie pour lesquels VE vaut ve_j. Les attractions sont alors calculés en utilisant une formule identique en remplaçant les monômes par les vecteurs de sortie. Ici, cette formule évalue le nombre de fois que les sorties utilisent un couple de symboles d'entrée, et prédit la création d'expressions communes entre ces sorties.

6.3.2.2. Affection des codes

Pour chacune des variables symboliques, JEDI considère la matrice des attractions m_{jk} pour tout couple de valeurs de cette variable. Le but est d'alors de trouver un codage des symboles qui minimise la somme $\sum_{j=1}^{n} \sum_{k=i+1}^{n} m_{jk} * H(c_j, c_k)$, où c_j représente le code du $j^{ième}$ symbole. Minimiser cette somme revient à donner des codes très proches aux symboles ayant une forte attraction entre eux. Ensuite, une heuristique est utilisée pour minimiser cette somme. A tout moment, le symbole non codé ayant la plus forte attraction pour les symboles codés est pris en compte. Le code inutilisé le plus proche de ceux des symboles codés lui est affecté. Ceci est répété jusqu'à ce que tous les symboles soient codés.

JEDI considère le codage d'automates comme un cas particulier où les états sont les symboles à coder. Néanmoins, il n'est pas spécifié comment JEDI traite ce cas particulier, car ici les états sont à la fois symboles d'entrée et symboles de sortie.

6.3.3. MUSTANG

MUSTANG est un autre outil de l'université de Californie développé par S. Devadas ([DEV87], [DEV88]), qui s'intéresse aussi à la cible multicouche. Il procède d'une manière analogue à JEDI, à ceci près qu'il ne s'intéresse pas au problème général du codage symbolique, mais se restreint au codage des automates. L'algorithme procède en deux étapes: un calcul d'attractions entre états et un codage par proximité identique à celui de JEDI.

Le calcul d'attractions se fait au vu du graphe du contrôleur. Deux stratégies sont adoptées pour MUSTANG: une stratégie dite "fanin oriented" (guidée par les entrées) et une autre dite "fanout oriented" (guidée par les sorties).

6.3.3.1. Calcul d'attractions

Stratégie fanin

Dans la stratégie fanin, MUSTANG considère les états successeurs produits par des entrées similaires et des états présents similaires. Tout couple d'états admettant beaucoup d'état prédécesseurs communs et produits pour plusieurs entrées similaires auront une très forte attraction. Par cette stratégie, MUSTANG espère provoquer la création d'un grand nombre de monômes communs entre plusieurs fonctions.

L'attraction entre deux états s_i et s_j est alors composée d'une première partie égale à $N/2*\sum_{s_k} pred(s_k, s_i)*pred(s_k, s_j)$, où N est le nombre de bits de codage et $pred(s_x, s_y)$ dénote le nombre de fois où l'état s_x apparaît comme prédécesseur de s_y. Pour la deuxième partie de l'attraction, MUSTANG considère chaque entrée i et son complément, et calcule pour chaque état le nombre de fois où cet état est produit par une transition comportant cette entrée. Cette occurrence est notée $succ(i, s_x)$ pour l'entrée i et l'état s_x. L'attraction est alors complétée par le terme $\sum_i succ(i, s_k)*succ(i, s_l) + succ(\bar{i}, s_k)*succ(\bar{i}, s_l)$.

Stratégie fanout

Dans cette stratégie, MUSTANG considère les états qui produisent des sorties similaires et qui admettent des état suivants similaires. Les états ayant beaucoup d'états suivants communs et émettant beaucoup de commandes identiques seront donnés de fortes attractions. Le calcul d'attraction pour tout couple d'états se fait de manière analogue au cas précédent en considérant maintenant les états en tant que successeurs et les sorties. Ici, MUSTANG recherche à maximiser la taille des monômes communs entre différentes fonctions.

6.3.3.2. Affectation des codes

La technique d'immersion utilisée par MUSTANG est connue sous le terme anglais de "wedge clustering" (regroupement par paquets). Dans cette technique, à chaque étape, un état s et un ensemble de N états associés $\{s_1, ..., s_N\}$ tels que la somme $\sum_{k=1}^{N} att(s, s_k)$ soit maximale sont sélectionnés. Un ensemble de N+1 codes inutilisés est choisi pour ces états de manière à minimiser la distance existant entre eux (notons que certains de ces états peuvent déjà avoir été codés). Une fois cela fait, l'état s est supprimé de l'ensemble des états considérés et le procédé est recommencé avec les états restants jusqu'à ce que tous les états aient été codés.

6.3.4. NOVA

NOVA a été développé par T. Villa à l'université de Californie ([VIL89a], [VIL89b]). Cet outil s'intéresse principalement à la cible deux-couches, quoique des évaluations sur la cible multicouche aient été effectuées et donné des résultats satisfaisants pour l'auteur.

6.3.4.1. Constitution des groupes d'adjacence

Grâce à la cible deux-couches, NOVA peut utiliser de manière efficace l'algorithme de minimisation symbolique de De Micheli [DEM85a] par le biais du meilleur minimiseur deux-couches existant à l'heure actuelle, c'est-à-dire Espresso ([RUD85], [BRA85]). Un ensemble de groupes d'adjacence est alors obtenu après cette étape de minimisation de même que la relation d'ordre à respecter sur les codes des états. Ces deux types de contraintes sont obtenus car le codage d'états d'un automate correspond au codage d'entrées et de sorties symboliques.

6.3.4.2. Immersion

NOVA effectue un codage par ligne consistant à affecter ces groupes d'adjacence aux sous-cubes de l'hypercube. La méthode consiste à construire la fermeture par intersection de ces groupes d'adjacence d'une part et à construire le treillis de tous les sous-cubes de l'hypercube d'autre part. Notons que ce treillis est différent du treillis booléen lui-même. L'immersion consiste alors à trouver un isomorphisme entre l'ensemble des groupes d'adjacence de la

fermeture et une partie du treillis des cubes. Cela revient en fait à associer à chaque groupe d'adjacence un cube, de telle sorte que les relations d'intersection et d'inclusion entre groupes d'adjacence soient respectées pour les cubes. En cela, la méthode de NOVA ne diffère pas d'ASYL, à la différence près que la représentation des cubes n'est pas la même, et que NOVA génère exhaustivement tous les cubes et les teste jusqu'à obtenir le bon. ASYL procède différemment en construisant le cube au vu des intersections voulues.

Selon l'option choisie, NOVA s'attaque aussi à la résolution des contraintes d'ordre imposées sur le code des états par la minimisation symbolique. Néanmoins, une plus grande priorité est donnée aux groupes d'adjacence. Pour prendre en compte les contraintes d'ordre, NOVA modifie sa génération de cubes pour un groupe d'adjacence en vérifiant de plus si le cube choisi ne viole pas ces contraintes.

6.3.5. Autres logiciels

La plupart des autres logiciels présentés dans la littérature ne sont en fait que l'utilisation de KISS avec ([COP86]) ou sans modifications ([TSE86]).

CASTOR [RIE90] est un logiciel développé à l'université de Karslruhe et intégré dans le système CADDY [ROS88]. CASTOR ressemble beaucoup à KISS, sauf pour la manière de traiter les contraintes de codage. En effet, CADDY ne considère pas les contraintes une par une, mais sélectionne un ensemble de contraintes deux à deux disjointes. Les contraintes sont codées sur un nombre suffisant de bits, et le processus est recommencé avec un autre ensemble de contraintes. CASTOR ne s'adresse qu'aux PLAs pour le moment, des améliorations étant en cours pour pouvoir prendre en compte la cible multicouche.

Une tentative intéressante a été faite dans [WOL88] pour réaliser un autre type de codage guidé par la création de noyaux. Les auteurs utilisent le modèle d'un automate où les codes des états peuvent être programmés de l'extérieur. Ceci impose alors des circuits internes de reconnaissance des états avec des valeurs de codes préprogrammés. L'idée est alors de choisir de bonnes valeurs de codes de manière à créer des noyaux intéressants. Malheureusement, la méthode n'a pas été probante, les auteurs ayant constaté que des codages aléatoires donnaient de meilleurs résultats que leur logiciel.

6.3.6. Conclusion

Nous avons vu que tous les algorithmes de codage reposent principalement sur deux étapes: une recherche de situations dans le graphe de l'automate et une immersion des états dans l'hypercube.

La recherche de situations se fait par analyse du graphe, ou par minimisation symbolique. Selon le cas, des contraintes d'adjacence sont définies sur les états, ou alors ce sont des attractions qui sont évaluées pour tous les couples d'états. D'autres contraintes comme la définition d'une relation d'ordre entre les codes des états peuvent être obtenues.

La phase d'immersion se fait alors par deux types de codage, soit par ligne ou par colonne. L'immersion des groupes d'adjacence peut se faire par l'un des deux types quelconques. Les problèmes d'attraction ou de couverture (relation d'ordre) se font plus aisément par codage par ligne, puisque tous les bits sont nécessaires pour effectuer les évaluations.

Le tableau suivant résume les méthodes utilisées par les différents algorithmes.

	immersion cubique	codage par proximité	codage par colonne
minimisation symbolique	NOVA	JEDI	KISS CAPPUCCINO
calcul d'attractions		MUSTANG ASYL (L2)	
recherche de situations	ASYL (L1)		

CONCLUSION GENERALE

Nous avons vu que, d'une manière générale, tous les algorithmes de codage procèdent en deux étapes: une première étape détecte des situations pouvant amener des simplifications et une deuxième étape qui consiste à satisfaire les contraintes imposées par la première étape. Nous avons montré la recherche de ces situations en nous basant sur les travaux de Stearns et de Hartmanis. En ce qui concerne la deuxième étape, nous avons proposé une nouvelle théorie dite de cubes intersectants permettant de gérer au mieux les contraintes obtenues.

Le premier chapitre nous a permis de poser les bases du problème. Les définitions de base ont été données, et la notion de situation prédictive de minimisation a été introduite. A cet effet, les différentes situations amenant des simplifications ont été illustrées.

Dans le deuxième chapitre, nous avons présenté des théorèmes fondamentaux de la théorie de Stearns et Hartmanis, et montré leur application à la recherche de situations dites à fusion de monômes. Une généralisation nous a ensuite permis de définir les situations à factorisation, en complément des situations à fusion de monômes. Des notions de coûts ont été introduites pour définir des priorités sur ces situations.

Avant de pouvoir décrire l'algorithme de codage, la théorie des cubes intersectants a été développé au troisième chapitre permettant ainsi d'avoir des outils de base efficaces pour la phase d'immersion cubique. Plusieurs conditions nécessaires et/ou suffisantes ont été énoncées pour la recherche de cubes vérifiant des relations d'intersection, d'inclusion et de disjonction avec d'autres cubes.

Dans le quatrième chapitre, l'extension de groupes d'adjacence est abordée. En effet, pour certains automates, tout l'espace des codes n'est pas utilisé. Cela permet la création d'états fictifs, appelés états fantômes, qu'on peut rajouter à l'automate sans changer son comportement. Ces états fantômes permettront d'étendre tout groupe d'adjacence de telle sorte qu'il puisse être associé bijectivement à un sous-cube de l'hypercube. L'algorithme d'immersion cubique est présenté, de même qu'un algorithme "glouton" qui fige le code des états au vu des attractions entre états définies par les situations à factorisation.

Le cinquième chapitre concerne l'évaluation de l'algorithme de codage. Une première étape consiste à faire des évaluations par rapport à un codage aléatoire. Ceci permet d'analyser l'efficacité des différentes stratégies utilisées. Une

deuxième étape s'occupe de la comparaison d'ASYL avec JEDI, un des meilleurs logiciels de codage connus. La supériorité des résultats d'ASYL est démontrée, confirmant l'idée que les situations à fusion de monômes sont celles qui permettent de réduire grandement la surface dûe aux connexions entre cellules de base. Enfin dans une dernière étape, l'algorithme est évalué sur d'autres cibles. Dans ce cas aussi, les résultats sont probants.

Dans le sixième chapitre, les techniques de codage les plus connues sont présentées après un bref rappel des travaux antérieurs dans ce domaine. La description de chaque technique donne une idée des différentes façons de mettre en oeuvre un algorithme de codage, qui, néanmoins, restent toutes basées sur les deux étapes mentionnées plus haut.

Il est surprenant de noter que le codage d'automates qui a été abordé dans les années 50 est encore un sujet d'actualité. En effet, avec la complexité croissante des circuits synthétisés de nos jours et l'apparition sans cesse renouvelée de nouveaux circuits, les techniques de minimisation, et de ce fait, les outils pouvant prévoir ces minimisations, doivent être adaptées. Le sujet n'est donc pas clos même si les logiciels actuels atteignent un degré d'optimalité très élevé.

BIBLIOGRAPHIE

[ARM62a] D.B. Armstrong
"A programmed algorithm for assigning internal codes to sequential machines"
IRE Transactions on Electronic Computers, pp 466-472
Août 1962.

[ARM62b] D.B. Armstrong
"On the Efficient Assignment of Internal Codes to Sequential Machines"
IRE Transactions on Electronic Computers, pp 466-472
Octobre 1962.

[BRA82] R.K. Brayton and C.T. McMullen
"The Decomposition and Factorization of Boolean Functions"
ISCAS Proceedings
Avril 82.

[BRA85] R.K. Brayton, G. D. Hachtel, C. T. McMullen, A. Sangiovanni-Vincentelli
"Logic Minimization Algorithms for VLSI Synthesis"
Kluwer Academic Publishers
1985.

[BRO81] Douglas W. Brown
"A State Machine Synthesizer - SMS"
18th Design Automation Conference, Nashville
Juin 1981

[COP86] A.J. Coppola
"An Implementation of a State Assignment Heuristic"
23rd Design Automation Conference,
Juillet 1986

[CRA88] M. Crastes de Paulet, C. Duff, P. Sicard, G. Saucier, F. Poirot
"Controller synthesis in the ASYL system"
IFIP working conference on design methodologies for VLSI and computer architecture, Pise, Italie
Septembre 1988 (Paru aussi en Proceedings de l'édition North Holland)

[CRA89] M. Crastes de Paulet, C. Duff, R. Leveugle, F. Poirot, G. Saucier, P. Sicard
"ASYL : a logic and architecture design automation system"
Colloque EURO-ASIC 89, pp. 183-209, Grenoble, France
Janvier 1989

[CUR62] H. Allen Curtis
"Multiple Reduction of Variable Dependency of Sequential Machines"
Journal of the ACM, vol. 9, No 3, pp. 324-344
Juillet 1962

[DEM83] G. de Micheli, A. Sangiovanni-Vincentelli
"Computer-Aided Synthesis of PLA-Based Finite State Machines"
ICCAD 83, Santa Clara, pp. 154-156
Septembre 1983

[DEM84a] G. de Micheli, A. Sangiovanni-Vincentelli and R.K. Brayton
"KISS: a Program for Optimal State Assignment of Finite State Machines"
ICCAD 84, Santa Clara
Novembre 1984

[DEM84b] G. de Micheli
"Optimal Encoding of Control Logic"
Int. Conference on Circuit and Computer Design
Septembre 1984

[DEM85a] G. de Micheli
"Symbolic Minimization of Logic Functions"
ICCAD 85, pp 293-295
Novembre 1985.

[DEM85b] G. de Micheli, R.K. Brayton, A. Sangiovanni-Vincentelli
"Optimal State Assignment of Finite State Machines"
IEEE Transactions on CAD, pp 269-284
Juillet 1985

[DEM86] G. de Micheli
"Symbolic Design of Combinational and Sequential Logic Circuits Implemented by Two-level Macros"
IEEE Transactions on CAD, pp 597-616
Octobre 1986

[DEV87] S. Devadas, H. T. Ma, R. Newton, A. Sangiovanni-Vincentelli
"MUSTANG: State Assignment of Finite State Machines for Optimal Multi-Level Logic Implementations"
ICCAD 87, pp 16-19

[DEV89] S. Devadas
"General Decomposition of Sequential Machines: Relationships to State Assignment"
26th Design Automation Conference, Las Vegas, pp 314-320
Juin 1989

[DOL64] T. A. Dolotta, E. J. McCluskey
"The Coding of Internal States of Sequential Circuits"
IEEE Transactions on Electronic Computers, vol. EC-13, pp 549-562
Octobre 1964

[DUF88] C. Duff, G. Saucier, P. Sicard, F. Poirot
"Des contrôleurs sur PAL, PLA, EPLD et logique aléatoire"
Col. Nat. Conception de Circuits à la Demande, Grenoble, France
Janvier 1988

[HAR61a] J. Hartmanis
"On the State Assignment Problem for Sequential Machines. I"
IRE Transactions on Electronic Computers, vol. EC-10, pp 157-165
Juin 1961

[HAR61b] J. Hartmanis, R. Stearns
"On the State Assignment Problem for Sequential Machines. II"
IRE Transactions on Electronic Computers, vol. EC-10, pp 593-603,
Decembre 1961

[HAR66] J. Hartmanis, R.E. Stearns
"Algebraic structure, Theory of Sequential Machines"
Prentice Hall, 1966

[HUF54] D. A. Huffman
"The Synthesis of Iterative Switching Circuits"
Res Lab of Electronics , MIT, Quarterly Progress Report

[HUF54] D. A. Huffman
"The Synthesis of Sequential Switching Circuits"
J. Franklin Inst., vol. 257, pp. 169-190, pp. 275-303
Mars et Avril 1959

[KAI88] K. Kaiser, R. Luber, H.M. Lipp
"Extended Symbolic minimization for pla-based FSM"
Institute for technical information processing
Uni. of Karlsruhe.
1988

[KAR64] R. Karp
"Some Techniques for State Assignment for Synchronous Sequential Machines"
IEEE Transactions on Electronic Computers, vol. EC-13, pp 507-518
Octobre 1964

[KOH78] Z.Kohavi
Switching and Finite Automata Theory
McGraw-Hill Book Company, USA
1978

[KUN68] J. Kuntzmann
Algèbre de Boole
Edition Dunod, Paris
1969

[LIN89] B. Lin and A.R. Newton
"Synthesis of Multiple-Level Logic from Symbolic High Level Description Languages"
VLSI'89 Conference, Munich, pp 187-196
Août 1989

[LIU63] C.N. Liu
"A State Variable Assignment Method for Asynchronous Sequential Switching Circuits"
ACM
Avril 63

[MCC59] E. J. McCluskey, S. H. Unger
"A Note on the Number of Internal Variable Assignments for Sequential Switching Circuits"
IRE Transactions on Electronic Computers, vol. EC-8, pp 439-440
Decembre 1959

[MCNC87] MCNC Set of Benchmarks
Septembre 1987

[MEY84] M. J. Meyer, P. Agarwal
"A VLSI FSM Design System"
21st Design Automation Conference
Juin 1984

[MOO56] E. F. Moore
"Gedanken-Experiments on Sequential Circuits" in "Automata Studies"
Princeton University Press, Princeton, N.Jersey
1956

[MEA55] G. H. Mealy
"A Method for Synthesizing Sequential Circuits"
Bell Sys. Tech. Journal, vol. 34, pp. 1045-1079
Septembre 1955

[NIC65] A. Nichols, A. Bernstein
"State Assignment in Combinational Networks"
IEEE Transactions on Electronic Computers, vol EC-14, pp. 343-349
Juin 1965

[POI88] F. Poirot et al.
"Controller Synthesis in the ASYL System"
Proc. of the Intern. Workshop on Logic and Architecture Synthesis for Silicon Compilers, Grenoble FRANCE
Mai 1988
(Paru aussi en Proceedings de l'édition Elsevier Science Publishers (North Holland)

[RIE90] G. Rietsche, M. Nieher
"CASTOR: State Assignment in a Finite State Machine Synthesis System"
EUROASIC'90, Paris, France
Mai-Juin 1990

[ROS88] W. Rosenstiel et al.
"Datapath and Control Synthesis in the CADDY System"
International Workshop on logic and architecture synthesis for silicon compilers, Grenoble, France
Mai 88

[RUD85] R. Rudell, A. Sangiovanni-Vincentelli
"ESPRESSO-MV: Algorithms for Multivalued Logic Minimization"
Proc. Custom Integrated Circuit Conference, Portland, Oregon
Mai 85

[SAU68] G. Saucier
"Encoding of Asynchronous Sequential Networks"
IEEE Transactions on Electronic Computers vol 16 pp 365-369

[SAU70] G. Saucier
"Codage des Automates Asynchrones"
Thèse de Doctorat, Institut de Mathématiques Appliquées de Grenoble, Université de Grenoble
Novembre 1970

[SAU72a] G. Saucier
"State Assignment of Asynchronous Sequential Machines using Graph Techniques"
IEEE Transactions on Computers, vol. C-21, No. 3, pp. 282-288
Mars 1972.

[SAU72b] G. Saucier
"Next State Equations of Asynchronous Sequential Machines"
IEEE Transactions on Computers
Novembre 1972.

[SAU87] G. Saucier, M. Crastes, P. Sicard
"ASYL: A Rule-Based System for Controller Synthesis"
IEEE Transactions on CAD, pp. 1088-1097
Novembre 1987

[SAU89a] G. Saucier, C. Duff, F. Poirot
"A new embedding method for state assignment"
International Workshop on Logic Synthesis
Microelectronics Center of North Carolina, USA, Mai 1989

[SAU89b] G. Saucier, C. Duff, F. Poirot
"State Assignment Using a New Embedding Method Based on an Intersecting Cube Theory"
26th Design Automation Conference, Las Vegas, pp 321-326
Juin 1989

[SAU89c] G. Saucier, C. Duff
"Cube-Sharing in the Hypercube"
1er Colloque Européen sur les Hypercubes et Calculateurs Distribués, Rennes, Octobre 89

[SAU90] G. Saucier, C. Duff, F. Poirot
"State Assignment of Controllers for Optimal Area Implementation"
The European Design Automation Conference, Glasgow, Scotland, pp. 547-551
Mars 90

[SAU90] G. Saucier, C. Duff
"Controller State Assignment for Optimal Global Area Implementation"
Synthesis and Simulation Meeting and International Interchange, Japan, pp. 95-102
Octobre 90

[SIC88a] P. Sicard, C. Duff,
"An alternative to Espresso II"
Workshop on logic and architecture synthesis for silicon compilers INPG, Grenoble
Mai 88
(Paru aussi en Proceedings de l'édition Elsevier Science Publishers (North Holland)

[SIC88a] P. Sicard
"Nouvelles méthodes de synthèse logique"
Thèse de Doctorat, Institut National Polytechnique de Grenoble,
Septembre 88

[SPA84] L. Spaanenburg, G. J. Kleissen, H. Van der Veen
"Direct Implementation of State Diagram Specifications"
Proc. Custom Integrated Circuits Conference, New York
Mai 1984

[STO72] H. Story, H. J. Harrison, E. A. Reinhard
"Optimum State Assignment for Synchronous Sequential Circuits"
IEEE Transactions on Computers, vol. C-21, No. 12, pp. 1365-1373
Decembre 1972.

[TOR68] H. C. Torng
"An Algorithm for Finding Secondary Assignments of Synchronous Sequential Circuits"
IEEE Transactions on Computers, vol. C-17, No. 5, pp. 461-469
Mai 1968

[TRA66] J. H. Tracey
"Internal State Assignments for Asynchronous Sequential Machines"
IEEE Transactions on Electronic Computers, vol. EC-15, pp. 551-560
Août 1966

[TSE86] C. Tseng, A. Prabhu, C. Li, Z. Mehmood, M. Tong
"A Versatile Finite State Machine Synthesizer"
ICCAD 86, Santa Clara, California
Novembre 1986

[UNG63] S.H. Unger
"A Row Assignment for Delay-Free Realizations of Flow Tables Without Essential Hazards"
IEEE Transactions on Electronic Computers,Vol 17, pp 146-151

[VIL89a] T. Villa, A. Sangiovanni-Vincentelli
"NOVA: State Assignment of Finite State Machines for Optimal Two-Level Logic Implementations"
26th Design Automation Conference, Las Vegas, pp 327-332
Juin 1989

[VIL89b] T. Villa, A. Sangiovanni-Vincentelli
"Algorithms for State Assignment of Finite State Machines for Optimal Two-Level Logic Implementations"
International Workshop on Logic Synthesis
Microelectronics Center of North Carolina, North Carolina
Juin 1989

[WOL88] W. Wolf, K. Keutzer, J. Akella
"A Kernel-Finding State Assignment Algorithm for Multi-Level Logic"
25th Design Automation Conference, Anaheim Convention Center, pp. 433-438
Juin 1988

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Grenoble, le 06 Février 1991

DÉPARTEMENT DES ÉTUDES DOCTORALES

Affaire suivie par
Tél : 76.57.

N/Réf. :

Objet :

AUTORISATION de SOUTENANCE

Vu les dispositions de l'arrêté du 23 Novembre 1988 relatif aux Etudes Doctorales
Vu les rapports de présentation de :

- Monsieur DE MICHELI
- Monsieur COSTES

Monsieur DUFF Christopher

est autorisé(e) à présenter une thèse en soutenance en vue de l'obtention du diplôme de DOCTEUR de l'INSTITUT NATIONAL POLYTECHNIQUE de GRENOBLE, spécialité :

"Microélectronique"

Pour le Président de l'I.N.P.-G.
et par délégation,
le Vice-Président
M. GARNIER