Traitement de données ambigues dans un système de base de données. Application aux bases de données démographiques

Yves Chiaramella

THESE

présentée à

l' Université Scientifique et Médicale de Grenoble

et à

l' Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR D'ETAT ES- SCIENCES
«Mathématiques»

par

Yves CHIARAMELLA

TRAITEMENT DE DONNEES AMBIGUES DANS UN SYSTEME DE BASE DE DONNEES. APPLICATION AUX BASES DE DONNEES DEMOGRAPHIQUES.

Thèse soutenue le 26 Juin 1981 devant la commission d'examen.

L. BOLLIET	Président
J. C. BOUSSARD	Examinateurs
C. DELOBEL	
A. LENTIN	
G. VEILLON	
M. SKOLNICK	Invité

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

Monsieur Gabriel CAU : Président
Monsieur Joseph KLEIN : Vice-Président

MEMBRES DU CORPS ENSEIGNANT DE L'U.S.M.G.

PROFESSEURS TITULAIRES

MM.	AMBLARD Pierre	Clinique de dermatologie
	ARNAUD Paul	Chimie
	ARVIEU Robert	I.S.N.
	AUBERT Guy	Physique
	AYANT Yves	Physique approfondie
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale A
	BEAUDOING André	Clinique de pédiatrie et puériculture
	BELORIZKY Elie	Physique
	BARNARD Alain	Mathématiques pures
Mme	BERTRANDIAS Françoise	Mathématiques pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques pures
	BEZES Henri	Clinique chirurgicale et traumatologie
	BLAMBERT Maurice	Mathématiques pures
	BOLLIET Louis	Informatique (I.U.T. B)
	BONNET Jean-Louis	Clinique ophtalmologie
	BONNET-EYMARD Joseph	Clinique hépato-gastro-entérologie
Mme	BONNIER Marie-Jeanne	Chimie générale
MM.	BOUCHERLE André	Chimie et toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques appliquées
	BOUTET DE MONVEL Louis	Mathématiques pures
	BRAVARD Yves	Géographie
	CABANEL Guy	Clinique rhumatologique et hydrologique
	CALAS François	Anatomie
	CARLIER Georges	Biologie végétale
	CARRAZ Gilbert	Biologie animale et pharmacodynamie

.../...

MM.	CAU Gabriel	Médecine légale et toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques pures
	CHARACHON Robert	Clinique ot-rhino-laryngologique
	CHATEAU Robert	Clinique de neurologie
	CHIBON Pierre	Biologie animale
	COEUR André	Pharmacie chimique et chimie analytique
	COUDERC Pierre	Anatomie pathologique
	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DELORMAS Pierre	Pneumophtisiologie
	DEPORTES Charles	Chimie minérale
	DESRE Pierre	Métallurgie
	DODU Jacques	Mécanique appliquée (I.U.T. I)
	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	FONTAINE Jean-Marc	Mathématiques pures
	GAGNAIRE Didier	Chimie physique
	GALVANI Octave	Mathématiques pures
	GASTINEL Noël	Analyse numérique
	GAVEND Michel	Pharmacologie
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques pures
	GERMAIN Jean-Pierre	Mécanique
	GIRAUD Pierre	Géologie
	JANIN Bernard	Géographie
	KAHANE André	Physique générale
	KLEIN Joseph	Mathématiques pures
	KOSZUL Jean-Louis	Mathématiques pures
	KRAVTCHENKO Julien	Mécanique
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
Mme	LAJZEROWICZ Janine	Physique
MM.	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie pharmaceutique
	LAURENT Pierre	Mathématiques appliquées
	LEDRU Jean	Clinique médicale B
	LE ROY Philippe	Mécanique (I.U.T. I)

.../...

MM.	LLIBOUTRY Louis	Géophysique
	LOISEAUX Jean-Marie	Sciences nucléaires
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LOUP Jean	Géographie
Mlle	LUTZ Elisabeth	Mathématiques pures
MM.	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Clinique cardiologique
	MAYNARD Roger	Physique du solide
	MAZARE Yves	Clinique Médicale A
	MICHEL Robert	Minéralogie et pétrographie
	MICOUD Max	Clinique maladies infectieuses
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	NEGRE Robert	Mécanique
	NOZIERES Philippe	Spectrométrie physique
	OZENDA Paul	Botanique
	PAYAN Jean-Jacques	Mathématiques pures
	PEBAY-PEYROULA Jean-Claude	Physique
	PERRET Jean	Séméiologie médicale (neurologie)
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REVOL Michel	Urologie
	RINALDI Renaud	Physique
	DE ROUGEMONT Jacques	Neuro-Chirurgie
	SARRAZIN Roger	Clinique chirurgicale B
	SEIGNEURIN Raymond	Microbiologie et hygiène
	SENGEL Philippe	Zoologie
	SIBILLE Robert	Construction mécanique (I.U.T. I)
	SOUTIF Michel	Physique générale
	TANCHE Maurice	Physiologie
	VAILLANT François	Zoologie
	VALENTIN Jacques	Physique nucléaire
Mme	VERAIN Alice	Pharmacie galénique
MM.	VERAIN André	Physique biophysique
	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale

.../...

PROFESSEURS ASSOCIES

MM.	CRABBE Pierre	CERMO
	SUNIER Jules	Physique

PROFESSEURS SANS CHAIRE

Mlle	AGNIUS-DELORS Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBROISE-THOMAS Pierre	Parasitologie
	ARMAND Gilbert	Géographie
	BENZAKEN Claude	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
	BILLET Jean	Géographie
	BOUCHET Yves	Anatomie
	BRUGEL Lucien	Energétique (I.U.T. I)
	BUISSON René	Physique (I.U.T. I)
	BUTEL Jean	Orthopédie
	COHEN-ADDAD Jean-Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie médicale
	CONTE René	Physique (I.U.T. I)
	DELOBEL Claude	M.I.A.G.
	DEPASSEL Roger	Mécanique des fluides
	GAUTRON René	Chimie
	GIDON Paul	Géologie et minéralogie
	GLENAT René	Chimie organique
	GROULADE Joseph	Biochimie médicale
	HACQUES Gérard	Calcul numérique
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et médecine préventive
	IDELMAN Simon	Physiologie animale
	JOLY Jean-René	Mathématiques pures
	JULLIEN Pierre	Mathématiques appliquées
Mme	KAHANE Josette	Physique
MM.	KRAKOWIACK Sacha	Mathématiques appliquées
	KUHN Gérard	Physique (I.U.T. I)
	LUU DUC Cuong	Chimie organique - pharmacie
	MICHOULIER Jean	Physique (I.U.T. I)
Mme	MINIER Colette	Physique (I.U.T. I)

.../...

MM.	PELMONT Jean	Biochimie
	PERRIAUX Jean-Jacques	Géologie et minéralogie
	PFISTER Jean-Claude	Physique du solide
Mlle	PIERY Yvette	Physiologie animale
MM.	RAYNAUD Hervé	M.I.A.G.
	REBECQ Jacques	Biologie (CUS)
	REYMOND Jean-Charles	Chirurgie générale
	RICHARD Lucien	Biologie végétale
Mme	RINAUDO Marguerite	Chimie macromoléculaire
MM.	SARROT-REYNAULD Jean	Géologie
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale
MM.	STIEGLITZ Paul	Anesthésiologie
	VIALON Pierre	Géologie
	VAN CUTSEM Bernard	Mathématiques appliquées

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

MM.	ARMAND Yves	Chimie (I.U.T. I)
	BACHELOT Yvan	Endocrinologie
	BARGE Michel	Neuro-chirurgie
	BEGUIN Claude	Chimie organique
Mme	BERIEL Hélène	Pharmacodynamie
MM.	BOST Michel	Pédiatrie
	BOUCHARLAT Jacques	Psychiatrie adultes
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BRODEAU François	Mathématiques (I.U.T. B) (Personne étrangère habilitée à être directeur de thèse)
	BERNARD Pierre	Gynécologie
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHARDON Michel	Géographie
	CHERADAME Hervé	Chimie papetière
	CHIAVERINA Jean	Biologie appliquée (EFP)
	COLIN DE VERDIERE Yves	Mathématiques pures
	CONTAMIN Charles	Chirurgie thoracique et cardio-vasculaire
	CORDONNER Daniel	Néphrologie
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie

.../...

MM.	CYROT Michel	Physique du solide
	DENIS Bernard	Cardiologie
	DOUCE Roland	Physiologie végétale
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	FAURE Gilbert	Urologie
	GAUTIER Robert	Chirurgie générale
	GIDON Maurice	Géologie
	GROS Yves	Physique (I.U.T. I)
	GUIGNIER Michel	Thérapeutique
	GUITTON Jacques	Chimie
	HICTER Pierre	Chimie
	JALBERT Pierre	Histologie
	JUNIEN-LAVILLAVROY Claude	O.R.L.
	KOLODIE Lucien	Hématologie
	LE NOC Pierre	Bactériologie-virologie
	MACHE Régis	Physiologie végétale
	MAGNIN Robert	Hygiène et médecine préventive
	MALLION Jean-Michel	Médecine du travail
	MARECHAL Jean	Mécanique (I.U.T. I)
	MARTIN-BOUYER Michel	Chimie (CUS)
	MASSOT Christian	Médecine interne
	NEMOZ Alain	Thermodynamique
	NOUGARET Marcel	Automatique (I.U.T. I)
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (I.U.T. B) (Personnalité étrangère habilitée à être directeur de thèse)
	PEFFEN René	Métallurgie (I.U.T. I)
	PERRIER Guy	Géophysique-glaciologie
	PHELIP Xavier	Rhumatologie
	RACHALL Michel	Médecine interne
	RACINET Claude	Gynécologie et obstétrique
	RAMBAUD Pierre	Pédiatrie
	RAPHAEL Bernard	Stomatologie
Mme	RENAUDET Jacqueline	Bactériologie (pharmacie)
MM.	ROBERT Jean-Bernard	Chimie-physique
	ROMIER Guy	Mathématiques (I.U.T. B) (Personnalité étrangère habilitée à être directeur de thèse)
	SAKAROVITCH Michel	Mathématiques appliquées

.../...

MM.	SCHAERER René	Cancérologie
Mme	SEIGLE-MURANDI Françoise	Crytogamie
MM.	STOEBNER Pierre	Anatomie pathologie
	STUTZ Pierre	Mécanique
	VROUSOS Constantin	Radiologie

MAITRES DE CONFERENCES ASSOCIES

MM.	DEVINE Roderick	Spectro Physique
	KANEKO Akira	Mathématiques pures
	JOHNSON Thomas	Mathématiques appliquées
	RAY Tuhina	Physique

MAITRE DE CONFERENCES DELEGUE

M.	ROCHAT Jacques	Hygiène et hydrologie (pharmacie)

Fait à Saint Martin d'Hères, novembre 1977

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1979-1980

Président : M. Philippe TRAYNARD

Vice-Présidents : M. Georges LESPINARD
M. René PAUTHENET

PROFESSEURS DES UNIVERSITES

MM.	ANCEAU François	Informatique fondamentale et appliquée
	BENOIT Jean	Radioéléctricité
	BESSON Jean	Chimie Minérale
	BLIMAN Samuel	Electronique
	BLOCH Daniel	Physique du Solide - Cristallographie
	BOIS Philippe	Mécanique
	BONNETAIN Lucien	Génie Chimique
	BONNIER Etienne	Métallurgie
	BOUVARD Maurice	Génie Mécanique
	BRISSONNEAU Pierre	Physique des Matériaux
	BUYLE-BODIN Maurice	Electronique
	CHARTIER Germain	Electronique
	CHERADAME Hervé	Chimie Physique Macromoléculaires
Mme	CHERUY Arlette	Automatique
MM.	CHIAVERINA Jean	Biologie, Biochimie, Agronomie
	COHEN Joseph	Electronique
	COUMES André	Electronique
	DURAND Francis	Métallurgie
	DURAND Jean-Louis	Physique Nucléaire et Corpusculaire
	FELICI Noël	Electrotechnique
	FOULARD Claude	Automatique
	GUYOT Pierre	Métallurgie Physique
	IVANES Marcel	Electrotechnique
	JOUBERT Jean-Claude	Physique du Solide - Cristallographie
	LACOUME Jean-Louis	Géographie - Traitement du Signal
	LANCIA Roland	Electronique - Automatique
	LESIEUR Marcel	Mécanique
	LESPINARD Georges	Mécanique
	LONGEQUEUE Jean-Pierre	Physique Nucléaire Corpusculaire
	MOREAU René	Mécanique
	MORET Roger	Physique Nucléaire Corpusculaire
	PARIAUD Jean-Charles	Chimie - Physique
	PAUTHENET René	Physique du Solide - Cristallographie
	PERRET René	Automatique

.../...

MM.	PERRET Robert	Electrotechnique
	PIAU Jean-Michel	Mécanique
	PIERRARD Jean-Marie	Mécanique
	POLOUJADOFF Michel	Electrotechnique
	POUPOT Christian	Electronique - Automatique
	RAMEAU Jean-Jacques	Chimie
	ROBERT André	Chimie Appliquée et des matériaux
	ROBERT François	Analyse numérique
	SABONNADIERE Jean-Claude	Electrotechnique
Mme	SAUCIER Gabrielle	Informatique fondamentale et appliquée
M.	SOHM Jean-Claude	Chimie - Physique
Mme	SCHLENKER Claire	Physique du Solide - Cristallographie
MM.	TRAYNARD Philippe	Chimie - Physique
	VEILLON Gérard	Informatique fondamentale et appliquée
	ZADWORNY François	Electronique

CHERCHEURS DU C.N.R.S. (Directeur et Maître de Recherche)

M.	FRUCHART Robert	Directeur de Recherche
MM.	ANSARA Ibrahïm	Maître de Recherche
	BRONOEL Guy	Maître de Recherche
	CARRE René	Maître de Recherche
	DAVID René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	KAMARINOS Georges	Maître de Recherche
	KLEITZ Michel	Maître de Recherche
	LANDAU Ioan-Doré	Maître de Recherche
	MERMET Jean	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

Personnalités habilitées à diriger des travaux de recherche (décision du Conseil Scientifique)

E.N.S.E.E.G.

MM.	ALLIBERT Michel	
	BERNARD Claude	
	CAILLET Marcel	
Mme	CHATILLON Catherine	
MM.	COULON Michel	
	HAMMOU Abdelkader	
	JOUD Jean-Charles	
	RAVAINE Denis	
	SAINFORT	C.E.N.G.

.../...

MM. SARRAZIN Pierre
SOUQUET Jean-Louis
TOUZAIN Philippe
URBAIN Georges — Laboratoire des Ultra-Réfractaires ODEILLO

E.N.S.M.E.E.

MM. BISCONDI Michel
BOOS Jean-Yves
GUILHOT Bernard
KOBILANSKI André
LALAUZE René
LANCELOT Françis
LE COZE Jean
LESBATS Pierre
SOUSTELLE Michel
THEVENOT François
THOMAS Gérard
TRAN MINH Canh
DRIVER Julian
RIEU Jean

E.N.S.E.R.G.

MM. BOREL Joseph
CHEHIKIAN Alain
VIKTOROVITCH Pierre

E.N.S.I.E.G.

MM. BORNARD Guy
DESCHIZEAUX Pierre
GLANGEAUD François
JAUSSAUD Pierre
Mme JOURDAIN Geneviève
MM. LEJEUNE Gérard
PERARD Jacques

E.N.S.H.G.

M. DELHAYE Jean-Marc

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

MM. COURTIN Jacques
LATOMBE Jean-Claude
LUCAS Michel
VERDILLON André

Je tiens à remercier:

Monsieur L. BOLLIET, Professeur à l'Université des Sciences Sociales de Grenoble, qui m'a fait l'honneur de présider le jury de cette thèse.

Monsieur J.C. BOUSSARD, Professeur à l'Université de Nice, qui dirigea tout d'abord mon travail lorsqu'il était en poste à Grenoble, et qui m'a toujours prodigué encouragements et conseils. Je tiens à le remercier tout particulièrement pour la justesse de ses remarques et pour le temps considérable qu'il m'a consacré au cours de multiples entrevues à l'université de Nice.

Monsieur C. DELOBEL, Professeur à l'Université Scientifique et Médicale de Grenoble, Directeur du Laboratoire IMAG, qui a accepté de faire partie de ce jury.

Monsieur A. LENTIN, Professeur à l'Université de Paris V, qui a accepté de lire le manuscrit et d'être le rapporteur de cette thèse. Qu'il me soit permis de le remercier également pour l'intérêt qu'il a bien voulu porter à mon travail.

Monsieur G. VEILLON, Professeur à l'Institut National Polytechnique de Grenoble, Directeur de l'ENSIMAG, qui a assuré la direction de ma thèse, et dont les conseils m'ont été particulièrement précieux. Je tiens à lui exprimer toute ma reconnaissance pour l'aide qu'il m'a apportée.

Monsieur M. SKOLNICK, Associate Professor à l'Université d'Utah, pour l'intérêt qu'il a toujours porté à mon travail, et pour m'avoir permis d'engager une collaboration très fructueuse avec le groupe de recherche dont il est responsable à Salt Lake City. Je voudrais lui exprimer ici ma gratitude pour ses encouragements constants, la chaleur de son accueil, ainsi que mon admiration pour le travail de pionnier dont il est l'auteur dans le domaine de la reconstitution automatique de populations.

Je tiens également à remercier Monsieur N. GASTINEL, Professeur à l'Université Scientifique et Médicale de Grenoble, qui est à l'origine de cette étude,

ainsi que Madame M. BORNAREL, chercheur en démographie historique, qui m'a fourni le cas pratique à partir duquel j'ai pu expérimenter les différents logiciels.

Je remercie enfin Mesdames DOREL, DUBOIS et ROCHE, pour leur excellent travail de dactylographie, ainsi que l'ensemble du personnel du service de reprographie.

TABLE DES MATIERES

1.- INTRODUCTION

1.1. Présentation générale du problème

L'objet de cette étude est la définition d'une méthode de rapprochement d'informations floues. Cette qualification a été choisie faute d'un terme plus approprié ; il ne faut pas y voir une référence explicite à la notion d'ensemble flou (cf. § 1.3).

La notion de fichier est considérée ici d'un point de vue très général ; chaque enregistrement constitue l'image d'un élément du monde réel, définie par les valeurs d'un ensemble prédéterminé de caractéristiques (description)

La constitution de ces images pour un ensemble de départ donné, conformément à une description donnée (opération de saisie) est une phase critique de traitement en ce sens que :

- le choix de la description est un a priori tenant compte des traitements envisagés sur les données ; considérer d'autres traitements a posteriori peut être problématique du fait de l'inadéquation possible des informations disponibles ;
- selon les conditions dans lesquelles elle se déroule, l'opération de saisie peut donner lieu à une grande variété d'erreurs qui ne sont pas toujours rectifiables, voire seulement décelables ;
- le plus souvent, pour des raisons matérielles, l'opération de saisie ne peut être effectuée qu'une seule fois, de sorte que les données prennent d'emblée un caractère définitif.

Le traitement envisagé ici consiste à rechercher, dans des fichiers distinct relatifs à des ensembles non disjoints d'éléments du monde réel, tous les ensembles d'enregistrements relatifs à un même élément du monde réel. La méthode proposée ici tient compte du cas général où un tel traitement n'a pas été envisagé lors de la saisie initiale des informations (choix des descriptions de fichiers), et où cette saisie s'accompagne d'erreurs non redressables (par des saisies partielles complémentaires par exemple).

Le terme d'information floue recouvre donc ici deux caractères fondamentaux des informations traitées dans les différents fichiers :

- caractéristique non discriminante relativement aux éléments du monde réel : plusieurs éléments peuvent avoir la même valeur pour la caractéristique considérée ;

- caractéristique dont les valeurs sont sujettes à erreurs lors des opérations de saisie. Bien que toute caractéristique entre a priori dans cette catégorie, on doit plus particulièrement considérer celles pour lesquelles les possibilités de contrôle sont les plus limitées, ainsi que celles dont le processus de transmission est le moins fiable.

Compte tenu des propriétés ci-dessus des informations traitées, le processus de rapprochement peut se décomposer de la façon suivante :

- localisation dans les différents fichiers de départ des enregistremen pouvant logiquement être relatifs à un même élément du monde réel (enregistrements cohérents). Cette phase dite de couplage est essentiellement fondée sur l'évaluation de relations existant entre les informations disponibles. Tenir compte des erreurs revient ici à considérer éventuellement des relations de substitution (modèles de variation) établies à la suite d'une analyse détaillée du processus de transmission des informations. Erreurs et non discriminance se traduisent dans cette phase par une ambiguïté dans les résultats obtenus (plusieurs solutions possibles, éventualité de ne pas avoir obtenu la bonne). On cherchera donc avant tout à obtenir ici une couverture de la solution réelle ;

- sélection, dans un ensemble d'enregistrements cohérents produit par la phase de couplage, des solutions préservant un maximum de vraisemblance. Cette seconde phase, dite de jumelage, nécessite la définition d'un critère de ressemblance entre enregistrements, et une heuristique de sélection des solutions optimales par rapport à un critère prédéterminé (recherche des solutions les plus sûres, ou recherche d'un maximum de solutions) ;

- toute décision de jumelage prise dans la phase précédente doit se traduire par une dernière étape de restructuration des données.

La phase initiale de couplage est critique car elle détermine l'espace des solutions dans lequel toute décision définitive est prise. De ce point de vue, l'analyse préalable des informations devient très importante :

- contrôles systématiques pour limiter au maximum les erreurs pouvant être commises lors de la saisie informatique des informations, afin de ne pas les altérer davantage ;
- du fait de l'impossibilité de redresser les erreurs originelles, un effort particulier d'analyse doit être entrepris pour tenir compte au moins des possibilités connues de variation des informations (élaboration de modèles).

Toutes les expériences antérieures dans ce domaine soulignent l'importance de cette étude préliminaire (cf. 1.2).

Il est rapidement apparu que le problème, posé en termes aussi généraux, présentait une telle variété de composantes et un tel nombre d'interactions entre elles, que le plus sûr moyen de les préciser et d'en analyser les incidences était de passer par une étape de formalisation qui fait l'objet du chapitre II.

Le chapitre III est consacré à la définition des objectifs généraux fixés par la réalisation des différents logiciels, ainsi qu'à l'exposé de leurs principes généraux de fonctionnement (aspect algorithmique). Les produits actuellement réalisés sont présentés, ainsi que leurs caractéristiques d'implémentation et leurs restrictions éventuelles par rapport aux objectifs définis.

Si l'aspect formel de l'étude développée au chapitre II peut paraître important, l'aspect pratique développé au chapitre III l'est tout autant de notre point de vue. Le point de départ de cette étude a été une application concrète en Sciences Humaines, et les nombreux contacts établis depuis avec les spécialistes de diverses disciplines ont clairement démontré la double nécessité d'une analyse approfondie des composantes du problème et de l'existence de logiciels de niveau correspondant, pouvant être adaptés aux particularités de chaque application et ouverts à l'utilisateur final dont la compétence est bien souvent irremplaçable.

La conclusion donnée au chapitre IV porte tant sur les aspects théoriques que pratiques évoqués dans cette étude.

1.2. Les expériences antérieures et actuelles

Le point de départ de cette étude ayant été une application en démographie historique [A.BOC], il est utile de la situer par rapport aux expériences antérieures tentées dans le domaine. A notre connaissance, c'est dans ce secteur qu'ont été menées, sur le plan universitaire, les premières grandes applications concernant le rapprochement d'informations floues.

Science relativement récente, la démographie historique a pour objet l'étude des populations anciennes. On peut situer son véritable essor il y a seulement une vingtaine d'années lorsqu'elle reçut ses principaux fondements méthodologiques grâce aux travaux de MM. Fleury et Henry [A.FLH]. Depuis, on a assisté à une floraison de "monographies de villages" puisant leurs sources dans le patrimoine historique, jusqu'alors quasiment ignoré, des archives des villages et, plus récemment, des villes. Basées tout d'abord sur l'exploitation des registres paroissiaux (qui sont l'ancêtre de notre état-civil), les recherches puisent maintenant à des sources d'informations de plus en plus larges : recensements, registres notariaux, militaires, hospitaliers, etc. Cet élargissement des sources historiques utilisées s'accompagne d'un accroissement considérable du volume des informations traitées (pour les villes en particulier), et d'une ambition sans cesse accrue dans la finesse des résultats.

Il était donc naturel que, rapidement, les méthodes de dépouillement manuelles, pour efficaces qu'elles soient, deviennent de plus en plus impraticables, ne serait-ce que sur le plan matériel : la collecte et le dépouillement des informations prennent des années d'un travail de plus en plus fastidieux, les comptages et calculs statistiques deviennent de plus en plus longs et difficilement contrôlables.

Pour mieux situer cette "crise", disons que les travaux basés sur des méthodes manuelles pouvaient difficilement dépasser l'exploitation de 100.000 actes, ce qui représentait un travail de l'ordre de cinq années pour une personne. Actuellement, l'ambition des historiens démographes

porte sur des fichiers de l'ordre du million d'actes, et plus dans certains cas. Suivant l'évolution parallèle de l'informatique, les historiens démographes se sont tout d'abord tournés vers la mécanographie pour passer enfin le cap de l'application informatique lorsqu'ils ont pu trouver l'appui nécessaire dans des centres universitaires.

Dans le même temps, d'importantes équipes se créaient aux Etats-Unis, en Angleterre, en Italie, en Allemagne et au Canada notamment. En France, les premières expériences sont dues à MM. J.C. Perrot et Y. Daubeze [A.DAP], MM. J. dupaquier et M. Demonet [A.DUD], Mlle A. Chamoux [A.CHA]. A l'étranger, les principales applications sont dues à R.S. Schofield [A.SCH] en Angleterre, à MM. H. Charbonneau, J. Légaré, P. Beauchamp [A.BCH],et G. Bouchard [A.BOU] au Canada.

Il est important de remarquer que, dans tous les cas, les informaticiens n'ont joué qu'un rôle de réalisateurs d'outils généralement très liés à l'application considérée. Notons également que, dans toutes ces applications, le problème crucial a été et reste le traitement des informations floues, la restructuration des données ainsi que les traitements statistiques ne posant pas de difficulté majeure. Enfin, la plupart des logiciels existants ont pour support des langages et des systèmes très classiques; les systèmes de base de données notamment n'ont fait qu'une apparition très récente dans ce domaine.

Au-delà de ces premières expériences, le besoin s'est rapidement fait sentir d'une approche méthodologique plus adaptée à l'outil informatique, qui permette notamment de mieux utiliser sa puissance de calcul et ses possibilités de structuration de l'information.

Chronologiquement, notre application a débuté sensiblement à cette époque, et l'étude présentée dans le chapitre II, bien que n'ayant pas été directement motivée par ce besoin, y a rapidement trouvé sa justification, en même temps que de larges possibilités de contacts et d'échanges s'ouvraient et permettaient de mieux cerner les objectifs à suivre dans l'élaboration des logiciels adaptés à la résolution de ce type de problème.

Actuellement une nouvelle classe d'applications très prometteuse dans ce domaine est en cours de développement : la génétique des populations. La plus importante est certainement le projet développé par M. Skolnick aux USA[B.SKO]sur la population des Mormons d'Utah. Traitant des données plus actuelles, ce type d'applications trouve actuellement des débouchés extrêmement utiles, en même temps qu'il offre des aspects scientifiques probablement assez inédits sur le plan informatique.

1.3. Les domaines scientifiques proches

Les secteurs de la reconnaissance des formes, de la classification automatique paraissent relativement proches de notre étude en ce sens que, dans tous les cas, les données de départ sont floues :

- forme plus ou moins nette,
- critère de classification dont on ignore a priori l'efficacité,
- caractéristiques floues donnant une image faussée et incomplète, de la réalité, et interdisant la reconnaissance directe des éléments du monde réel représentés.

De plus, l'objectif est dans tous les cas l'obtention d'un invariant extrait de l'information disponible :

- rattachement d'une forme à une catégorie prédéterminée,
- rattachement d'un élément à une classe donnée,
- rattachement d'un ensemble d'images à un élément du monde réel.

Il nous est cependant apparu que les caractéristiques spécifiques au problèm étudié nous interdisaient pratiquement l'application directe de ces techniques :

- très gros volumes de données (quelques dizaines de milliers à plusieurs centaines de millions d'enregistrements),
- grande variété de descripteurs par fichier (plusieurs dizaines),
- grand nombre de valeurs pour certains descripteurs (plusieurs centaines à plusieurs milliers).

Dans ces conditions, l'utilisation des méthodes matricielles couramment utilisées en reconnaissance des formes ou en classification automatique sont à exclure sauf éventuellement pour traiter des cas déjà fortement délimités dans le corpus des données.

La théorie des ensembles flous paraissait également très proche de nos pr préoccupations, et il a été envisagé un temps de l'utiliser comme base de l'étude formelle du chapitre II. Finalement, le formalisme mathématique courant s"étant avéré suffisant, nous l'avons préféré pour l'instant.

En définitive, les domaines auxquels nous avons le plus emprunté dans cette étude sont la théorie de l'information, la théorie des questionnaires, l'intelligence artificielle, la documentation automatique et les bases de données.

Cette variété dans les bases de notre travail ne fait qu'illustrer la multiplicité des aspects du problème posé.

2.- ETUDE FORMELLE

CHAPITRE 2.1. - Notations - Définitions de base

2.1. Notations-définitions de base

2.1.1. Notations

L'annexe 1 contient un récapitulatif de l'ensemble des notations utilisées dans ce chapitre. Si elles résultent souvent de conventions personnelles, c'est soit par souci de clarté, soit parce que celles présentées dans la littérature sont très variables selon les domaines considérés.

2.1.2. Saisie-Fichier

Certaines définitions données ci-dessous peuvent paraître restrictives relativement aux concepts qu'elles recouvrent (ex : caractéristiques, fichiers, ...) ; il faut les interpréter comme l'expression des seules propriétés qui nous intéressent ici.

a) Ensemble de départ

Nous appelons ensemble de départ un ensemble fini A d'éléments notés a_j appelés acteurs.

Soit $A = \{a_j\}$, $j \in \{1,\ldots,N\}$ et $|A| = N$ (cardinal de A).

Nous notons $A^i = \{a^i_j\}$ lorsque nous avons à considérer plusieurs ensembles de départ.

b) Caractéristique

Nous appelons caractéristique K une application des éléments de A sur un ensemble fini de valeurs $V = \{v_\ell\}$.

Soit : $\forall a \in A$, $K(a) = v \in V$.

Lorsque plusieurs caractéristiques sont définies sur un même ensemble A, nous notons ces caractéristiques K_k, et V_k les ensembles de valeurs correspondants, avec $V_k = \{v_{k,\ell}\}$.

c) Valeur indéterminée

Nous considérons que toute caractéristique possède dans son ensemble de valeurs un élément noté 'u', appelé valeur indéterminée ou valeur inconnue.

d) Caractéristique inverse

La caractéristique inverse associée à K, notée $\bar{K}$, est l'application de V dans les parties de A. Etant donné $v_\ell \in V$, nous notons A_ℓ le sous-ensemble de A contenant tous les éléments de A tels que $K(a) = v_\ell$.

Lorsqu'il est nécessaire de distinguer les caractéristiques entre elles, nous notons $A_{k,\ell} = \bar{K}_k(v_{k,\ell}) \subseteq A$ le sous-ensemble de A tel que :

$$\forall a \in A_{k,\ell},\ K_k(a) = v_{k,\ell}.$$

e) Description

Une description D^i est un ensemble fini et ordonné de caractéristiques définies sur A^i.

Soit $D^i = \{K_1^i,\ldots,K_d^i\}$ que nous noterons $D = \{K_1,\ldots,K_d\}$ quand il n'y a pas d'ambiguïté sur l'ensemble de départ considéré.

Quel que soit $a \in A$, le d-uplet $(K_1(a),\ldots,K_2(a))$ est un élément de l'ensemble produit $\pi = V_1 \times V_2 \times \ldots \times V_d$.

Nous considérons désormais cet ensemble comme un alphabet, et π^+ l'ensemble des mots non vides composés d'éléments de cet alphabet.

Exemple : Soit D = {NOM,DATE} = $\{K_1,K_2\}$, et V_1 = {'DUPOND','DURAND'},
V_2 = {1970,1972}.

On a alors $\pi = V_1 \times V_2$ = {('DUPOND',1970), ('DUPOND',1972)...,('DUPOND',u),
..., (u,u)}.

Un mot de π^+ est par exemple :
('DUPOND',1970), ('DUPOND',1970), ('DURAND',u).

+++

f) Enregistrement

On entend par enregistrement e toute occurrence d'un élément de π dans un mot de π^+ ; un enregistrement est donc caractérisé par sa valeur (l'élément de π), et par sa position dans le mot (symboliquement notée comme indice).

La description D définissant π étant ordonnée, tout enregistrement relatif à une description donnée est un d-uplet ordonné des valeurs de ses caractéristiques composantes.

g) Fichier

Etant donné un mot de π^+, on entend par fichier E l'ensemble totalement ordonné des occurrences des éléments de π qui le composent. La relation d'ordre total est communément appelée ordre séquentiel des enregistrements (elle est symbolisée ici par la séquence des indices).

Exemple : Au mot donné dans l'exemple précédent correspond le fichier :

$$E = \{('DUPOND',1970)_1,('DUPOND',1970)_2,('DURAND',u)_3\} = \{e_1,e_2,e_3\}.$$

+++

Dans tout ce qui suit, nous faisons abstraction de cette relation pour ne retenir que l'aspect "ensemble d'occurrences d'éléments de π" ; l'ordre n'a d'importance qu'au niveau des méthodes d'accès que l'on peut vouloir définir sur le fichier (les numéros de séquence peuvent être mis en relatio avec des clés d'accès pour définir une fonction d'adressage).

h) Agrégat

Considérons deux fichiers E^1, E^2 correspondant respectivement aux ensembles de départ A^1, A^2 pour les descriptions D^1, D^2, on a :

$$E^1 = \{(c_k)_i\},\ i \in \{1,\ldots,m\},\ c_k \in \pi^1$$
$$E^2 = \{(b_\ell)_j\},\ j \in \{1,\ldots,n\},\ b_\ell \in \pi^2$$

Un agrégat $\mathcal{E}$ des fichiers E^1 et E^2 est constitué par tout ensemble totalemer ordonné des occurrences d'éléments de π^1 et π^2 constituant E^1 et E^2.

Exemple : Soit $E^1 = \{(c_1)_1, (c_3)_2, (c_3)_3\},\ \forall c_i \in \pi^1$

$$E^2 = \{(b_1)_1, (b_2)_2, (b_2)_3, (b_3)_4\},\ \forall b_i \in \pi^2.$$

Les ensembles suivants sont des agrégats de E^1, E^2 :

$$\mathcal{E}^1 = \{(c_3)_1, (b_1)_2, (b_2)_3, (c_1)_4, (c_3)_5, (b_3)_6, (b_2)_7\}$$
$$\mathcal{E}^2 = \{(b_1)_1, (b_3)_2, (c_1)_3, (b_2)_4, (c_3)_5, (b_2)_6, (c_3)_7\}.$$

+++

Nous notons $\mathcal{E} = E^1+E^2$ pour exprimer l'agrégation de E^1 et E^2 dans $\mathcal{E}$.

i) Saisie

La saisie d'un élément a_j de A, relativement à une description D, regroupe l'ensemble des opérations conduisant à la constitution d'un enregistrement e_m, occurrence d'un élément de π.

Soit $D = \{K_1,\ldots,K_d\}$, on a :

$e_m = (K_1(a_j),\ldots,K_d(a_j)) \in E$.

La saisie des éléments d'un ensemble A relativement à D est une opération conduisant à la constitution d'un fichier ; nous considérons toujours que l'on peut caractériser la correspondance entre les éléments de A et ceux de E par une bijection notée S représentant une saisie unique de A. S est donc une application de A sur E telle que $\bar{S}$ existe. Quand cela est nécessaire, on note S_k^i où i représente l'ensemble A^i de départ, k la description D_k utilisée.

j) Caractéristique fichier

Une caractéristique fichier est une application C de E dans un ensemble de valeurs V ainsi définie :

$\forall e \in E,\ C(e) = K(\bar{S}(e)) \in V$.

Exemple : Soit $A = \{a_1,a_2,a_3\} \xleftrightarrow{S} \{e_1,e_2,e_3\}$

avec $D = \{AGE,PROFESSION\} = \{K_1,K_2\}$

et : $K_1(a_1) = 25$; $K_1(a_2) = 30$; $K_1(a_3) = 25$

$K_2(a_1) =$ 'PROGRAMMEUR' ; $K_2(a_2) =$ 'MENUISIER' ; $K_2(a_3) =$ 'TAILLEUR'.

A la description D, pour l'ensemble A et la saisie S, correspond le fichier E = {(25,'PROGRAMMEUR'), (30,'MENUISIER'), (25,'TAILLEUR')} et les caractéristiques fichier C_1 et C_2 correspondant à K_1 et K_2 :

$C_1(e_1) = C_1(e_3) = 25$; $C_1(e_2) = 30$

$C_2(e_1) =$ 'PROGRAMMEUR' ; $C_2(e_2) =$ 'MENUISIER' ; $C_2(e_3) =$ 'TAILLEUR'.

+++

De la même façon que précédemment, nous pouvons définir pour C l'applicati inverse de V dans les parties de E, notée $\bar{C}$; étant donné $v_{k,\ell} \in V_k$, nous notons $E_{k,\ell}$ le sous-ensemble de E tel que :

$$\forall e \in E_{k,\ell},\ C_k(e) = v_{k,\ell} \in V.$$

Pour toute caractéristique C, l'application $\bar{C}$ détermine une partition de E ; construire cette partition constitue l'inversion de E pour la caractéristique C.

k) Caractéristique discriminante

Une caractéristique K est dite discriminante relativement à l'ensemble A sur lequel elle est définie, si tous les éléments de A possèdent des valeurs distinctes pour cette caractéristique.

Une telle caractéristique est donc une bijection de A dans V et :

$$\forall v \in V,\ \bar{K}(v) = \{a_j\} \in A.$$

Nous dirons qu'un ensemble de caractéristiques $K_1,\ldots,K_n$ définies sur A est discriminant si tous les éléments de A possèdent des n-uplets différents de valeurs de ces caractéristiques.

Soit :

$$\forall v_{1,k} \in V_1,\ldots,\ \forall v_{n,\ell} \in V_n,\ A_{1,k} \cap \ldots \cap A_{n,\ell} = \{a_j\} \in A \text{ ou } \emptyset.$$

Dans ces conditions, l'ensemble discriminant de caractéristiques établit une bijection entre A et un sous-ensemble de $\pi = V_1 \times \ldots \times V_n$.

Il est important de noter que cette propriété est définie relativement à un ensemble de départ particulier, et qu'on ne peut conclure au caractère discriminant (en général) d'une caractéristique par le seul examen de ses valeurs pour une saisie donnée.

Exemple : La caractéristique Numéro de Sécurité Sociale est discriminante pour l'ensemble de la population française considérée sur une période d'un siècle. Ceci est une propriété établie a priori du fait que la codification réalise une énumération des individus nés dans une période déterminée.

La caractéristique NOM n'est pas discriminante même si, pour une saisie donnée, il se trouve que tous les individus ont des noms distincts ;

inversement une caractéristique discriminante peut ne l'être apparamment pas (au vu des valeurs enregistrées) du fait d'erreurs à la saisie.

+++

La question de l'évaluation du pouvoir discriminant d'une caractéristique est importante, et nous y reviendrons en 2.4.

2.1.3. Saisies multiples - Conséquences

Considérons à présent n ensembles de départ $A^1,\ldots,A^n$ non nécessairement disjoints, et n saisies $S^1,\ldots,S^n$ respectives de ces ensembles relativement aux descriptions $D^1,\ldots,D^n$. On obtient n fichiers $E^1,\ldots,E^n$.

Considérons l'agrégat $E = E^1+E^2+\ldots+E^n$, et $A = A^1 \cup A^2 \cup \ldots \cup A^n$.

Par définition (cf. 2.1.2.h), les différents fichiers E^i constituent une partition de E et nous supposons que l'on peut toujours, étant donné un élément de E, déterminer à quel E^i il appartient (on peut toujours discerner les différents fichiers composant l'agrégat).

a) Enregistrements homologues

Dans le cas où les A^i ne sont pas disjoints, on aura dans E des enregistrements images par différentes saisies d'un même élément a de A ; nous dirons que ces enregistrements sont homologues, et nous noterons $\sim$ cette relation.

Soit :

$$\forall e_i,e_j \in E,\ e_i \sim e_j \iff \exists a \in A,\ \exists S^k,\ \exists S^\ell \mid S^k(a) = e_i,\ S^\ell(a) = e_j.$$

Propriété : La relation d'homologie est une relation d'équivalence sur E.

Démonstration : La réflexivité et la symétrie sont évidentes ; nous ne démontrons ici que la transitivité.

$$\forall e_i,e_j \in E,\ e_i \sim e_j \iff \exists a_1 \in A,\ \exists S^k,\ \exists S^\ell \mid S^k(a_1) = e_i,\ S^\ell(a_1) = e_j$$

$$\forall e_k \in E,\ e_j \sim e_k \iff \exists a_2 \in A,\ \exists S^m,\ \exists S^n \mid S^m(a_2) = e_j, S^n(a_2) = e_k.$$

Les saisies étant des bijections (cf. 2.1.2.i), e_j ne peut correspondre qu'à un seul élément de A, et on a donc $a_1 = a_2$, et $S^\ell = S^m$.

D'où :

$$\forall e_i, e_j, e_k \in E,\ e_i \sim e_j \text{ et } e_j \sim e_k \Rightarrow \exists a_l \in A,\ \exists S^k,\ \exists S^n \mid S^k(a_l) = e_i,\ S^n(a_l) = e_k \Rightarrow e_i \sim e_k$$

+++

Nous notons H la partition de E par la relation d'homologie ; H_j la classe d'enregistrements homologues correspondant à $a_j \in A$.

b) Caractéristiques synonymes

A toute caractéristique définie sur un ensemble de départ correspond une propriété déterminée des éléments de cet ensemble ; les valeurs correspondantes de la caractéristique représentent des modalités discrètes de cette propriété.

Ceci nous conduit à considérer le problème posé par l'existence d'expressio particulières d'une même propriété lors de différentes saisies ; on peut avoir :

- des identificateurs différents,
- des ensembles de valeurs correspondant à des codifications différent des modalités de la propriété.

Soit $\mathcal{D}$ l'ensemble des propriétés des éléments d'un ensemble A que l'on sait exprimer par la définition de caractéristiques (propriétés pour lesquelles on sait définir un nombre fini de modalités discrètes, donc codifiables dans un ensemble fini de valeurs). Nous dirons que toutes les caractéristiq correspondant à une même propriété de $\mathcal{D}$ sont des caractéristiques synonymes.

On peut toujours retrouver les règles de transcodification entre les différents modes de représentation d'un même ensemble fini de valeurs, aussi supposerons-nous désormais que, pour une même propriété, toutes les caractéristiques synonymes correspondantes ont des ensembles de valeurs de même nature.

Dans ce qui suit, nous donnons à toutes les caractéristiques synonymes (quelle que soit la description à laquelle elles appartiennent) le même indice inférieur qui est celui de la propriété exprimée (élément de $\mathcal{D}$), et nous distinguons la description dont elles font partie par un indice supérieur.

Exemple : Soit $\mathcal{D}$ = {NOM, PRENOM, DATE-NAISSANCE, AGE, PROFESSION, ...} = $\{K_1, K_2, K_4, K_5, \ldots\}$ dont les ensembles de valeurs sont $V_1, V_2, \ldots$
un ensemble de propriétés relatives à A, ensemble d'acteurs-personnes.

Nous pouvons avoir par exemple :

$A^1 \subset A$, D^1 = {NOM, PROFESSION} = $\{K_1^1, K_5^1\}$ et $A^1 \xleftrightarrow{S^1} E^1$

$A^2 \subset A$, D^2 = {NOM,PRENOM,METIER} = $\{K_1^2, K_2^2, K_5^2\}$ et $A^2 \xleftrightarrow{S^2} E^2$.

Les caractéristiques K_1^1 et K_1^2, K_5^1 et K_5^2 sont respectivement synonymes et leurs ensembles de valeurs sont notés de manière analogue
$V_1^1, V_1^2 \subseteq V_1, V_5^1, V_5^2 \subseteq V_5$.

+++

Le même système de notation est étendu aux caractéristiques fichier correspondantes.

c) Valeurs homologues

Lorsqu'un élément a de A appartient au domaine de définition d'un ensemble de caractéristiques synonymes, l'ensemble de valeurs prises par ces caractéristiques pour a est appelé ensemble de valeurs homologues. Nous notons $\mathcal{H}$ cette relation.

Soit $K_k^1, \ldots, K_k^n$ un ensemble de caractéristiques synonymes correspondant à la propriété K_k ; pour tout a appartenant à l'ensemble de définition commun à ces caractéristiques correspond l'ensemble de valeurs homologues :

$$\{K_k^1(a), \ldots, K_k^n(a)\} \subset V_k^1 \cup \ldots \cup V_k^n.$$

Exemple : Soit $A = \{a_1, a_2, a_3\}$ avec $A^1 = \{a_1, a_2\}$, $A^2 = \{a_2, a_3\}$
et D^1 = {NOM,METIER} = $\{K_1^1, K_5^1\}$, D^2 = {METIER,AGE} = $\{K_5^2, K_4^2\}$.
a_2 appartient à la définition des caractéristiques synonymes K_5^1 et K_5^2.
Si $K_5^1(a_2)$ = 'PROGRAMMEUR' et $K_5^2(a_2)$ = 'ANALYSTE', ces deux caractéristiques déterminent pour a_2 l'ensemble de valeurs homologues
{'PROGRAMMEUR', 'ANALYSTE'} $\subset V_5$.

+++

Une définition équivalente des valeurs homologues peut être donnée pour les caractéristiques fichiers correspondantes $C_k^1, \ldots, C_k^n$.

Pour $H_j \subseteq E$ correspondant à $a_j \in A$, les valeurs de $C_k^1, \ldots, C_k^n$ relatives aux éléments de H_j constituent un ensemble de valeurs homologues.

Par définition, la relation d'homologie entre valeurs est réflexive et symétrique ; elle n'est généralement pas transitive.

Démonstration : Considérons l'ensemble de n caractéristiques synonymes $K_i^1, \ldots, K_i^n$ et $V = V_i^1 \cup \ldots \cup V_i^n$; par définition de l'homologie entre valeurs, on a :

$$\forall v_j, v_k \in V_i,\ v_j \not\sim v_k \iff \exists a_1 \in A,\ \exists K_i^p \mid K_i^p(a_1) = v_j,\ K_i^q(a_1) = v_k$$

$$\forall v_\ell \in V_i,\ v_k \not\sim v_\ell \iff \exists a_2 \in A,\ \exists K_i^r,\ \exists K_i^s \mid K_i^r(a_2) = v_k,\ K_i^s(a_2) = v_\ell.$$

Ces deux conditions peuvent être vérifiées avec $a_1 \neq a_2$:

- si K_i^q et K_i^r sont différentes,
- si K_i^q et K_i^r sont la même caractéristique, il suffit que K_i^q ne soit pas une bijection, et donc que $K_i^q(a_1) = K_i^q(a_2)$, donc $v_j \not\sim v_k$ et $v_k \not\sim v_\ell \not\Rightarrow v_j \not\sim v_\ell$.

+++

d) Caractéristiques invariantes

On désigne ainsi des caractéristiques synonymes dont les ensembles de valeurs homologues ne peuvent comporter qu'un seul élément (quel que soit le nombre de saisies considéré).

Par opposition, on appellera caractéristique non invariante toute caractéristique dont les ensembles de valeurs homologues peuvent contenir plusieurs éléments.

Exemple : Un numéro de Sécurité Sociale est une caractéristique invariante. Les caractéristiques du type profession, âge, lieu de résidence sont non-invariantes.

+++

e) Relations entre caractéristiques

Nous considérons deux catégories de relations pouvant exister entre des caractéristiques :

- les relations dites internes entre éléments d'une même description,
- les relations dites externes entre éléments de descriptions différentes.

Nous notons $\mathbb{R}_{j,k}$ une relation définie sur $V_j \times V_k$, et nous différencions éventuellement diverses relations définies sur un même ensemble produit par un indice supérieur.

Pour une relation interne R_{jk} entre K_j et K_k définies sur A, on a par définition :

$$\forall a \in A,\ K_j(a)\ R_{jk}\ K_k(a) \text{ ou } a \in A \Rightarrow K_j(a)\ R_{jk}\ K_k(a).$$

Ce type de relation se traduit, pour les caractéristiques fichiers correspondantes C_j et C_k, par :

$$e \in E \Rightarrow C_j(e)\ R_{jk}\ C_k(e).$$

Pour une relation externe R_{jk} entre deux caractéristiques K_j^i et K_k^ℓ définies respectivement sur A^i et A^ℓ, on a par définition :

$$\forall a \in A^i \cap A^\ell,\ K_j^i(a)\ R_{jk}\ K_k^\ell(a) \text{ ou } a \in A^i \cap A^\ell \Rightarrow K_j^i(a)\ R_{jk}\ K_k^\ell(a).$$

Ce type de relation se traduit, pour les caractéristiques fichiers C_j^i et C_k^ℓ correspondantes de E^i et E^ℓ, par

$$\forall e_m \in E^i,\ \forall e_n \in E^\ell,\ e_m\ \gamma_{jk}^{i\ell}\ e_n \Rightarrow C_j^i(e_m)\ R_{jk}\ C_k^\ell(e_n).$$

Nous notons γ_{jk} la relation de cohérence entre enregistrements définie sur $E^i \times E^\ell$, induite par R_{jk}, quand il n'y a pas d'ambiguité sur les fichiers concernés.

f) Cas particulier

Un cas particulier important est celui des relations externes entre caractéristiques synonymes ; si K_j^i et K_j^ℓ sont synonymes relativement à la propriété K_j (cf. 2.1.3.b), toute relation R_{jj} entre ces caractéristiques peut être considérée comme étant définie sur $V_j \times V_j$, avec $V_j = V_j^i \cup V_j^\ell$.

La relation $\gamma_{jj}^{i\ell}$ peut être alors considérée comme une relation sur $E \times E$, avec $E = E^i + E^\ell$ (cf. 2.1.2.h) ; elle sera notée plus simplement $\gamma_j^{i\ell}$ ou γ_j.

g) Cohérence totale entre enregistrements

Par extension, deux enregistrements e_m et e_n appartenant respectivement aux fichiers E^i et E^ℓ sont dits totalement cohérents si toutes les relations externes existantes entre les caractéristiques constituant les descriptions de E^i et E^ℓ sont vérifiées.

Nous notons $\mathcal{R}^{i\ell}$ cette relation définie sur $E^i \times E^\ell$.

Soit, pour un ensemble de relations externes $R_{jk},\ldots,R_{pq}$ définies entre des caractéristiques appartenant aux descriptions de E^i et E^ℓ :

$$\forall e_m \in E^i,\ \forall e_n \in E^\ell,\ e_m\ \mathcal{R}^{i\ell}\ e_n \iff e_m\ \mathcal{R}_{jk}\ e_n\ \underline{\text{et}}\ \ldots\ \underline{\text{et}}\ e_m\ \mathcal{R}_{pq}\ e_n.$$

Les propriétés de cette relation dépendent naturellement de celles des relations composantes ; un cas particulier important est celui où toutes les relations concernent des caractéristiques synonymes. D'après 2.1.3.f, la relation $\mathcal{R}^{i\ell}$ est alors définie sur $E \times E$. D'après l'expression ci-dessus de la cohérence totale, il est en particulier évident que cette relation est une relation d'équivalence si et seulement si toutes les relations composantes sont des relations d'équivalence.

Notons $\bar{\mathcal{R}}^{i\ell}$ la relation inverse de $\mathcal{R}^{i\ell}$, obtenue par symétrisation des relations composantes de $\mathcal{R}^{i\ell}$.

h) Classe de cohérence

Etant donné une relation de cohérence $\mathcal{R}^{i\ell}$ définie sur $E^i \times E^\ell$, considérons le graphe induit par cette relation sur $E = E^i + E^\ell$.

Par définition, nous appelons classe de cohérence mutuelle toute partie de E constituant une composante connexe de ce graphe.

Exemple : Soit $A = \{a_1,a_2,a_3,a_4\} = A^1 \cup A^2$, avec $A^1 = \{a_1,a_2,a_3\}$, $A^2 = \{a_2,a_3,a_4\}$.

Soit D^1 = {NOM,DATE-NAISSANCE} = $\{K_1^1,K_2^1\}$

D^2 = {NOM,DATE-MARIAGE} = $\{K_1^2,K_3^2\}$

et les fichiers correspondants pour les saisies S^1 et S^2 :

E^1 = {('DUPOND',1900), ('DURAND',1910), ('DUPOND',1920)} = $\{e_1,e_2,e_3\} \subset E$

E^2 = {('DUPOND',1940), ('DUPOND',1920), ('DURAND',1920)} = $\{e_4,e_5,e_6\} \subset E$

avec $E = E^1+E^2$.

Etant donné les relations externes :

R_{11} : $\forall e_i \in E^1$, $\forall e_j \in E^2$, $e_i \sim e_j \Rightarrow K_1^1(e_i) = K_1^2(e_j)$.

R_{22} : $\forall e_i \in E^1$, $\forall e_j \in E^2$, $e_i \sim e_j \Rightarrow K_3^2(e_j) - K_2^1(e_i) \geq 18$.

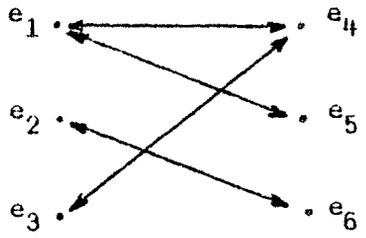
La relation de cohérence totale $\mathcal{Y}^{12}$ correspondante est définie par :

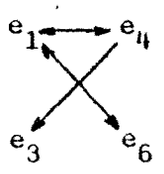
$\forall e_i \in E^1$, $\forall e_j \in E^2$, $e_i \sim e_j \Rightarrow e_i \mathcal{Y}_{11} e_j$ et $e_i \mathcal{Y}_{23} e_j$

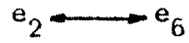
soit, sur $E^1 \times E^2$:

(e_1,e_4), (e_1,e_5), (e_2,e_6), (e_3,e_4)

et le graphe induit sur E avec ses composantes connexes :







E_1 E_2

$E = E^1+E^2$

Remarquons enfin que $\mathcal{Y}_{11}$ (égalité des noms) peut être considérée comme une relation d'équivalence sur E (caractéristiques K_1^1 et K_1^2 synonymes), alors que $\mathcal{Y}_{23}$ ne peut être considérée comme une relation sur E (caractéristiques K_2^1 et K_3^2 non synonymes).

+++

L'objectif poursuivi dans la construction des classes de cohérence est de déterminer une partition de E dont chaque élément regroupe, pour tout enregistrement lui appartenant, l'ensemble des enregistrements qui lui sont cohérents.

L'intérêt de cette opération est évident lors de la phase de décision : l'ensemble des solutions possibles à tout cas de rapprochement envisagé est limité à une classe de cohérence. De fait, les classes de cohérence partitionnent le problème général en sous-problèmes bien distincts.

i) Construction des classes de cohérence : couplage

Les graphes induits sur E par $\mathcal{X}^{i\ell}$ et la relation inverse correspondante $\bar{\mathcal{X}}^{i\ell}$ sont identiques ; étant donné deux fichiers E^i et E^ℓ, une relation de cohérence totale $\mathcal{X}^{i\ell}$, la construction des classes correspondantes peut donc s'effectuer indifféremment sur $E^i \times E^\ell$ par $\mathcal{X}^{i\ell}$, ou sur $E^\ell \times E^i$ par $\bar{\mathcal{X}}^{i\ell}$ (symétrie).

L'opération de construction des classes de cohérence est appelée couplage des fichiers.

j) Coupes dans une relation

Etant donné $R \subseteq E \times F$, nous notons R_m la coupe de R dans F pour tout élément $x_m \in E$, $\bar{R}_n$ la coupe de $\bar{R}$ dans E pour tout élément $y_n \in F$; soit :

(a) $\forall x_m \in E,\ R_m = \{y_n \in F \mid x_m R y_n\} \subseteq F$

$\forall y_n \in F,\ \bar{R}_n = \{x_m \in E \mid y_n \bar{R} x_m\} \subseteq E$

Cette notation se transpose de la façon suivante aux différentes relations définies jusqu'ici :

(b) - pour toute relation $R_{jk} \subseteq V_j \times V_k$, les coupes sont notées :

$\forall v_r \in V_j,\ R_{jk,r} \subseteq V_k$ et $\forall v_s \in V_k,\ \bar{R}_{jk,s} \subseteq V_j$

- pour la relation correspondante $\mathcal{X}_{jk} \subseteq E^i \times E^\ell$, les coupes sont notées :

$\forall e_m \in E^i,\ \mathcal{X}_{jk,m} \subseteq E^\ell$ et $\forall e_n \in E^\ell,\ \bar{\mathcal{X}}_{jk,n} \subseteq E^i$

- pour la relation de cohérence $\mathcal{X}^{i\ell} \subseteq E^i \times E^\ell$, les coupes sont notées :

$\forall e_m \in E^i,\ \mathcal{X}^{i\ell}_m \subseteq E^\ell$ et $\forall e_n \in E^\ell,\ \bar{\mathcal{X}}^{i\ell}_n \subseteq E^i$

- ou plus simplement $\mathcal{X}_m$ et $\mathcal{X}_n$ quand il n'y a pas d'ambiguité à propos des fichiers concernés.

Rappelons (cf. 2.1.2.j) que l'ensemble noté $E^{\ell}_{k,s} \subseteq E^{\ell}$ est donné par $\bar{C}_k(v_s)$; on a donc :

(c) $\forall e_m \in E^i$, $\mathfrak{X}_{jk,m} = \bigcup_{v_s \in R_{jk,r}} E^{\ell}_{k,s}$, avec $v_r = C_j(e_m)$.

De même :

$\forall e_n \in E^{\ell}$, $\bar{\mathfrak{X}}_{jk,n} = \bigcup_{v_r \in \bar{R}_{jk,s}} E^{i}_{j,s}$, avec $v_s = C_k(e_n)$.

On a donc la propriété suivante :

Propriété 1 :

$$\forall v_r \in V_j,\ \forall e_m, e_n \in E^i_{j,r},\ \mathfrak{X}_{jk,m} = \mathfrak{X}_{jk,n}$$

et

$$\forall v_s \in V_k,\ \forall e_m, e_n \in E^{\ell}_{k,s},\ \bar{\mathfrak{X}}_{jk,m} = \bar{\mathfrak{X}}_{jk,n}$$

Pour une relation R_j entre caractéristiques synonymes, on note $\mathfrak{X}_{j,m}$ la coupe de $\mathfrak{X}_j$ dans E pour tout élément e_m de E, et $\bar{\mathfrak{X}}_{j,m}$ celle de $\bar{\mathfrak{X}}_j$.

Considérons à présent un ensemble de relations $R_{jk} \subseteq V^i_j \times V^{\ell}_k, \ldots, R_{pq} \subseteq V^i_p \times V^{\ell}_q$ et notons $\Pi^i = V^i_j \times \ldots \times V^i_p$, $\Pi^{\ell} = V^{\ell}_k \times \ldots \times V^{\ell}_q$. Notons :

(d) $\Pi^i = \{\alpha\} = \{(v_{j,r}, \ldots, v_{p,s})\}$

et

$\Pi^{\ell} = \{\beta\} = \{(v_{k,t}, \ldots, v_{q,u})\}$.

Considérons la relation $R^{i\ell} \subseteq \Pi^i \times \Pi^{\ell}$ définie par :

(e) $\forall \alpha \in \Pi^i,\ \forall \beta \in \Pi^{\ell},\ \alpha R^{i\ell} \beta \iff v_{j,r}\ R_{jk}\ v_{k,t}$ et ... et $v_{p,s}\ R_{pq}\ v_{q,u}$

Les coupes de $R^{i\ell}$ dans Π^{ℓ} pour les éléments α de Π^i sont notées $R^{i\ell}_{\alpha}$; celles de la relation inverse $\bar{R}^{i\ell}$ pour les éléments β de Π^{ℓ} sont notées $\bar{R}^{i\ell}_{\beta}$.

Comme précédemment, nous noterons plus simplement R_{α} et $\bar{R}_{\beta}$ lorsqu'il n'y a pas d'ambiguïté à propos des fichiers concernés.

Il est facile de vérifier que, d'après les définitions (d) et (e), on a :

(e) $\forall \alpha \in \Pi^i$, $R_\alpha = R_{jk,r} \times \ldots \times R_{pq,s} \subseteq V_k \times \ldots \times V_q = \Pi^\ell$

et

$\forall \beta \in \Pi^\ell$, $\bar{R}_\beta = \bar{R}_{jk,t} \times \ldots \times \bar{R}_{pq,u} \subseteq V_j \times \ldots \times V_p = \Pi^i$

α et β étant définis d'après (d), notons E^i_α et E^ℓ_β les sous-ensembles de E^i et E^ℓ définis par

(f) $\forall \alpha \in \Pi^i$, $E^i_\alpha = E^i_{j,r} \cap \ldots \cap E^i_{p,s} \subseteq E^i$

$\forall \beta \in \Pi^\ell$, $E^\ell_\beta = E^\ell_{k,t} \cap \ldots \cap E^\ell_{q,u} \subseteq E^\ell$.

Selon l'interdépendance pouvant exister entre les partitions de E^i par les différentes caractéristiques inverses $\bar{C}_j,\ldots,\bar{C}_p$, on peut naturellement avoir $E^i_\alpha = \emptyset$. Il en va de même pour les partitions de E^ℓ définies par $\bar{C}_k,\ldots,\bar{C}_q$.

Il est facile de vérifier que $\forall \alpha \in \Pi^i$, $\forall \beta \in \Pi^\ell$, les ensembles E^i_α, E^ℓ_β correspondants déterminent une partition de E^i et E^ℓ respectivement. Nous notons E^i/C^i et E^ℓ/C^ℓ ces partitions.

Par extension de la notion de caractéristique inverse, on note :

(g) $\forall e_m \in E^i_\alpha$, $C^i(e_m) = \alpha \in \Pi^i$ et $\forall X \subseteq E^i$, $C^i(X) = \bigcup_{x_m \in X} C^i(e_m) \subseteq \Pi^i$

$\forall W \subseteq \Pi^i$, $\bar{C}^i(W) = \bigcup_{\alpha \in W} E^i_\alpha \subseteq E^i$.

Les définitions correspondantes sont données pour $C^\ell(Y)$, $\forall Y \subseteq E^\ell$, et $\bar{C}^\ell(Z)$, $\forall Z \subseteq \Pi^\ell$.

Nous noterons $\mathfrak{X}$ pour $\mathfrak{X}^{i\ell}$ lorsqu'il n'y a pas d'ambiguité à propos des fichiers concernés ; d'après la définition de $\mathfrak{X}$ et la définition (f), on peut donner les expressions ci-dessous des coupes de $\mathfrak{X}$ dans E^ℓ et de $\bar{\mathfrak{X}}$ dans E^i :

(h) $\forall e_m \in E^i_\alpha$, $\mathfrak{X}_m = \mathfrak{X}_{jk,m} \cap \ldots \cap \mathfrak{X}_{pq,m} = \bigcup_{\beta \in R_\alpha} E^\ell_\beta = E_{R_\alpha} \subseteq E^\ell$

$\forall e_m \in E^\ell_\beta$, $\bar{\mathfrak{X}}_m = \bar{\mathfrak{X}}_{jk,m} \cap \ldots \cap \bar{\mathfrak{X}}_{pq,m} = \bigcup_{\alpha \in \bar{R}_\beta} E^i_\alpha = E_{\bar{R}_\beta} \subseteq E^i$.

Enfin, la notion de coupe d'une relation $R \subseteq E \times F$ peut s'étendre à des sous-ensembles de E et F ; nous notons :

$$(i)\ \forall X \subseteq E,\ X^* = \bigcup_{x_m \in X} R_m \subseteq F$$

$$\forall Y \subseteq F,\ Y^\dagger = \bigcup_{y_m \in Y} \bar{R}_m \subseteq E$$

Cette notation pourra être appliquée à tous les types ou relations définies jusqu'à présent.

k) Test de productivité dans une relation

Etant donné $R \subseteq E \times F$, considérons le prédicat $rec(x_m,x_n)$ défini sur $E \times E$, et le prédicat $\overline{rec}(y_m,y_n)$ défini sur $F \times F$:

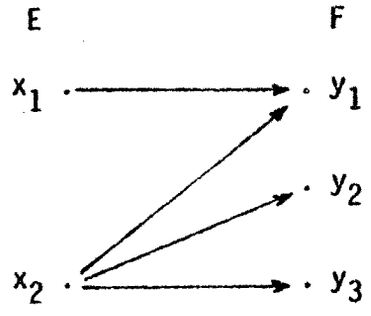
$$(a)\ \forall x_m,x_n \in E,\ rec(x_m,x_n) \iff R_m \subseteq R_n$$

$$\forall y_m,y_n \in F,\ \overline{rec}(y_m,y_n) \iff \bar{R}_m \subseteq \bar{R}_n.$$

Lorsque le prédicat $rec(x_m,x_n)$ est vrai, tous les éléments de la coupe de x_m sont donnés par celle de x_n ; on dit que x_m n'est pas productif par rapport à x_n. Quand ce prédicat est faux, la coupe de x_m n'est pas donnée par celle de x_n et on dit que x_m est productif par rapport à x_n.

Exemple :

Considérons la relation $R \subseteq E \times F$ donnée par le graphe ci-dessous :



- On a : $R_1 = \{y_1\}$; $R_2 = \{y_1,y_2,y_3\}$

$\bar{R}_1 = \{x_1,x_2\}$; $\bar{R}_2 = \bar{R}_3 = \{x_2\}$

d'où par exemple :

$rec(x_1,x_2)$ est vrai car $R_1 \subset R_2$

$rec(x_2,x_1)$ est faux car $R_2 \supset R_1$

$\overline{rec}(y_1,y_2)$ est faux car $\bar{R}_1 \supset \bar{R}_2$

$rec(y_2,y_3)$ est vrai car $\bar{R}_2 = \bar{R}_3$

+++

La définition (a) de ces prédicats s'étend aux relations étudiées jusqu'ici avec les notations suivantes :

(b) Pour toute relation $R_{jk} \subseteq V_j \times V_k$, on les note Rec_{jk} et $\bar{R}ec_{jk}$.
Pour toute relation correspondante $\mathcal{R}_{jk}$, on les note REC_{jk} et $\overline{REC}_{jk}$
Pour toute relation $R^{i\ell} \subseteq \Pi^i \times \Pi^\ell$, on les note $\overset{i\ell}{Rec}$ et $\overset{i\ell}{\bar{R}ec}$, ou plus simplement Rec et $\bar{R}ec$ s'il n'y a pas d'ambiguité à propos des fichiers concernés.
De même, on note REC et $\overline{REC}$ les prédicats correspondant à la relation de cohérence $\mathcal{R} \subseteq E^i \times E^\ell$ correspondante.

Revenons à la définition générale (a) de ces prédicats ; on démontre facilement que rec et $\overline{rec}$ définissent respectivement un préordre noté ρ sur $E \times E$ et un préordre noté $\bar{\rho}$ sur $F \times F$:

$$\forall x_m, x_n \in E,\ x_m \rho x_n \iff rec(x_m, x_n)$$
$$\forall y_m, y_n \in F,\ y_m \bar{\rho} y_n \iff \overline{rec}(y_m, y_n).$$

La réflexivité et la transitivité de ρ et $\bar{\rho}$ sont évidentes.

Considérons les relations notées γ et $\bar{\gamma}$ ainsi définies :

(c) $\forall x_m, x_n \in E,\ x_m \gamma x_n \iff R_m = R_n$
$\forall y_m, y_n \in F,\ y_m \bar{\gamma} y_n \iff \bar{R}_m = \bar{R}_n$

Ce sont évidemment des relations d'équivalence sur E et F respectivement. Nous notons E/γ et $F/\bar{\gamma}$ les ensembles quotients correspondants. Nous noterons également souvent par (x_m) toute classe de E/γ de représentant x_m, (y_n) toute classe de $F/\bar{\gamma}$ de représentant y_n.
Si on considère la restriction de ρ (resp. $\bar{\rho}$) aux éléments de γ (resp. $\bar{\gamma}$), l'antisymétrie de γ (resp. $\bar{\gamma}$) est alors évidente et on a :

Propriété 2 - Ordre large :

Les prédicats rec et $\overline{rec}$ induisent une relation d'ordre large notée $\preceq$ sur E/γ et $F/\bar{\gamma}$ respectivement. Ces ensembles sont partiellement ordonnée par $\preceq$.

Etant donné $R \subseteq E \times F$, nous disons que R possède un <u>élément universel</u> $x_m \in E$ si et seulement si :

$$\forall y_n \in F,\ x_m\ R\ y_n$$

De même, $y_m \in F$ est élément universel de $\bar{R}$ si et seulement si :

$$\forall x_n \in E,\ y_m\ \bar{R}\ x_n.$$

<u>Propriété 3 - Sup-demi-treillis</u> :

Une condition nécessaire et suffisante pour que l'ensemble E/γ (resp. $F/\bar{\gamma}$) soit un sup-demi-treillis est que R (resp. $\bar{R}$) possède au moins un élément universel.

<u>Démonstration</u> : La condition suffisante est évidente, on démontre la condition nécessaire en montrant que si R (resp. $\bar{R}$) n'admet aucun élément universel, tous les couples d'éléments de E/γ (resp. $F/\bar{\gamma}$) ne sont pas majorés.

+++

La propriété 2 et la propriété 3 sont transposées avec les notations suivantes aux divers prédicats définis par (b) :

(d) On note $\underset{\sim}{\prec}_{jk}$ l'ordre défini sur V_j/γ_{jk} et $V_k/\bar{\gamma}_{jk}$ par les prédicats Rec_{jk} et $\bar{R}ec_{jk}$.

On note $\leq_{jk}$ l'ordre défini sur E^i/γ'_{jk} et $E^\ell/\bar{\gamma}'_{jk}$ par les prédicats REC_{jk} et $\overline{REC}_{jk}$, où γ'_{jk} et $\bar{\gamma}'_{jk}$ sont les notations de γ et $\bar{\gamma}$ pour la définition c) relative aux coupes de $\mathcal{H}_{jk}$.

On note $\underset{\sim}{\prec}^{i\ell}$ l'ordre défini sur $\Pi^i/\gamma^{i\ell}$ et $\Pi^\ell/\bar{\gamma}^{i\ell}$ par les prédicats $Rec^{i\ell}$ et $\bar{R}ec^{i\ell}$ (on note souvent plus succinctement $\underset{\sim}{\prec}$ et Π^i/γ, $\Pi^\ell/\bar{\gamma}$).

On note $\leq^{i\ell}$ l'ordre défini sur $E^i/\gamma'^{,i\ell}$ et $E^\ell/\bar{\gamma}'^{,i\ell}$ par les prédicats $REC^{i\ell}$ et $REC^{i\ell}$, ou plus succinctement $\leq$ et E^i/γ', $E^\ell/\bar{\gamma}'$.

La définition (c) des relations γ et $\bar{\gamma}$, ainsi que la propriété 1 de 2.1.3.j entraînent la propriété suivante relative aux partitions E^i/C^i et E^i/γ', E^ℓ/C^ℓ et $E^\ell/\bar{\gamma}'$:

Propriété 4 - Partitions hiérarchisées :

$$\text{On a : } \forall (e_m) \in E^i/C^i,\ \exists (e_p) \text{ unique} \in E^i/\gamma' \mid (e_m) \subseteq (e_p)$$

$$\forall (e_m) \in E^\ell/C^\ell,\ \exists (e_p) \text{ unique} \in E^\ell/\bar{\gamma}' \mid (e_m) \subseteq (e_p)$$

Démonstration : D'après la propriété 1, on a :
$\forall \alpha \in \Pi^i$, $\forall e_m, e_n \in E^i_\alpha$, $\mathcal{P}_m = \mathcal{P}_n \Rightarrow e_m\ \gamma'\ e_m$ d'après la définition générale (c) de γ. Si $e_p \in E^i$ est le représentant de la classe de E^i/γ' contenant e_m et e_n, on a : $E^i_\alpha \subseteq (e_p)$. γ' étant une relation d'équivalence, on montre facilement que (e_p) est unique.

+++

Reprenons la définition donnée en 2.1.3.j des éléments de Π^i :

$$\alpha_m = (v_{j,r},\ldots,v_{p,s}) \text{ et } \alpha_n = (v_{j,t},\ldots,v_{p,u}).$$

On démontre que :

$$\text{(e) } \forall \alpha_m,\alpha_n \in \Pi^i,\ REC(\alpha_m,\alpha_n) \Longleftrightarrow Rec_{jk}(v_{j,r},v_{j,t}) \text{ et } \ldots \text{ et } Rec_{pq}(v_{p,s},v_{p,u}).$$

Montrons la condition suffisante :

Par définition, $REC(\alpha_m,\alpha_n) \Longleftrightarrow R_{\alpha_m} \subseteq R_{\alpha_n}$

D'après la définition (e) de 2.1.3.j, on a donc :

$$R_{\alpha_m} = R_{jk,r} \times \ldots \times R_{pq,s} \subseteq R_{jk,t} \times \ldots \times R_{pq,u} = R_{\alpha_n}$$

et ceci ne peut être vérifié que si :

$$R_{jk,r} \subseteq R_{jk,t},\ \ldots,\ R_{pq,s} \subseteq R_{pq,u}$$

donc $Rec_{jk}(v_{j,r},v_{j,t})$ et ... et $Rec_{jk}(v_{p,s},v_{p,u})$.

La condition nécessaire est démontrée de manière analogue.

+++

La propriété correspondante est également vérifiée pour tout couple d'éléments de Π^{ℓ}.

On a alors le corollaire suivant :

Propriété 5 - Productivité :

Une condition nécessaire et suffisante pour que $\alpha_m \in \Pi^i$ (resp. $\beta_m \in \Pi^{\ell}$) soit productif par rapport à α_n (resp. β_n), est qu'il existe au moins une composante de α_m (resp. β_m) productive par rapport à la composante correspondante de α_n (resp. β_n).

On démontre également la propriété suivante :

Propriété 6 - Sup-demi-treillis :

Une condition nécessaire et suffisante pour que les ensembles Π^i/γ et $\Pi^{\ell}/\bar{\gamma}$ soient des sup-demi-treillis est que toutes les relations externes appartenant à la définition de R possèdent au moins un élément universel.

Démonstration : Elle découle directement de la propriété 3, en remarquant que Π^i/γ et $\Pi^{\ell}/\bar{\gamma}$ sont alors des produits directs de sup-demi-treillis.

+++

Propriété 7 :

$$\forall \alpha_r, \alpha_s \in \Pi^i,\ \mathrm{Rec}(\alpha_r, \alpha_s) \Rightarrow \mathrm{REC}(e_m, e_n),\ \forall e_m \in E^i_{\alpha_r},\ \forall e_n \in E^i_{\alpha_s}$$

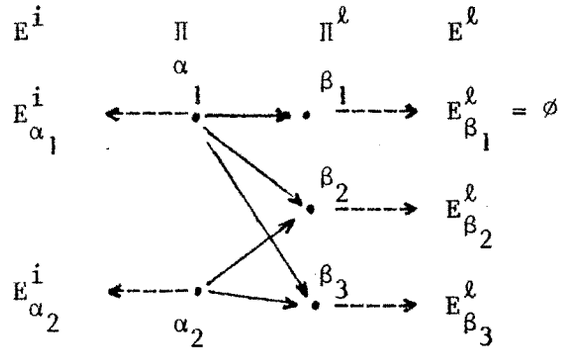
$$\forall \beta_r, \beta_s \in \Pi^{\ell},\ \bar{\mathrm{R}}\mathrm{ec}(\beta_r, \beta_s) \Rightarrow \overline{\mathrm{REC}}(e_m, e_n),\ \forall e_m \in E^{\ell}_{\beta_r},\ \forall e_n \in E^{\ell}_{\beta_s}$$

La réciproque n'est généralement pas vraie.

Démonstration :

La propriété découle de la définition des prédicats et de la propriété 4. La réciproque n'est pas toujours vérifiée, c'est le cas notamment lorsqu'il existe une interdépendance entre les partitions sur E^i et E^{ℓ} liées aux différentes caractéristiques inverses. Cette interdépendance peut se traduire par le fait que certains E^i_{α}, E^{ℓ}_{β} sont vides. Considérons par

exemple la figure ci-dessous qui représente une relation $R \subset \Pi^i \times \Pi^\ell$ et des éléments correspondants des partitions E^i/C^i et E^ℓ/C^ℓ :



On a :

$R_{\alpha_1} = \{\beta_1, \beta_2, \beta_3\}$

$R_{\alpha_2} = \{\beta_1, \beta_2\}$

et $Rec(\alpha_1, \alpha_2)$ est <u>faux</u>

On a :

$\forall e_m \in E^i_{\alpha_1}$, $\gamma_m = E^\ell_{\beta_1} \cup E^\ell_{\beta_2} \cup E^\ell_{\beta_3} = E^\ell_{\beta_2} \cup E^\ell_{\beta_3}$ car $E^\ell_{\beta_1} = \emptyset$

et $\forall e_n \in E^i_{\beta_2}$, $\gamma_n = E^\ell_{\beta_2} \cup E^\ell_{\beta_3} = \gamma_m$.

Donc $REC(e_m, e_n)$ est <u>vrai</u> alors que $Rec(\alpha_1, \alpha_2)$ est <u>faux</u>.

+++

Nous verrons ultérieurement (cf. 2.5) l'utilité de cette notion de productivité pour optimiser la construction des classes de cohérence en minimisant le nombre de constructions de coupes des relations (donc en diminuant le nombre d'accès aux différents fichiers).

2.1.4. <u>Conclusion</u>

Ces éléments de définition étant posés, nous pouvons définir comme suit le problème du rapprochement des informations :
étant donné N fichiers $E^1,\ldots,E^N$ produits respectivement par N saisies $S^1,\ldots,S^N$ à partir d'ensembles de départ $A^1,\ldots,A^N$ a priori non disjoints, le problème du rapprochement de ces informations est celui de la construction sur $E = E^1+\ldots+E^N$ de la partition H par la relation d'homologie entre enregistrements.

Au vu des informations disponibles, nous ne pouvons que construire la partition de E en classes de cohérence ; il nous faut donc établir dans quelles conditions ces deux partitions sont identiques, puis envisager une solution pour le cas général où ces conditions ne sont pas respectées.

CHAPITRE 2.2. Conditions d'existence d'une solution unique

2.2. Conditions d'existence d'une solution unique

2.2.1. Ambiguïté inhérente au problème

Etant donné deux fichiers E^i et E^ℓ, et abstraction faite pour l'instant de toute possibilité d'erreurs sur les informations, construire la partition de $E = E^i + E^\ell$ par la relation d'homologie consiste à déterminer un processus qui, pour tout $e_m \in E^i$, tout $e_n \in E^\ell$ permette de décider si $e_m \sim e_n$.

Les seuls éléments logiques à notre disposition sont les relations externes pouvant exister entre les éléments de D^i et D^ℓ. Remarquons que, de ce point de vue, les relations internes aux éléments de D^i et D^ℓ ne peuvent être utiles que si elles peuvent être composées avec des relations externes ; le résultat de cette composition est alors une nouvelle relation externe utilisable.

D'après les définitions données en 2.1.3.e des relations externes, et celle donnée en 2.1.3.g de la relation de cohérence totale, nous pouvons écrire :

(1) $\forall e_m \in E^i, \forall e_n \in E^\ell, e_m \sim e_n \Rightarrow e_m \, \chi^{i\ell} \, e_n$.

La réciproque n'étant généralement pas vraie, la cohérence totale n'apparaît donc que comme une condition nécessaire à l'homologie, et il nous faut rechercher les caractéristiques d'une condition suffisante, c'est-à-dire les propriétés que doit satisfaire $\chi^{i\ell}$ pour que :

(2) $\forall e_m \in E^i, \forall e_n \in E^\ell, e_m \, \chi^{i\ell} \, e_n \Rightarrow e_m \sim e_n$

ou :

(3) $\forall e_m \in E^i, \forall e_n \in E^\ell, e_m \, \chi_{jk} \, e_n$ et ... et $e_m \, \chi_{pq} \, e_n \Rightarrow e_m \sim e_n$.

Ces propriétés constituent les conditions d'existence d'une solution au problème posé.

2.2.2. Conditions d'existence d'une solution unique

Pour simplifier les notations, nous considérons ci-dessous les conditions que doivent vérifier deux caractéristiques K_j et K_k définies respectivement sur A^1 et A^2, une relation externe R_{jk} entre ces deux caractéristiques, pour que la relation (3) soit vérifiée :

$$\forall e_m \in E^1,\ \forall e_n \in E^2,\ e_m \not\sim_{jk} e_n \Rightarrow e_m \sim e_n.$$

Cette relation est équivalente à la suivante :

$$(4)\ \forall a_m \in A^1,\ \forall a_n \in A^2,\ K_j(a_m)\ R_{jk}\ K_k(a_n) \Rightarrow a_m = a_n.$$

Propriété 1 : Une condition nécessaire à l'existence de (4) est que les caractéristiques K_j et K_k soient discriminantes pour leurs ensembles de définition respectifs.

Démonstration : Supposons que K_j ne soit pas discriminante pour A^1 ; on a d'après 2.1.2.k :

$$\exists a_m \in A^1,\ \exists a_n \in A^2,\ a_m \neq a_n \mid K_j(a_m) = K_j(a_n) = v_j \in V_j.$$

Supposons que $a_n \in A^1 \cap A^2$ et que $K_k(a_n) = v_k \in V_k$.

On a donc $v_j\ R_{jk}\ v_k$ (par définition d'une relation externe) et :

$$K_j(a_m)\ R_{jk}\ K_k(a_n), \text{ avec } a_m \neq a_n.$$

+++

Propriété 2 : Si nous considérons la propriété 1 ci-dessus vérifiée, une autre condition nécessaire à l'existence de (4) est que R_{jk} soit une relation biunivoque dans $V_j \times V_k$.

Démonstration : Rappelons qu'une relation biunivoque R dans A × B est telle qu'on puisse lui associer une bijection entre A et B, soit :

$\forall a \in A,\ \exists b$ unique $\mid$ a R b, et réciproquement.

- Supposons que $\mathbb{R}_{jk}$ ne soit pas biunivoque et par conséquent :

$$\exists v_{j,p},\ v_{j,q} \in V_j,\ \exists v_{k,\ell} \in V_k \mid v_{j,p}\ R_{jk}\ v_{k,\ell} \text{ et } v_{j,q}\ R_{jk}\ v_{k,\ell}$$

- K_j étant discriminante sur A^1 par hypothèse, on a :

$$\exists a_m, a_n \in A^1,\ a_m \neq a_n \mid K_j(a_m) = v_{j,p},\ K_j(a_n) = v_{j,q}$$

- Supposons que $a_n \in A^1 \cap A^2$, et $K_k(a_n) = v_{k,\ell}$, on a alors :

$$K_j(a_m)\ R_{jk}\ K_k(a_n) \text{ avec } a_m \neq a_n$$

+++

Supposons vérifiées les propositions 1 et 2 pour K_j, K_k, R_{jk}. Notons R_{jk} la fonction associée à R_{jk} (relation fonctionnelle biunivoque), $W_j \subseteq V_j$ son domaine de définition, $W_k \subseteq V_k$ son ensemble de valeurs. Soit $B_j \subseteq A^1 = \bigcup_{v_{j,p} \in W_j} A_{j,p} = \bar{K}_j(W_j)$.

Considérons l'application $\rho = R_{jk} \circ K_j$ de B_j sur W_k ; cette application peut être considérée comme une extension de K_k à $B_j \subseteq A^1$. De la même façon, on peut définir $\rho = \bar{R}_{jk} \circ K_k$ comme l'extension de K_j à $B_k \subseteq A^2$, avec $B_k = \bar{K}_k(W_k)$.

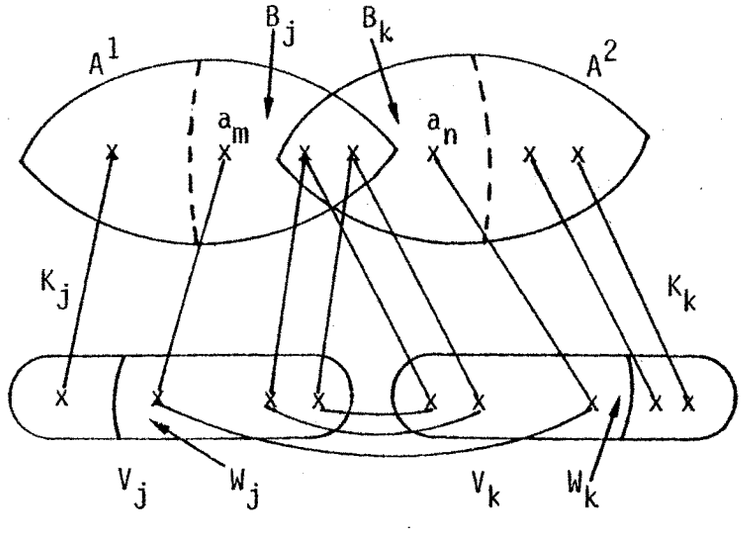


Figure 1

Propriété 3 : Une condition nécessaire à l'existence de la relation (4) est que l'extension ρ de K_k à B_j et l'extension ρ' de K_j à B_k conservent à ces deux caractéristiques leur discriminance sur $A^2 \cup B_j$ et $A^1 \cup B_k$ respectivement.

Démonstration :

- On suppose que K_j, K_k, R_{jk} satisfont aux conditions énoncées dans les propriétés 1 et 2.

- Supposons que l'extension de K_k à B_j par l'application ρ ne soit pas discriminante sur $A^2 \cup B_j$:

$$\exists a_m \in B_j, \exists a_n \in A^2, a_m \neq a_n \mid R_{jk}(K_j(a_m)) = K_k(a_n).$$

On a donc :

$$K_j(a_n) \; R_{jk} \; K_k(a_n), \; a_m \neq a_n.$$

(Voir l'illustration de ce cas dans la figure 1 ci-dessus).

- La démonstration est identique pour le cas symétrique de l'extension de K_j à B_k par ρ'.

+++

Propriété 4 : Une condition nécessaire et suffisante pour que la relation (4) soit vérifiée est que K_j, K_k, R_{jk} vérifient les conditions énoncées ci-dessus dans les propriétés 1, 2, et 3.

Démonstration :

La condition nécessaire a été démontrée ; considérons la condition suffisante.

Par hypothèse, K_j, K_k, R_{jk} vérifient les conditions énoncées dans les propriétés 1, 2, 3.

D'après les notations précédentes (résumées par la Figure 1), on a :

$$\forall a_m \in A^1, \forall a_n \in A^2, K_j(a_m) \; R_{jk} \; K_k(a_n) \Rightarrow a_m \in B_j \subseteq A^1 \text{ et } a_n \in B_k \subseteq A^2.$$

D'après les conditions énoncées dans les propriétés 1 et 2, on a :

$$\forall a_m \in B_j, \exists a_n \text{ unique} \in B_k \mid K_j(a_m) \; R_{jk} \; K_k(a_n) \text{ et réciproquement.}$$

Considérons $K_k(a_n) = v_{k,p} \in V_k$; on a :

$$\rho(a_m) = R_{jk}(K_j(a_m)) = K_k(a_n) = v_{k,p}.$$

Si la condition posée dans la propriété 3 est vérifiée, l'extension de K_k à $A^2 \cup B_j$ par l'application ρ est discriminante ; on a alors $a_m = a_n$ et $B_j = B_k \subseteq A^1 \cap A^2$.

Par définition d'une relation externe, on a : $\forall a_n \in A^1 \cap A^2$, $K_j(a_m)$ R_{jk} $K_k(a_n)$ et par conséquent $A^1 \cap A^2 \subseteq B_j = B_k$, soit : $A^1 \cap A^2 = B_j = B_k$.

+++

Il est possible de généraliser les propriétés ci-dessus à des ensembles de caractéristiques et de relations entre ces caractéristiques ; les démonstrations correspondantes sont du même type que celles présentées ci-dessus. Nous ne les faisons donc pas figurer ici.

Considérons deux ensembles A^1, A^2, des saisies S^1 et S^2 de ces ensembles relativement aux descriptions D^1 et D^2 donnant les fichiers E^1 et E^2. Considérons l'ensemble des relations externes $R_{jk} \ldots R_{pq}$ existant entre les éléments de D^1 et D^2 et la relation (3) énoncée plus haut :

(3) $\forall e_m \in E^1$, $\forall e_n \in E^2$, $e_m \mathcal{R}_{jk} e_n$ et ... et $e_m \mathcal{R}_{pq} e_n \Rightarrow e_m \sim e_n$.

Propriété 5 : Une condition nécessaire à l'existence de la relation (3) est que D^1 et D^2 comprennent chacune au moins un ensemble discriminant d^1 et d^2 de caractéristiques.

Propriété 6 : Une condition nécessaire à l'existence de la relation (3) est que, la propriété 5 étant satisfaite, les relations externes liant les éléments de d^1 et d^2 déterminent une relation R fonctionnelle biunivoque entre π^1 et π^2 respectivement ensembles produits des ensembles de valeurs des caractéristiques constituant d^1 et d^2.

Propriété 7 : Une condition nécessaire à l'existence de la relation (3) est que, la propriété 6 étant satisfaite, l'extension $\rho = R \circ d^1$ de d^2 dans A^1, l'extension $\rho' = \bar{R} \circ d^2$ de d^1 dans A^2 conservent à d^1 et d^2 leur discriminance.

Propriété 8 : Une condition nécessaire et suffisante pour que la relation (3) soit vérifiée est que le système d'informations constitué par D^1, D^2, l'ensemble des relations externes $R_{jk}, \ldots, R_{pq}$, vérifie les conditions énoncées dans les propriétés 5, 6, 7 di-dessus.

2.2.3. Caractéristiques - relations utiles

Une conséquence de la propriété 8 est que, étant donné la relation de cohérence totale entre enregistrements définie sur $E^1 \times E^2$, on peut éventuellement restreindre sa définition aux seuls éléments constituant un système d'information vérifiant cette propriété (caractéristiques, relations utiles). Les autres informations sont dites redondantes pour le rapprochement de E^1 et E^2.

Exemple :

Soit D^1 = {NO-SS, NOM, AGE-MARIAGE} = $\{K_1^1, K_2^1, K_3^1\}$ définie sur A^1

D^2 = {NO-SS, NOM, AGE-DECES} = $\{K_1^2, K_2^2, K_4^2\}$ définie sur A^2.

La caractéristique Numéro de Sécurité Sociale (NO-SS) est discriminante sur A et invariante ; la relation externe correspondante est l'égalité qui est une relation fonctionnelle biunivoque. La relation de cohérence totale sur $E^1 \times E^2$ est donnée par :

$$\forall e_n \in E^1, \forall e_n \in E^2, e_m \not\sim e_n \iff e_m \not\sim_{11} e_n \text{ et } e_m \not\sim_{22} e_n \text{ et } e_m \not\sim_{34} e_n$$

où R_{11} et R_{22} sont des relations d'égalité, R_{34} la relation $\leq$.

Le système d'informations vérifie la propriété 8, avec $d^1 = \{K_1^1\}$, $d^2 = \{K_1^2\}$, R = R_{11} et on a donc :

$$\forall e_m \in E^1, \forall e_n \in E^2, e_m \not\sim e_n \Rightarrow e_m \sim e_n.$$

Pour le couplage de E^1 et E^2, on peut cependant limiter l'évaluation de $\not\sim$ à la relation $\not\sim_{11}$; K_2^1, K_3^1, K_2^2, K_4^2, R_{22}, R_{34} sont alors redondantes pour ce traitement.

+++

Une optimisation du processus de rapprochement consistera donc à rechercher le plus petit ensemble de caractéristiques et de relations vérifiant la propriété 8 dans le système d'informations disponibles.

2.2.4. Conclusion

Les conditions théoriques d'existence d'une solution unique au problème posé sont très restrictives ; dans le cas général qui nous intéresse ici où il n'y a pas de caractéristique ni de relation satisfaisant de manière évidente ces conditions, il convient de procéder à une analyse détaillée du système d'informations. Trois types de situations peuvent logiquement se présenter :

a) Existence de solutions multiples :

Lorsque la relation de cohérence totale ne satisfait pas les conditions de la propriété 8, toute classe de cohérence construite à partir de cette relation peut contenir plusieurs classes de H.

Une solution au problème est alors toute partition de $\bar{E}$ satisfaisant la relation de cohérence totale, mais un tel choix ne peut être qu'arbitraire compte tenu des informations disponibles. L'ambiguïté ne peut être levée que par l'apport d'information supplémentaire ; ceci peut s'envisager de deux façons :

- saisie complémentaire d'informations permettant de résoudre les cas ambigus en rapprochant l'expression de la relation de cohérence des conditions d'obtention d'une solution unique. Comme précisé dans l'introduction, ceci est le plus souvent matériellement impossible. Toutefois, le recours à des sources d'information annexes (non enregistrées) peut aider à résoudre certains cas d'ambiguïté ;
- établissement d'un processus d'évaluation du degré de vraisemblance d'une solution par rapport aux autres. Ceci nécessite une étude approfondie des informations disponibles et la définition, pour chaque relation utile, d'une mesure de vraisemblance.

Dans tous les cas, on recherche un processus de décision homogène garantissant à la solution trouvée un maximum de cohérence.

b) Existence d'une solution unique :

Le système d'informations satisfait aux conditions d'existence d'une solution unique ; il y a équivalence entre cohérence et homologie des enregistrements. La solution peut être construite avec le minimum de difficultés puisqu'il suffit de construire les classes de cohérence.

c) Aucune solution possible :

Ce cas limite se produit lorsqu'il n'existe aucune relation externe entre les différentes descriptions, soit lorsque l'on constate que toutes les classes de cohérence se réduisent à un seul élément, ce qui signifie que les ensembles de départ sont totalement disjoints.

L'analyse préalable des informations, sauf cas évident, doit donc porter sur les points suivants :

- situer le système d'informations par rapport aux trois cas logiques présentés ci-dessus. Analyse des propriétés des relations, recherche des ensembles discriminants de caractéristiques ;
- le cas échéant, optimiser le processus de couplage par la sélection des caractéristiques et relations utiles.

Nous reviendrons plus en détails sur ces points au chapitre 2.4. Dans l'étude qui précède, il a été fait abstraction des erreurs pouvant affecter le système d'informations ; il importe, dans un second temps, d'en donner une caractérisation aussi précise que possible et d'en analyser les conséquences par rapport aux conditions idéales énoncées jusqu'ici.

CHAPITRE 2.3. - Caractérisation des erreurs

2.3. Caractérisation des erreurs

Etant donné un ensemble A, une description D des éléments de A, une saisie S de cet ensemble, on peut résumer aux trois catégories suivantes les erreurs pouvant affecter le fichier E correspondant :

- variations de valeurs pour une caractéristique,
- omissions de valeurs pour une caractéristique,
- sur-enregistrement, sous-enregistrement des éléments de A pour la saisie S.

Considérons les conséquences de ces différents types d'erreurs sur le processus de couplage des informations.

2.3.1. Les variations

Il y a variation de valeurs pour une caractéristique $K \in D$ et une saisie S produisant le fichier E quand, pour certains éléments de A, les valeurs enregistrées de la caractéristique sont différentes des valeurs réelles données par K. En d'autres termes, la caractéristique fichier C correspondant à K n'est pas égale à $\bar{S} \circ K$; pour certains éléments $a_i \in A$ et pour les enregistrements correspondants $e_i = S(a_i) \in E$, on a $K(a_i) \neq C(e_i)$. L'exemple ci-dessous illustre ce cas d'erreur et ses conséquences sur le processus de couplage.

L'incidence de ce type d'erreurs est double :

a) L'ensemble V' des valeurs enregistrées pour la caractéristique fichier C peut ne pas être identique à l'ensemble V des valeurs originelles données par la caractéristique K ; une variation peut soit produire une valeur parasite (n'appartenant pas à V), soit réutiliser abusivement une valeur de V différente de la valeur originelle. Dans le premier cas, le nombre de valeurs augmente artificiellement, dans le second cas il a tendance à diminuer ; le nombre de valeurs observées varie donc dans un sens généralement imprévisible.
De plus, les variations ont une incidence sur les fréquences observé des valeurs de la caractéristique K : elles diminuent la fréquence des valeurs correctes et augmentent celle des valeurs erronées.

b) Pour une relation de cohérence totale donnée, les variations de valeurs peuvent rendre incohérents des enregistrements homologues, et par là interduire leur couplage ; elles peuvent aussi rendre cohérents des enregistrements non homologues, créant ainsi une ambiguïté supplémentaire (les classes de cohérence sont artificiellement élargies).

Exemple :

Soit D^1 = {NOM, DATE-NAISSANCE} = $\{K_1^1, K_2^1\}$ définie sur $A^1 = \{a_1, a_2\}$

D^2 = {NOM, DATE-MARIAGE} = $\{K_1^2, K_3^2\}$ définie sur $A^2 = \{a_1, a_2, a_3\}$

avec : $K_1^1(a_1) = K_1^2(a_1)$ = 'DUPONT'

$K_1^1(a_2) = K_1^2(a_2)$ = 'DUPOND' ; $K_1^2(a_3)$ = 'LA SALLE'

$K_2^1(a_1)$ = 1920 ; $K_2^1(a_2)$ = 1900

$K_3^2(a_1)$ = 1940 ; $K_3^2(a_2)$ = 1920 ; $K_3^2(a_3)$ = 1930.

Soit une saisie S^1 de A^1 relativement à D^1 donnant le fichier :

E^1 = {('DUPONT',1920), ('DUPOND',1900)} = $\{S^1(a_1), S^1(a_2)\} = \{e_1, e_2\}$.

Soit une saisie S^2 de A^2 relativement à D^2 donnant le fichier :

E^2 = {('DUPOND',1940), ('DUPOND',1920), 'LASALLE',1930)} =

$\{S^2(a_1), S^2(a_2), S^2(a_3)\} = \{e_3, e_4, e_5\}$.

S^2 s'accompagne de variations sur K_1^2 pour a_1 (valeur 'DUPOND' enregistrée au lieu de 'DUPONT' = $K_1^2(a_1)$) et pour a_3 (valeur enregistrée 'LASALLE' au lieu de 'LA SALLE' = $K_1^2(a_3)$).

L'ensemble $V^1 = V_1^1 \cup V_1^2$ = {'DUPONT', 'DUPOND', 'LA SALLE'} des valeurs de K_1^1 et K_1^2 est différent de l'ensemble V'^1 des valeurs enregistrées pour les caractéristiques fichier correspondantes C_1^1 et C_2^1 :

V'^1 = {'DUPONT', 'DUPOND', 'LASALLE'}.

Considérons la relation de cohérence totale avec E^1 et E^2 ainsi définie :

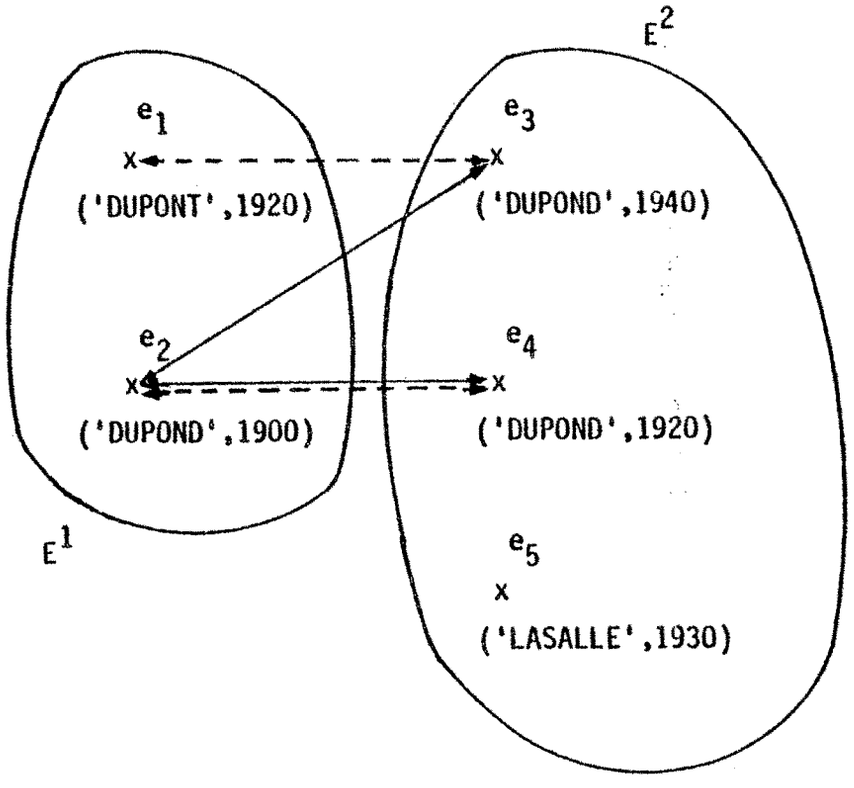
$$\forall e_m \in E^1, \forall e_n \in E^2, \; e_m \mathcal{R} e_n \iff e_m \mathcal{R}_{11} e_n \text{ et } e_m \mathcal{R}_{23} e_n$$

avec :

$e_m \mathcal{R}_{11} e_n \iff C_1^1(e_m) = C_1^2(e_n)$

$e_m \mathcal{R} e_n \iff 80 \geq C_3^2(e_n) - C_2^1(e_m) \geq 15.$

La figure ci-dessous représente en pointillés la relation d'homologie (la réalité à reconstituer), et en traits pleins la relation de cohérence totale entre enregistrements (l'apparence).



Les classes de cohérence constituées ne recouvrent pas les classes de la relation d'homologie. En raison de la variation de nom sur e_3, les enregistrements e_1 et e_3 sont incohérents bien qu'homologues, et leur couplage est impossible. La même erreur rend cohérents les enregistrements e_2 et e_3 qui ne sont pas homologues ; une ambiguïté est créée dans la classe de cohérence $\{e_2, e_3, e_4\}$.

+++

2.3.2. Les omissions

Les omissions constituent un cas particulier de variations ; la valeur erronée enregistrée est égale à la valeur indéterminée 'u' (cf. 2.1.2.c). Nous considérons néanmoins ce cas séparément en raison de ses conséquences particulières sur le processus de couplage.

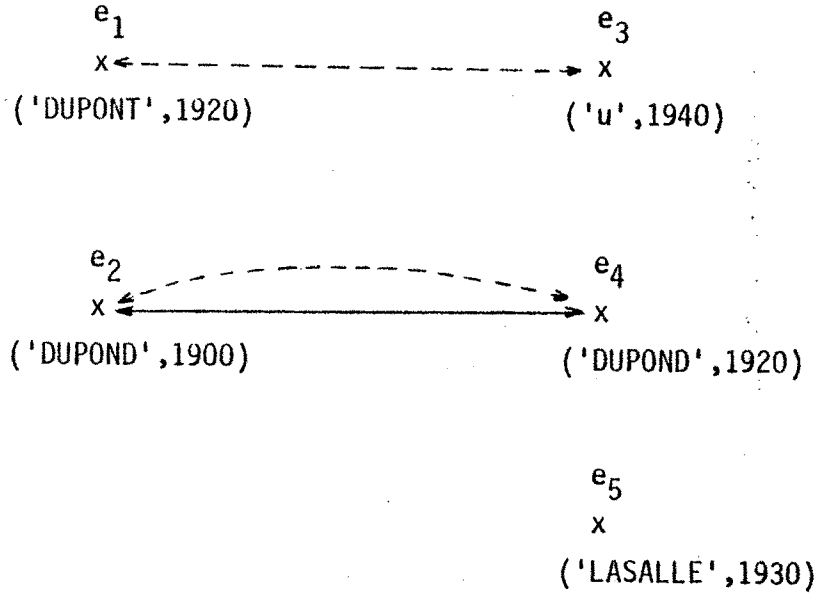
a) Comme pour les variations, l'ensemble des valeurs de la caractéristique est affecté ; on sait cependant, dans ce cas, que le nombre des valeurs de V' est inférieur à celui de V. Les fréquences des valeurs correctes sont diminuées, comme précédemment.

b) Lorsqu'une valeur de caractéristique est indéterminée, toute relation externe ayant pour argument cette valeur, devient non évaluable ; il s'ensuit que la relation de cohérence totale est également non évaluable.

Exemple : Reprenons l'exemple précédent. Supposons qu'au lieu d'une variation de nom sur e_3, on ait eu une omission de cette valeur, soit :

$$e_3 = (\text{'u'}, 1940).$$

La relation externe $\mathcal{R}_{11}$ (cohérence pour l'égalité des noms enregistrés) n'est pas vérifiée pour (e_1,e_3) et (e_2,e_3) et, par voie de conséquence, la relation de cohérence totale n'est pas vérifiée pour ces deux couples d'enregistrements. La seule classe de cohérence obtenue serait (e_2,e_4), l'homologie de e_1 et e_3 ne serait pas décelée.

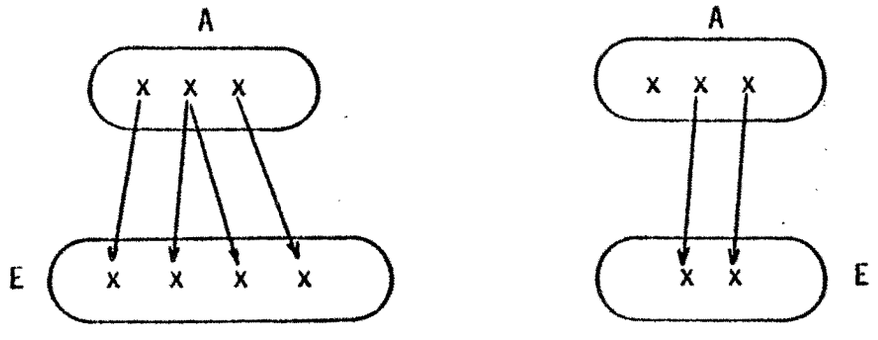


+++

2.3.3. Sur-enregistrement - Sous-enregistrement

Il y a sur-enregistrement des éléments d'un ensemble A relativement à S et D lorsque le fichier E correspondant contient plusieurs enregistrements relatifs à un même élément de A ; S n'est alors plus une bijection entre A et E.

Il y a sous-enregistrement des éléments d'un ensemble A relativement à S et D lorsque le fichier E correspondant ne contient pas l'image de tous les éléments de A ; S n'est alors qu'une bijection entre E et un sous-ensemble de A.



sur-enregistrement sous-enregistrement

Exemples :

- Sous-enregistrement lors d'un recensement de population : certaines familles n'ont pas été contactées par les agents recenseurs.
- Sur-enregistement lors d'un recensement de population : certaines familles ont été contactées plusieurs fois par divers agents recenseurs.

+++

Conséquences :

a) Le sur-enregistrement :

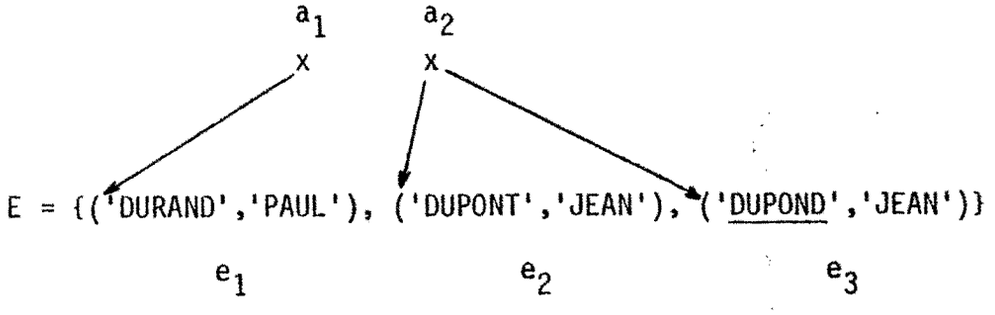
Il se traduit par une augmentation artificielle de l'effectif du fichier ; les conséquences peuvent être de deux sortes selon que la saisie redondante s'est accompagnée ou non d'erreurs.

Dans le cas où il n'y a pas d'erreur, les enregistrements objets d'une saisie multiple sont rigoureusement identiques. Les ensembles de valeurs des caractéristiques ne sont pas modifiés, seules les fréquences de

certaines valeurs sont affectées.

Dans le cas où le sous-enregistrement s'accompagne d'erreurs, on a à la fois les conséquences déjà vues et le risque de ne plus pouvoir détecter le sur-enregistrement ; on risque de considérer que l'ensemble de départ contient plus d'éléments que dans la réalité.

Exemple :



La variation sur le nom rend e_2 et e_3 différents ; lors du couplage de E avec un autre fichier, rien ne permet d'affirmer que e_2 et e_3 ne sont pas relatifs à des éléments distincts de A.

+++

b) Le sous-enregistrement :

Il se traduit par une diminution artificielle de l'effectif du fichier ; c'est certainement le type d'erreur le moins gênant en ce qui concerne le processus du couplage car il ne crée pas d'ambiguïté supplémentaire.

2.3.4. Conclusion

L'existence d'erreurs, notamment les variations et le sur-enregistrement créent une ambiguïté supplémentaire lors du processus de couplage ; même si le système d'informations répond en principe aux conditions d'existence d'une solution unique, les erreurs affectant les caractéristiques utiles replacent le problème dans le cas général d'existence de solutions multiples.

L'objectif prioritaire dans le choix des méthodes de résolution sera toujours de faire en sorte que tout couple d'enregistrements homologues appartienne à une même classe de cohérence. En d'autres termes, on modifiera la définition des relations de cohérence de façon à ce que la partition de E en classes de cohérence inclue la partition H par la relation d'homologie (partitions hiérarchisées). Ceci s'obtiendra au prix d'une ambiguïté supplémentaire qui se traduira par des classes de cohérence plus larges : différents moyens pour résoudre cette ambiguïté doivent alors être définis.

CHAPITRE 2.4. - Traitement des erreurs

2.4. Traitement des erreurs

2.4.1. Les contrôles de saisie

Au vu des conséquences que nous avons énumérées pour les différents types d'erreurs, il importe de s'assurer, dans un premier temps, que la saisie sur support informatique des informations originelles n'introduit pas de nouvelles altérations. Des contrôles systématiques sont donc à prévoir lors de la saisie ; nous nous limitons ici à en rappeler les principaux types, en mettant en évidence leurs limites.

a) Les contrôles directs

Pour toute caractéristique K, son ensemble potentiel V' de valeurs est défini par un type ; un contrôle direct des valeurs de K consiste à vérifier que l'ensemble V des valeurs saisies pour cette caractéristique est tel que $V \subseteq V'$. Un cas particulier est le contrôle direct de présence ; une information est dite à présence obligatoire si la valeur indéterminée 'u' n'appartient pas à la définition de son type ; elle est dite à présence facultative ou conditionnelle (cf. 2.4.1.b) sinon. Les limites des contrôles directs dépendent donc essentiellement de la finesse avec laquelle on peut spécifier l'ensemble des valeurs possibles de chaque caractéristique. Dans la plupart des cas, cette spécification via un type demeure assez floue, et ne permet que des contrôles assez superficiels. L'exemple des noms patronymiques est assez caractéristique de cette limitation.

D'une manière générale, ce type de contrôle ne permet pas de déceler toute variation conservant à la valeur enregistrée sa conformité au type déclaré pour la caractéristique.

b) Les contrôles indirects

A toute relation interne IR_{jk} peut être associée une procédure de contrôle indirect entre les valeurs saisies de K_j et K_k, qui consiste à vérifier que :

$$\forall e \in E,\ C_j(e)\ IR_{jk}\ C_k(e).$$

Ce type de contrôle peut naturellement être étendu à des relations internes n-aires $IR_{jk...n} \subseteq V_j \times V_k \times \ldots \times V_n$. Un cas particulier est le contrôle indirect de présence, ou contrôle de présence conditionnelle : une caractéristique K_j dont l'ensemble de valeurs V_j appartient à la définition d'une relation interne est dite à présence conditionnelle lorsque :

- la valeur indéterminée 'u' est compatible avec le type défini pour les valeurs de K_j ;
- il existe au moins un élément de la relation pour lequel K_j ne prend pas cette valeur.

Les limites des contrôles indirects sont de même nature que celles des contrôles directs ; tout dépend ici de la définition de la relation interne. Ce type de contrôle permet néanmoins un affinement des contrôles directs ; l'inconvénient est qu'il ne permet pas de localiser l'erreur sur l'une des valeurs intervenant dans l'évaluation de la relation.

Quelle que soit la finesse des contrôles prévus, il est donc clair que :

- des erreurs imputables aux données originelles ou au processus de saisie ne seront pas décelées (celles qui conservent à la valeur la conformité au type déclaré, par exemple) ;
- parmi les erreurs décelées, seules celles qui sont imputables au processus de saisie pourront être redressées (correction à partir des données originelles).

Il faut donc définir des moyens supplémentaires permettant, sinon de résoudre ces insuffisances, du moins de résoudre les conséquences précédemment évoquées pour chaque type d'erreur.

2.4.2. Traitement des variations. Première définition de la cohérence large

a) Modèles de variations

Les conséquences des variations évoquées en 2.3.2 sont traitées en substituant aux relations utiles des relations étendues fondées sur la définition de modèles de variations.

Un modèle de variations pour une caractéristique K consiste en une relation $M \subseteq V \times V$ correspondant à la définition générale suivante :

(a) $\forall v_r, v_s \in V$, $v_r \; M \; v_s \iff$ "il existe une possibilité de variation entre v_r et v_s".

Naturellement, l'évaluation de cette possibilité dépend de la nature des informations concernées, et de la connaissance que l'on peut avoir du processus réel de variation. Définir de tels modèles nécessite donc une analyse préalable détaillée des informations et du processus de transmission qui a conduit à leur obtention.

Exemple :

Dans les applications de démographie historique, le processus de transmission des informations se composait le plus souvent d'une transmission orale suivie d'une transcription qui ne pouvait être contrôlée par la personne ayant émis les informations (analphabétisme). Les variations observées dans les informations dépendent donc largement des difficultés d'interprétation de la part du transcripteur (problèmes liés à l'accent du terroir, par exemple).

Dans ces conditions, une large classe de variations peut être résolue en considérant des modèles fondés sur l'identité d'un invariant phonétique :

$\forall v_r, v_s \in V$, $v_r \; M \; v_s \iff$ "il existe un invariant phonétique commun à v_r et v_s".

Ce type de modèle a effectivement été utilisé pour résoudre les variations des noms patronymiques.

+++

b) Hypothèses relatives aux modèles

Nous ferons désormais les hypothèses suivantes pour tout modèle de variation $M \subseteq V \times V$:

- réflexivité : $\forall v_r,\ v_r\ M\ v_r$

  Cette hypothèse se justifie dans la mesure où elle correspond, dans le modèle, au cas limite de variation nulle pour toute valeur.

- symétrie : $\forall v_r, v_s \in V,\ v_r\ M\ v_s \Rightarrow v_s\ M\ v_r$

  Cette hypothèse est cohérente avec la définition générale donnée par (a). Elle présuppose qu'on ne privilégie pas un sens particulier de variation entre les valeurs, ce qui est généralement acceptable en raison du caractère aléatoire des erreurs.

c) Extension d'une relation externe par des modèles

Etant donné $R_{jk} \subseteq V_j \times V_k$, nous disons que la définition des modèles de variation M_j et M_k sur V_j et V_k respectivement induit la relation étendue $\mathcal{R}_{jk} \subseteq V_j \times V_k$ ainsi définie :

$$(b)\ \forall v_r \in V_j,\ \forall v_s \in V_k,\ v_r\ \mathcal{R}_{jk}\ v_s \Leftrightarrow \exists v_t \in V_j,\ \exists v_u \in V_k \mid v_t M_j v_r,\ v_u M_k v_s,\ v_t R_{jk} v_u.$$

Exemple : Considérons K_1 et K_2 telles que $V_1 = \{a,b,c\}$ et $V_2 = \{p,q,r\}$.

Les modèles de variation M_1 et M_2 définis par :

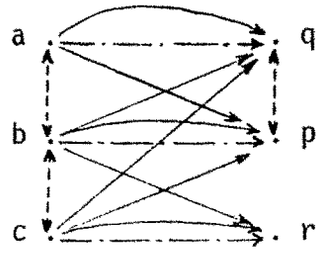
$$M_1 = \{(a,a),(b,b),(c,c),(a,b),(b,a),(b,c),(c,b)\} \subseteq V_1 \times V_1$$

$$M_2 = \{(p,p),(q,q),(r,r),(p,q),(q,p)\} \subseteq V_2 \times V_2$$

et la relation $\mathbb{R}_{12}$ définie par :

$$R_{12} = \{(b,p),(a,q),(c,r)\} \subseteq V_1 \times V_2.$$

Le graphe ci-dessous représente en pointillés les modèles de variations, en traits mixtes la relation R_{12} et en traits pleins la relation étendue $\mathcal{R}_{12}$:



On a par exemple :

$a\mathcal{R}_{12}q$ car :

aM_1b et qM_2p, avec $bR_{12}p$

etc.

+++

Un cas particulier important, cas fréquent dans la pratique, est celui où les caractéristiques considérées sont synonymes. Etant donné $R_j \subseteq V_j \times V_j$, un modèle de variations M_j pour ces caractéristiques peut être défini comme une relation $M_j \subseteq V_j \times V_j$.

Définir un modèle unique pour des caractéristiques synonymes est légitime dans la mesure où l'on suppose qu'elles présentent les mêmes potentialités de variations. Nous dirons alors que le modèle M_j est <u>général</u> pour la propriété K_j.

Dans ces conditions, la définition (b) devient :

$$(b')\ \forall v_r, v_s \in V_j,\ v_r R_j v_s \iff \exists v_t, v_u \in V_j \mid v_t M_j v_r, v_u M_j v_s, v_t R_j v_u.$$

d) <u>Modèles indépendants - Modèles complémentaires</u>

Il peut se produire que l'ensemble des variations affectant une caractéristique relève de plusieurs phénomènes pour lesquels on puisse définir des modèles distincts.

<u>Exemple</u> :

Pour les noms patronymiques, si la grande majorité des variations conservent l'invariant phonétique, certaines peuvent avoir des causes totalement différentes de celles évoquées plus haut. Par exemple, l'usage de surnoms qui viennent aléatoirement se substituer aux noms originels. Selon les applications, ce phénomène peut prendre une certaine importance et justifier la définition d'un modèle particulier. D'autres causes de variations peuvent également être envisagées, comme celles liées à la difficulté de déchiffrage des documents écrits. On peut alors envisager la définition de modèles analogues à ceux utilisés en reconnaissance des formes.

+++

Etant donné V_j sur lequel ont été définis les modèles $M_j^1, \ldots, M_j^n$, et une relation R_{jk}, on obtient par (b) les relations étendues $R_{jk}^1, \ldots, R_{jk}^n \subseteq V_j \times V_k$, qui doivent toutes être considérées dans la définition de la relation de cohérence $R \subseteq \Pi^i \times \Pi^\ell$.

Les modèles $M_j^1, \ldots, M_j^n$ sont dits <u>complémentaires</u> si la relation :

$R_{jk} = R_{jk}^1 \cap \ldots \cap R_{jk}^n$ est substituée à R_{jk} dans la définition de R.

Les modèles $M_j^1, \ldots, M_j^n$ sont dits <u>indépendants</u> si la relation :

$R_{jk} = R_{jk}^1 \cup \ldots \cup R_{jk}^n$ est substituée à R_{jk} dans la définition de R.

Les modèles indépendants sont naturellement adaptés au traitement de modalités de variations indépendantes, alors que les modèles complémentaires correspondent au traitement de phénomènes interdépendants.

e) <u>Modèles sélectifs - Modèles non sélectifs</u>

Le problème essentiel, lors de la définition d'un modèle particulier, est naturellement d'appréhender l'ensemble des variations correspondant au phénomène étudié, sous peine de ne pouvoir retrouver tous les couples d'enregistrements homologues lors du couplage. Selon les informations concernées, cela n'est pas toujours possible, et deux cas peuvent être considérés :

- les limites du modèle sont fixées aux <u>possibilités connues</u> de variation ; le modèle est dit <u>sélectif</u>. Toute relation étendue définie à partir de ce type de modèle est donc sélective car les variations non prévues dans le modèle sont rejetées lors du couplage. Ce sont ces modèles qui font l'objet de l'étude particulière qui suit ;
- aucune limite particulière ne peut être fixée pour le processus de variation qui n'est que partiellement connu. Restreindre le modèle à ces possibilités reviendrait à accepter de ne retrouver que ces variations dans le processus de couplage. Dans ce cas, une solution consiste à considérer le modèle comme une <u>relation universelle</u>, et à munir ses différents éléments d'une mesure évaluant la plausibilité de la variation correspondante (cf. 2.7).

Un tel modèle est dit non sélectif. Tout élément de l'ensemble des valeurs étant universel pour le modèle, il s'ensuit que toute relation étendue correspondante est également universelle (cf. propriété 6 ci-après), et devient non sélective dans le processus de couplage. Le choix de la meilleure solution est reporté à la phase suivante du processus général (phase de décision).

D'une manière générale, cette pondération peut être étendue aux modèles sélectifs, ce qui revient à classifier les possibilités autorisées de variations selon un critère de plausibilité. De la même façon que pour les modèles non sélectifs, cette pondération peut aider à la sélection des meilleures solutions dans la phase de décision.

f) Propriétés de la relation étendue R_{jk}

Propriété 1 - Inclusion :

Toute relation R_{jk} définie selon (b) par les modèles M_j et M_k, à partir de la relation externe R_{jk}, est une extension sur $V_j \times V_k$ de cette relation.

Soit :

$$\forall v_r \in V_j,\ \forall v_s \in V_k\ v_r R_{jk} v_s \Rightarrow v_r R_{jk} v_s.$$

La démonstration découle directement de la définition (b) et de l'hypothèse de réflexivité des modèles.

Propriété 2 - Corollaire - Modèle neutre :

Si les modèles M_j et M_k sont respectivement les relations d'égalité sur V_j et V_k, on a identité entre la relation étendue et la relation externe primitive : $R_{jk} = R_{jk}$.

La démonstration est immédiate.

Nous dirons que l'égalité sur V_j est le modèle de variations neutre pour V_j ; d'un point de vue formel, nous pourrons donc toujours considérer que tout ensemble de valeurs d'une caractéristique est muni d'un modèle qui est au moins le modèle neutre.

La propriété suivante découle directement de la propriété 1 :

Propriété 3 - Inclusion des coupes :

$$\forall v_r \in V_j,\ R_{jk,r} \subseteq \bar{R}_{jk,r}$$
$$\forall v_s \in V_k,\ \bar{R}_{jk,r} \subseteq \bar{\bar{R}}_{jk,s}.$$

Considérons à présent les conditions d'existence d'un élément universel pour la relation étendue.

Propriété 4 - Elément universel de R_{jk} :

Tout élément universel de R_{jk} (resp. $\bar{R}_{jk}$) est élément universel de toute relation étendue $\mathcal{R}_{jk}$ (resp. $\bar{\mathcal{R}}_{jk}$) déduite de cette relation.

Cette propriété est une conséquence directe de la propriété 1.

Propriété 5 - Elément universel d'un modèle :

Tout élément universel d'un modèle M_j (resp. M_k) est un élément universel pour toute relation étendue $\mathcal{R}_{jk}$ (resp. $\bar{\mathcal{R}}_{jk}$) définie à partir de ce modèle.

Démonstration :

Soit v_r un élément universel de M_j : $\forall v_s \in V_j,\ v_r M_j v_s$.

D'après l'hypothèse de réflexivité de M_k, $\forall v_t \in V_k,\ v_t M_k v_t$.

Par définition de R_{jk}, $\forall v_t \in V_k,\ \exists v_s \in V_j \mid v_s R_{jk} v_t$.
v_r étant universel pour M_j, on a aussi $v_r M_j v_s$, et donc :
$\forall v_t \in V_k,\ v_t M_k v_t$ et $\exists v_s \in V_j \mid v_s R_{jk} v_t,\ v_r M_j v_s$.
Donc, par définition même de $\mathcal{R}_{jk}$:

$\forall v_t \in V_k,\ v_r \mathcal{R}_{jk} v_t$, et v_r est élément universel de $\mathcal{R}_{jk}$.

La démonstration est analogue pour tout élément universel de M_k.

+++

Les propriétés 4 et 5 ci-dessus sont des conditions suffisantes pour que R_{jk} possède au moins un élément universel ; ce ne sont pas des conditions nécessaires. Dans le cas général, l'interdépendance possible entre les modèles et la relation externe peut induire l'existence d'éléments universels pour la relation étendue, alors que ni les modèles ni la relation externe n'en possèdent.

Exemple : Dans l'exemple précédent, R_{12}, M_1, M_2 ne possèdent aucun élément universel ; cependant, la relation étendue correspondante R_{12} (traits pleins dans le graphe) possède deux éléments universels b et $c \in V_1$. De même, la relation $\bar{R}_{12}$ possède l'élément universel $q \in V_2$, alors que $\bar{R}_{12}$ n'en possède pas.

+++

Notons Rec_{jk} et $\bar{R}ec_{jk}$ les prédicats correspondant à Rec_{jk} et $\bar{R}ec_{jk}$ pour les relations étendues R_{jk} et $\bar{R}_{jk}$:

$$(c)\ \forall v_r, v_s \in V_j,\ Rec_{jk}(v_r, v_s) \iff R_{jk,r} \subseteq R_{jk,s}$$

$$\forall v_r, v_s \in V_k,\ \bar{R}ec_{jk}(v_r, v_s) \iff \bar{R}_{jk,r} \subseteq \bar{R}_{jk,s}$$

La propriété suivante découle directement de la propriété 3 :

Propriété 6 :

Tout élément universel de R_{jk} (resp. $\bar{R}_{jk}$) est productif par rapport à tout élément non universel de R_{jk} (resp. $\bar{R}_{jk}$).

Les propriétés générales 2 et 3 de 2.1.3.k s'appliquent naturellement aux prédicats Rec_{jk} et $\bar{R}ec_{jk}$; nous notons toujours $\preceq_{jk}$ l'ordre large défini sur V_j/γ_{jk} et $V_k/\bar{\gamma}_{jk}$ (γ_{jk} et γ_{jk} sont définies à présent par rapport aux coupes de R_{jk} et $\bar{R}_{jk}$ respectivement).

En considérant les propriétés 4 et 5 ci-dessus et la propriété générale 3 de 2.1.3.k, on a :

Propriété 7 :

Une condition suffisante pour que V_j/γ_{jk} (resp. V_k/γ_{jk}) soit un sup-demi-treillis est que R_{jk} ou M_j (resp. $\bar{R}_{jk}$ ou $\bar{M}_k$) possèdent au moins un élément universel.

g) Relation étendue induite entre les enregistrements

A la relation $R_{jk} \subseteq V_j^i \times V_k^\ell$ correspond la relation notée $\mathcal{R}_{jk} \subseteq E^i \times E^\ell$ définie par :

(d) $\forall e_m \in E^i$, $\forall e_n \in E^\ell$, $e_m \mathcal{R}_{jk} e_n \iff C_j^i(e_m) R_{jk} C_k^\ell(e_n)$

et réciproquement : $\forall e_m \in E^i$, $\forall e_n \in E^\ell$, $e_m \bar{\mathcal{R}}_{jk} e_m \iff C_k^\ell(e_n), \bar{R}_{jk} C_j^i(e_m)$.

Les coupes de cette relation sont notées comme suit :

(e) $\forall e_m \in E^i$, $\mathcal{R}_{jk,m} \subseteq E^\ell$

$\forall e_m \in E^\ell$, $\bar{\mathcal{R}}_{jk,m} \subseteq E^i$

soit, d'après la définition (c) de 2.1.3.j :

$\forall e_m \in E^i$, $\mathcal{R}_{jk,m} = \bigcup_{v_s \in R_{jk,r}} E^\ell_{k,s}$, avec $C_j(e_m) = v_r \in V_j$

et

$\forall e_m \in E^\ell$, $\bar{\mathcal{R}}_{jk,m} = \bigcup_{v_s \in \bar{R}_{jk,r}} E^i_{j,s}$, avec $C_j(e_m) = v_r \in V_k$.

Les prédicats correspondant à REC_{jk} et $\overline{REC}_{jk}$ (cf. 2.1.3.k (b)) sont notés comme suit pour les relations étendues $\mathcal{R}_{jk}$ et $\bar{\mathcal{R}}_{jk}$:

(f) $\forall e_m, e_n \in E^i$, $REC_{jk}(e_m, e_n) \iff \mathcal{R}_{jk,m} \subseteq \mathcal{R}_{jk,n}$

$\forall e_m, e_n \in E^\ell$, $\overline{REC}_{jk}(e_m, e_n) \iff \bar{\mathcal{R}}_{jk,m} \subseteq \bar{\mathcal{R}}_{jk,n}$.

h) Relation étendue $\mathcal{R}$, propriétés

A une relation $R \subseteq \Pi^i \times \Pi^\ell$ (cf. 2.1.3.j) définie à partir des relations externes $R_{jk}, \ldots, R_{pq}$, correspond une relation notée $\mathcal{R} \subseteq \Pi^i \times \Pi^\ell$ après les extensions définies par (b) :

(g) $\forall \alpha \in \Pi^i$, $\forall \beta \in \Pi^\ell$, $\alpha R \beta \iff v_{j,r}\ R_{jk}\ v_{k,t}$ et ... et $v_{p,s}\ R_{pq}\ v_{q,u}$

et la relation inverse :

$\forall \alpha \in \Pi^i$, $\forall \beta \in \Pi^\ell$, $\beta \bar{R} \alpha \iff v_{k,t}\ \bar{R}_{jk}\ v_{j,r}$ et ... et $v_{q,u}\ \bar{R}_{pq}\ v_{p,s}$.

Et on note comme suit les prédicats correspondant à Rec et $\overline{\text{Rec}}$ pour la relation étendue :

(h) $\forall \alpha_m, \alpha_n \in \Pi^i$, $Rec(\alpha_m, \alpha_n) \iff R_m \subseteq R_n$

$\forall \beta_m, \beta_n \in \Pi^\ell$, $\bar{R}ec(\beta_m, \beta_n) \iff \bar{R}_m \subseteq \bar{R}_n$

Les propriétés 5 et 6 de 2.1.3.k s'appliquent à ces prédicats ; nous notons toujours $\preceq$ l'ordre large défini sur Π^i/γ et $\Pi^\ell/\bar{\gamma}$ où γ et $\bar{\gamma}$ sont définies à présent en fonction des coupes de R et $\bar{R}$.

On a donc :

Propriété 8 - Productivité dans Π^i et Π^ℓ

Une condition nécessaire et suffisante pour que α_m (resp. β_m) soit productif par rapport à α_n (resp. β_n) est qu'il existe au moins une composante $v_{j,r}$ de α_m (resp. $v_{p,t}$ de β_m) productive par rapport à la composante correspondante $v_{j,s}$ de α_n (resp. $v_{p,u}$ de β_n). Soit non $Rec_{jk}(v_{j,r}, v_{j,s})$, et non $\bar{R}ec_{jk}(v_{p,t}, v_{p,u})$.

Propriété 9 - Ordre large - Sup-demi-treillis

Le prédicat Rec (resp. $\bar{R}ec$) induit une relation d'ordre large notée $\preceq$ sur Π^i/γ (resp. $\Pi^\ell/\bar{\gamma}$). Une condition nécessaire et suffisante pour que ces ensembles soient des sup-demi-treillis est que toutes les relations étendues qui entrent dans la définition de R et $\bar{R}$ admettent au moins un élément universel.

i) Relation de cohérence large $\not\propto$ - Propriétés

La relation $R \subseteq \Pi^i \times \Pi^\ell$ induit une relation notée $\not\propto \subseteq E^i \times E^\ell$:

(i) $\forall e_m \in E^i$, $\forall e_n \in E^\ell$, $e_m \not\propto e_n \iff e_m \not\propto_{jk} e_n$ et ... et $e_m \not\propto_{pq} e_n$

et réciproquement :

$$\forall e_m \in E^i,\ \forall e_n \in E^\ell,\ e_n\ \bar{\chi} e_m \iff e_n\ \bar{\chi}_{jk}\ e_m \text{ et } \ldots \text{ et } e_n\ \bar{\chi}_{pq}\ e_m$$

Cette définition constitue la première définition de la cohérence large entre E^i et E^ℓ.

Comme pour la définition (h) de 2.1.3.j, les coupes de χ sont définies par :

$$(j)\ \forall e_m \in E^i,\ \chi_m = \chi_{jk,m} \cap \ldots \cap \chi_{pq,m} = \bigcup_{\beta \in R_\alpha} E^\ell_\beta$$

$$\forall e_m \in E^\ell_\beta,\ \chi_m = \bar{\chi}_{jk,m} \cap \ldots \cap \bar{\chi}_{pq,m} = \bigcup_{\alpha \in \bar{R}_\beta} E^i_\alpha$$

Les prédicats REC et $\overline{REC}$ sont notés, pour la relation de cohérence large :

$$(k)\ \forall e_m, e_n \in E^i,\ REC(e_m, e_n) \iff \chi_m \subseteq \chi_n$$

et

$$\forall e_m, e_n \in E^\ell,\ \overline{REC}(e_m, e_n) \iff \bar{\chi}_m \subseteq \bar{\chi}_n.$$

Les propriétés 4 et 7 de 2.1.3.k sont applicables :

Propriété 10 - Partitions hiérarchisées :

On a :

$$\forall (e_m) \in E^i/C^i,\ \exists (e_p) \text{ unique} \in E^i/\gamma' \mid (e_m) \subseteq (e_p)$$

$$\forall (e_m) \in E^\ell/C^\ell,\ \exists (e_p) \text{ unique} \in E^\ell/\bar{\gamma}' \mid (e_m) \subseteq (e_p)$$

où γ' et $\bar{\gamma}'$ sont à présent définies par rapport aux coupes de χ et $\bar{\chi}$.

Propriété 11 :

$$\forall \alpha_r, \alpha_s \in \Pi^i,\ Rec(\alpha_r, \alpha_s) \Rightarrow REC(e_m, e_n),\ \forall e_m \in E^i_{\alpha_r},\ \forall e_n \in E^i_{\alpha_s}.$$

$$\forall \beta_r, \beta_s \in \Pi^\ell,\ \bar{Rec}(\beta_r, \beta_s) \Rightarrow \overline{REC}(e_m, e_n),\ \forall e_m \in E^\ell_{\beta_r},\ \forall e_n \in E^\ell_{\beta_s}.$$

Dont on déduit facilement le corollaire suivant :

Propriété 12 :

$$\forall \alpha \in \Pi^i,\ \forall e_m, e_n \in E^i_\alpha,\ REC(e_m, e_n)$$

$$\forall \beta \in \Pi^\ell,\ \forall e_m, e_n \in E^\ell_\beta,\ \overline{REC}(e_m, e_n).$$

Les éléments d'une même classe de E^i/C^i ou de E^ℓ/C^ℓ sont non productifs les uns par rapport aux autres.

2.4.3. Traitement des omissions - Deuxième définition de la cohérenc large

a) Elément universel commun à tous les modèles

Les conséquences évoquées en 2.3.2 des omissions sont traitées en définissant des relations étendues à la valeur indéterminée u. Cette extension est obtenue en considérant que la valeur indéterminée est un élément universel pour tout modèle de variation, y compris le modèle neutre.

Cette extension générale des modèles suppose évidemment que la valeur indéterminée appartient à l'ensemble des valeurs de toute caractéristique utile, et elle est définie de la manière suivante :

- pour tout modèle M défini sur V, u est élément universel du modèle étendu $\mathcal{M}$
- les propriétés de réflexivité et de symétrie sont conservées et $M \subseteq \mathcal{M}$.

Soit :

(a) $\forall v \in V$, $u \mathcal{M} v$ et $v \mathcal{M} u$ et donc en particulier $u \mathcal{M} u$

$\forall v_r, v_s \in V$, $(v_r, v_s) \in M \Rightarrow (v_r, v_s) \in \mathcal{M}$.

Cette extension des modèles a pour conséquence que toutes les relations étendues (cf. 2.4.2.c) admettent la valeur indéterminée comme élément universel (cf. propriété 5 de 2.4.2).

La relation de cohérence χ obtenue en tenant compte de cette nouvelle spécification des modèles constitue la deuxième définition de la cohérence large, que nous utiliserons toujours désormais pour résoudre à la fois les variations et les omissions dans les processus de couplage de fichiers.

La totalité des propriétés définies dans le cas général en 2.4.2 sont naturellement valables pour cette nouvelle définition de la cohérence large.

La seule différence est que certaines propriétés particulières deviennent générales du fait de l'existence de la valeur indéterminée en tant qu'élément universel commun à toutes les relations.

b) Propriété générale des modèles étendus

L'extension définie par (a) de tous les modèles, y compris le modèle neutre, conserve à ceux-ci leur réflexivité et leur symétrie. On démontre facilement que, sauf cas limite où M_j est une relation universelle (cf. 2.4.2.e) la propriété suivante est toujours vérifiée :

Propriété 1 - Non transitivité des modèles étendus :

L'extension définie par (a) des modèles de variation est non transitive.

Démonstration : Sauf cas limite où M est une relation universelle, le modèle est tel que : $\exists v_r, v_s \in V_j \mid (v_r, v_s) \notin M_j$.

Par définition de M_j, on a :

$$(v_r, u) \in M_j \ (u, v_s) \in M_j \ (v_r, v_s) \notin M_j.$$

+++

Il en résulte en particulier que M_j ne peut être une relation d'équivalence, même si M en était une pour les valeurs définies de V_j.

c) Propriétés de la relation de cohérence étendue R_{jk}

La propriété 1 de 2.4.2 est toujours valable, à savoir : $R_{jk} \subseteq R_{jk} \subseteq V_j \times V_k$

Les propriétés générales 2 et 3 de 2.1.3.j sont toujours valables et on a :

Propriété 2 - Sup-demi-treillis :

Les ensembles V_j/γ_{jk} et $V_k/\bar{\gamma}_{jk}$ ordonnés par la relation $\preceq_{jk}$ sont des sup-demi-treillis.

Considérons à présent le cas d'un modèle général M_j relatif à un ensemble de caractéristiques synonymes dont l'ensemble de valeurs est V_j.

On démontre les propriétés suivantes relatives à toute relation étendue R_j, déduite selon la définition (b'), d'un modèle étendu $M_j \subseteq V_j \times V_j$ et d'une relation externe R_j :

Propriété 3 - Réflexivité de R_j :

Une condition suffisante pour que R_j soit réflexive est que la relation externe R_j soit réflexive.

Démonstration : $\forall v_r \neq u \in V_j$, $v_r R_j v_r$ d'après la réflexivité de R_j.

D'après l'hypothèse de réflexivité de M_j, $\forall v_r, v_r M v_r$.

D'où : $\forall v_r \neq u \in V_j$, $v_r M_j v_r$, $v_r R_j v_r \Rightarrow v_r R_j v_r$.

Montrons que R_j est également réflexive pour u :

Considérons un élément quelconque $(v_r, v_s) \in R_j$; u étant élément universel de M_j, on a : uM_jv_r, uM_jv_s, $v_rR_jv_s \Rightarrow uR_ju$.

+++

On démontre que, la relation externe R_j étant réflexive, on a également la propriété suivante :

Propriété 4 - Inclusion :

La relation externe R_j étant réflexive, on a $M_j \subseteq R_j \subseteq V_j \times V_j$.

Démonstration : Nous avons démontré dans la propriété précédente que uRu ; comme $R_j \subseteq R_j$, l'inclusion est donc démontrée en ce qui concerne la réflexivité. Considérons à présent un élément quelconque (v_r, v_s) de M_j, tel que $v_s = u$ (la démonstration est la même pour $v_r = u$, et pour $v_r \neq v_s \neq u$). Par définition de M_j on a $v_r M v_r$ et $v_r M u$. R_j étant réflexive, on a ainsi : $v_r R_j v_r$, soit :

$$v_r M_j v_r,\ v_r M_j u,\ v_r R_j v_r \Rightarrow v_r R_j u.$$

+++

Propriété 5 - Symétrie partielle de la relation étendue :

La relation étendue est symétrique pour la valeur indéterminée.

Soit :

$$\forall v \in V_j,\ vR_j u \text{ et } uR_j v.$$

Démonstration :

u étant universel pour R_j, on a : $\forall v_r,\ uR_j v_r$

et, par définition de R : $\forall v_r,\ \exists v_s \mid v_r R_j v_s$ ou $v_s R_j v_r$.

Supposons que $v_r R_j v_s$ (démonstration analogue dans l'autre cas) :

u étant élément universel pour M_j, on a $uM_j v_s$.

M_j étant réflexive, on a : $v_r M_j v_r$, et par conséquent, par définition de R_j :

$$\forall v_r,\ \exists v_s \mid v_r M_j v_r,\ uM_j v_s,\ v_r R_j v_s \Rightarrow v_r R_j u.$$

+++

Propriété 6 - Symétrie de la relation étendue :

Une condition suffisante pour que la relation étendue R_j soit symétrique est que la relation externe R_j soit symétrique.

Démonstration :

Par définition de R_j, on a :

$$\forall v_r, v_s,\ v_r R_j v_s \Leftrightarrow \exists v_t, v_u \mid v_t M_j v_r,\ v_u M_j v_s,\ v_t R_j v_u.$$

R_j étant symétrique : $v_t R_j v_u \Rightarrow v_u R_j v_t$ et :

$$\exists v_t, v_u \mid v_u M_j v_s,\ v_t M_j v_r,\ v_u R_j v_t \Rightarrow v_s R_j v_r$$

+++

L'étude de la transitivité de R_j ne peut être effectuée dans le cas général ; elle dépend pour chaque cas de l'interdépendance existant entre la définition des modèles et de la relation externe.

De ce point de vue, pour reprendre une remarque générale faite en 2.3.2, il est souhaitable de définir les modèles généraux comme des extensions de la relation externe ; on tient alors compte des variations conservant la cohérence des valeurs.

Exemple :

Considérons deux caractéristiques synonymes pour la propriété NOM ; soit V_j leur ensemble de valeurs :

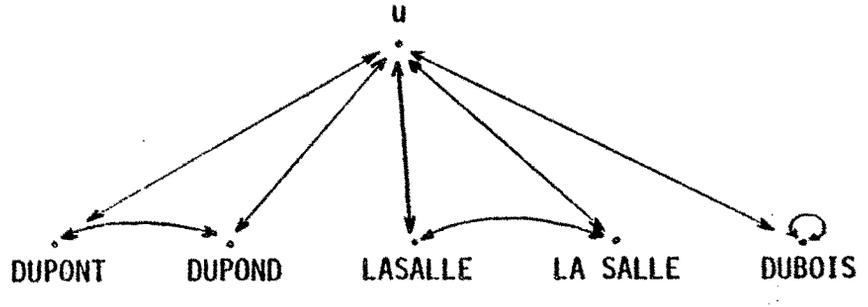
V_j = {'DUPONT', 'DUPOND', 'LASALLE', 'LA SALLE', 'DUBOIS'}.

Considérons le modèle M_j défini sur V_j de la façon suivante :

$\forall v_r, v_s$, $v_r M_j v_s$ $\Leftrightarrow$ "il existe un invariant phonétique commun à v_r et v_s".

Si à tout élément de V_j correspond une et une seule interprétation phonétique, M_j est une relation d'équivalence sur V_j dont le graphe est :

L'extension de M_j à la valeur indéterminée définie par a) donne la définition suivante de M_j :



Remarquons que, conformément à la propriété 1, M_j n'est pas transitive.

Si on considère l'égalité des noms comme relation externe, la relation étendue correspondante n'est pas transitive ; on a par exemple :

('DUPOND',u) $\in R_j$, (u,LASALLE) $\in R_j$, mais ('DUPOND','LASALLE') $\notin R_j$.

+++

Dans le cas où le modèle défini sur V_j est seulement le modèle neutre, on peut démontrer la propriété suivante :

Propriété 7 - Cas du modèle neutre :

Pour toute relation externe R_j non universelle, la relation étendue $\mathcal{R}_j$ correspondante pour le modèle neutre est non transitive.

Démonstration :

La relation externe n'étant pas universelle, $\exists v_r, v_s \mid (v_r, v_s) \notin R_j$.

u étant élément universel pour la relation étendue, on a : $u \mathcal{R}_j v_r$, $u \mathcal{R}_j v_s$

et d'après la propriété 5 : $v_r \mathcal{R}_j u$

donc : $\exists v_r, v_s$, $v_r \mathcal{R}_j u$, $u \mathcal{R}_j v_s$.

Par définition de $\mathcal{R}_j$, $v_r \mathcal{R}_j v_s \iff \exists v_t, v_u \mid v_t M_j v_r$, $v_u M_j v_s$, $v_t R_j v_u$.

Par définition du modèle neutre, v_t ne peut être que v_r ou u ;

il en est de même pour v_u. Or, R n'est pas définie pour la valeur indéterminée, et $(v_r, v_s) \notin R_j$, donc $(v_r, v_s) \notin \mathcal{R}_j$.

+++

d) Propriétés de la relation $\mathcal{R}$

Pour une relation $\mathcal{R} \subseteq \Pi^i \times \Pi^\ell$, la propriété 9 de 2.4.2 et la propriété 2 ci-dessus entraînent que :

Propriété 8 - Sup-demi-treillis :

Les ensembles Π^i/γ et $\Pi^\ell/\bar{\gamma}$, ordonnés par la relation $\preceq$, sont des sup-demi-treillis.

La propriété générale 8 de 2.4.2 et la propriété 8 ci-dessus induisent le corollaire suivant pour la valeur indéterminée :

Propriété 9 - Productivité de la valeur indéterminée :

Une condition suffisante pour que $\alpha_m \in \Pi^i$ (resp. $\beta_m \in \Pi^\ell$) soit productif par rapport à $\alpha_n \in \Pi^i$ (resp. $\beta_n \in \Pi^\ell$) est qu'au moins une des composantes de α_m (resp. β_m) soit indéterminée, la composante correspondante dans α_n (resp. β_n) n'étant pas un élément universel.

e) Relation de cohérence large - Propriétés

La relation $\not\asymp \subseteq E^i \times E^\ell$ induite entre les fichiers par une relation $R \subseteq \Pi^i \times \Pi^\ell$ et des modèles définis selon (a), constitue la seconde définition de la cohérence large entre deux fichiers. C'est de cette définition dont nous nous servirons désormais pour résoudre les problèmes de variations et d'omissions dans la définition des algorithmes de couplage.

Toutes les propriétés générales énoncées en 2.4.2 sont naturellement valables pour cette relation. Résumons les principales :

Propriété 10 - Sup-demi-treillis :

Les ensembles E^i/γ' et $E^\ell/\bar{\gamma}'$ sont des sup-demi-treillis ordonnés par la relation notée $\leq$.

Les relations γ' et $\bar{\gamma}'$ sont définies par rapport aux coupes de $\not\asymp$ et $\bar{\not\asymp}$.

Propriété 11 - Partitions hiérarchisées :

Les partitions E^i/C^i et E^i/γ' de E^i sont hiérarchisées, c'est-à-dire :
$\forall (e_m) \in E^i/C^i, \exists (e_p)$ unique $\in E^i/\gamma' \mid (e_m) \subseteq (e_p)$.
Les partitions E^ℓ/C^ℓ et $E^\ell/\bar{\gamma}'$ de E^ℓ sont hiérarchisées, c'est-à-dire :
$\forall (e_m) \in E^\ell/C^\ell, \exists (e_p)$ unique $\in E^\ell/\bar{\gamma}' \mid (e_m) \subseteq (e_p)$.

Propriété 12 :

$$\forall \alpha_p, \alpha_q \in \Pi^i,\ Rec(\alpha_p, \alpha_q) \Rightarrow REC(e_m, e_n),\ \forall e_m \in E^i_{\alpha_p},\ \forall e_n \in E^i_{\alpha_q}$$

$$\forall \beta_p, \beta_q \in \Pi^\ell,\ \bar{R}ec(\beta_p, \beta_q) \Rightarrow \overline{REC}(e_m, e_n),\ \forall e_m \in E^\ell_{\beta_p},\ \forall e_n \in E^\ell_{\beta_q}$$

La réciproque n'est généralement pas vraie.

Propriété 13 :

$$\forall e_m \in E^i_\alpha,\ \forall e_n \in E^\ell_\beta,\ e_m \not\asymp e_n \Rightarrow \alpha R \beta.$$

$$\forall e_m \in E^i_\alpha,\ \forall e_n \in E^\ell_\beta,\ e_n \bar{\not\asymp} e_m \Rightarrow \beta \bar{R} \alpha.$$

La réciproque n'est généralement pas vraie.

f) Exemple :

L'exemple ci-dessous résume à peu près tous les cas de figure relatifs aux variations et aux ambiguïtés qui y sont attachées.

Soit D^1 = {NOM,PRENOM,DATE-NAISSANCE} = $\{K_1^1,K_2^1,K_3^1\}$ et un fichier E^1 correspondant :

$E^1 = \{e_1,e_2,e_3,e_4,e_5\}$ dont les enregistrements sont représentés dans la figure ci-dessous

et D^2 = {NOM,PRENOM,DATE-MARIAGE} = $\{K_1^2,K_2^2,K_4^2\}$ et un fichier E^2 correspondant :

$E^2 = \{e_6,e_7,e_8,e_9\}$ dont les enregistrements sont représentés dans la figure ci-dessous.

Etant donné les relations externes suivantes :

$R_1 \subseteq V_1 \times V_1$, avec $V_1 = V_1^1 \cup V_1^2$ = {DUPONT,DUPOND,DUBOIS}
définie comme l'égalité

$R_2 \subseteq V_2 \times V_2$, avec $V_2 = V_2^1 \cup V_2^2$ = {JEAN,PAUL,PIERRE}
définie comme l'égalité

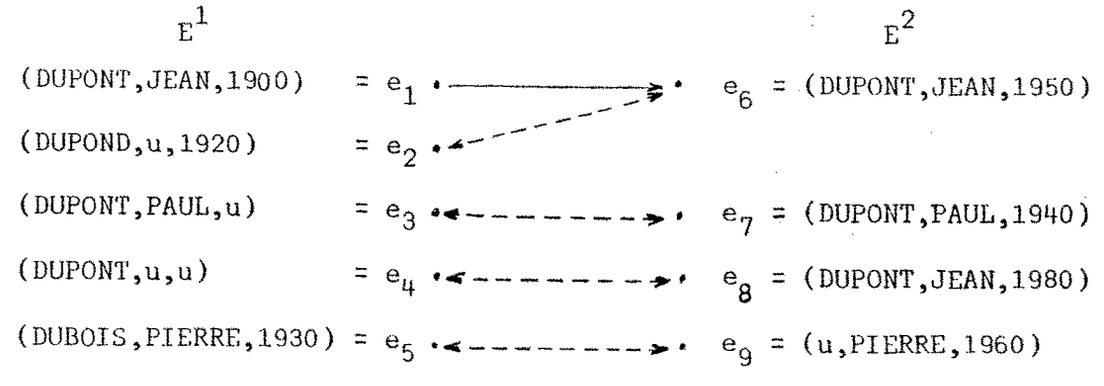
$R_{34} \subseteq V_3^1 \times V_4^2$, avec V_3^1 = {1900,1920,1930} et V_4^2 = {1940,1950,1960,1980}
définie par :

$\forall e_m \in E^1, \forall e_n \in E^2, e_m \sim e_n \Rightarrow C_3^1(e_m) R_{34} C_4^2(e_n) \iff 0 \leq C_4^2(e_n) - C_3^1(e_n) \leq$

La relation de cohérence totale qui s'en déduit est définie par :

$\forall e_m \in E^1, \forall e_n \in E^2, e_m \not\asymp e_n \iff e_m \not\asymp_1 e_n$ et $e_m \not\asymp_2 e_n$ et $e_m \not\asymp_{34} e_n$.

Le graphe ci-dessous représente cette relation (traits pleins) et la relation d'homologie que l'on veut retrouver (pointillés).



En raison des variations et des omissions, aucun des couples d'enregistrements homologues n'est retrouvé.

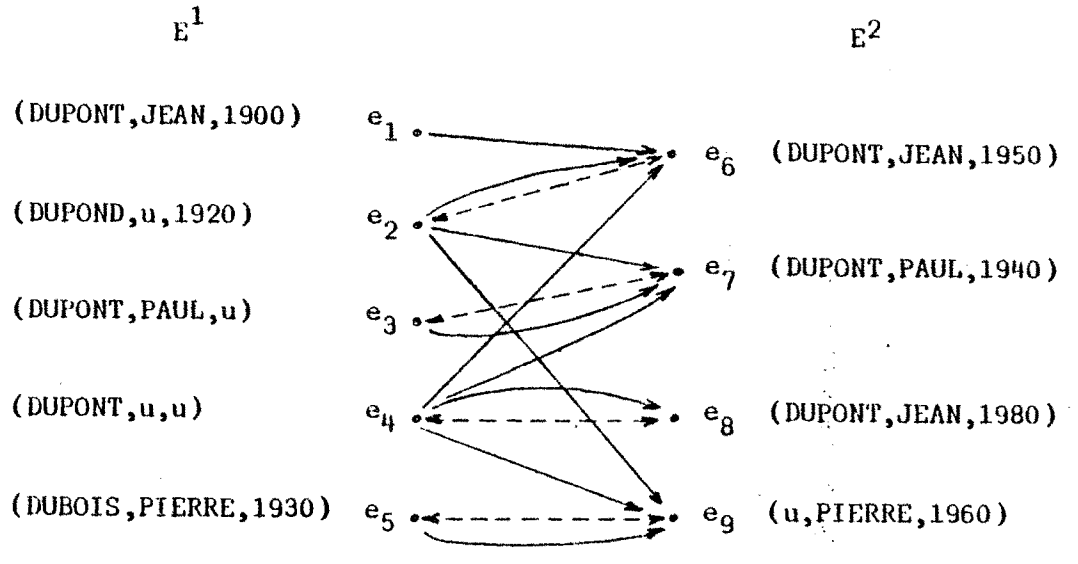
Si nous définissons les relations de cohérence étendue suivantes :

- R_1, par la définition sur V_1 d'un modèle sélectif fondé sur l'existence d'une interprétation phonétique commune de différents noms
- R_2, par la définition sur V_2 du modèle neutre étendu
- R_{34}, par la définition sur V_3^1 et V_4^2 du modèle neutre étendu

on obtient la définition suivante de la relation de cohérence large :

$$\forall e_m \in E^1,\ \forall e_n \in E^2,\ e_m \not\gamma\ e_n \iff e_m \not\gamma_1 e_n \text{ et } e_m \not\gamma_2 e_n \text{ et } e_m \not\gamma_{34} e_n$$

Le graphe ci-dessous représente cette relation (traits pleins), et la relation d'homologie (pointillés).



Tous les couples d'enregistrements homologues sont donnés par la relation de cohérence large ; cela est dû au fait que les modèles définis tiennent compte de toutes les variations effectives.

La classe de cohérence obtenue comprend en fait cinq classes de $\mathcal{H}$, cela est dû à l'ambiguïté créée par les modèles, mais aussi à l'ambiguïté naturelle du système d'information considéré (par exemple pour e_1, e_6).

Le fait que la valeur indéterminée soit un élément universel pour toutes les relations étendues explique le couplage de e_4 avec l'ensemble des éléments de E^2, ainsi que le couplage de e_2 et e_9, par exemple.

2.4.4. Conclusion

Le but recherché dans la définition des modèles de variation est d'élargir la notion de cohérence propre à toute relation externe ; il est évident que les relations obtenues ne sont jamais fonctionnelle et que le système d'informations ne répond pas aux conditions d'existence d'une solution unique étudiées en 2.2.

De plus, l'extension des relations est de nature à introduire des ambiguïtés supplémentaires qui se traduisent par un élargissement des classes de cohérence qui peuvent contenir plusieurs classes d'enregistrements homologues. La différence fondamentale avec la situation antérieur est que l'ambiguïté créée par les modèles est contrôlable : c'est l'utilisateur qui définit les modèles et il peut les modifier au besoin. Par ailleurs, cette ambiguïté est constructive : on peut effectivement coupler des enregistrements a priori non cohérents.

Comme nous l'avons déjà noté plus haut, une première solution pour limiter cette ambiguïté consiste à prendre en compte plus de relations externes dans la définition des relations de cohérence large, chaque fois que cela est possible. Une deuxième solution, qui a été évoquée en 2.4.2.e, consiste à pondérer les divers éléments de la relation de cohérence large, et de définir une méthodologie de sélection des solutions les meilleures compte tenu de ces évaluations (phase de sélection).

CHAPITRE 2.5. Algorithme général de couplage

2.5. Algorithme général de couplage

2.5.1. Algorithme de construction d'une classe de cohérence

2.5.1.a) Définitions - Notations

Etant donné deux fichiers E^i et E^ℓ, une relation de cohérence large $\mathcal{R} \subseteq E^i \times E^\ell$ définie comme en 2.4.3, la construction de la classe de cohérence relative à tout élément de $E = E^i + E^\ell$, consiste à trouver la composante connexe de $\mathcal{R}$ contenant cet élément.

Nous appellerons couvertures de la classe dans E^i et E^ℓ les ensembles de ses composants qui appartiennent respectivement à E^i et E^ℓ ; nous notons $X \subseteq E^i$ la couverture de la classe de E^i, $Y \subseteq E^\ell$ la couverture de la classe dans E^ℓ.

Nous appellerons enregistrement de départ l'élément noté $e_d \in E$ qui sert d'origine au processus de construction de la classe. Par définition même de celle-ci, tout élément de E qui lui appartient est un enregistrement de départ possible pour la construire ; nous verrons qu'en fait le choix de l'enregistrement de départ n'est pas indifférent du point de vue des performances de l'algorithme.

La construction de la classe à partir de e_d implique la détermination de coupes de $\mathcal{R}$ et $\bar{\mathcal{R}}$; la contrainte de connexité impose de considérer alternativement ces deux types de coupes jusqu'à ce que la classe soit entièrement obtenue.

On peut représenter de la façon suivante la construction des coupes de $\mathcal{R}$ et $\bar{\mathcal{R}}$ pour tout sous-ensemble de E^i et E^ℓ (cf. 2.1.3.j) :

Considérons les applications $X \to X^*$ et $Y \to Y^+$ définies par :

$$(a)\quad \forall X \subseteq E^i,\ X^* = \bigcup_{e_m \in X} \mathcal{R}_m \subseteq E^\ell$$

$$\forall Y \subseteq E^\ell,\ Y^+ = \bigcup_{e_n \in Y} \mathcal{R}_n \subseteq E^i$$

avec : $X = \emptyset \Rightarrow X^* = \emptyset$ et $Y = \emptyset \Rightarrow Y^+ = \emptyset$.

D'après ces définitions, on montre facilement la propriété suivante :

Propriété 1 :

$$\forall X',\ X \subseteq E^i,\ X' \subseteq X \Rightarrow X'^* \subseteq X^*$$
$$\forall Y',\ Y \subseteq E^\ell,\ Y' \subseteq Y \Rightarrow Y'^+ \subseteq Y^+$$

avec en particulier :

$$\forall X',\ X \subseteq E^i,\ X' = X \Rightarrow X'^* = X^*$$
$$\forall Y',\ Y \subseteq E^\ell,\ Y' = Y \Rightarrow Y'^+ = Y^+.$$

Considérons les compositions de ces applications notées $X \to (X^*)^+$ et $Y \to (Y^+)^*$ définies par :

$$(b)\quad \forall X \subseteq E^i,\ (X^*)^+ = \bigcup_{e_n \in X^*} \bar{\chi}_n \subseteq E^i$$

$$\forall Y \subseteq E^\ell,\ (Y^+)^* = \bigcup_{e_m \in Y^+} \chi_m \subseteq E^\ell$$

On vérifie aisément les propriétés suivantes de ces applications :

Propriété 2 - Extensivité :

$$\forall X \subseteq E^i,\ X \subseteq (X^*)^+$$
$$\forall Y \subseteq E^\ell,\ Y \subseteq (Y^+)^*$$

Propriété 3 - Isotonie :

$$\forall X',\ X \subseteq E^i,\ X' \subseteq X \Rightarrow (X'^*)^+ \subseteq (X^*)^+$$
$$\forall Y',\ Y \subseteq E^\ell,\ Y' \subseteq Y \Rightarrow (Y'^+)^* \subseteq (Y^+)^*$$

Considérons des sous-ensembles X et X' de E^i, et des sous-ensembles Y', $Y \subseteq E^\ell$. Si nous notons X-X' l'ensemble des éléments de X n'appartenant pas à X', et Y-Y' l'ensemble des éléments de Y n'appartenant pas à Y', on démontre la propriété suivante :

Propriété 4 :

$$\forall Y',\ Y \subseteq E^\ell,\ Y^+ - Y'^+ \subseteq (Y-Y')^+$$
$$\forall X',\ X \subseteq E^i,\ X^* - X'^* \subseteq (X-X')^*$$

Considérons un sous-ensemble X de E^i et un sous-ensemble Y de E^ℓ, si $\mathcal{X}$ et $\mathcal{Y}$ sont respectivement les ensembles d'élément maximaux de X et de Y, on a la propriété suivante :

Propriété 5 :

$$\forall X \subseteq E^i,\ X^* = \mathcal{X}^* \subseteq E^\ell$$
$$\forall Y \subseteq E^\ell,\ Y^+ = \mathcal{Y}^+ \subseteq E^i$$

La démonstration découle de la définition même des éléments maximaux.

2.5.1.b) Première définition de l'algorithme

Pour tout enregistrement de départ $e_d \in E$, la construction de la classe de cohérence correspondante peut être décrite de manière récurrente à l'aide des suites suivantes :

- X_n et Y_n qui représentent respectivement les couvertures de la classe dans E^i et E^ℓ à chaque pas de l'algorithme ;
- G_n qui représente l'ensemble des arêtes de la classe au pas n de l'algorithme ; F_n est l'ensemble des arêtes obtenues en construisant la coupe de X_n par $\mathcal{X}$ dans E^ℓ, $\bar{F}_n$ est l'ensemble des arêtes obtenues en construisant la coupe de Y_n par $\mathcal{Y}$ dans E^i.

On suppose désormais que $e_d \in E^i$; tout ce qui suit peut facilement être transposé au cas symétrique où $e_d \in E^\ell$.

Considérons les suites X_n, Y_n, G_n définies ci-dessous :

Initialisation :	$X_0 = \emptyset$	$Y_0 = \emptyset$	$G_0 = \emptyset,\ \bar{F}_0 = \emptyset$
pas 1 :	$X_1 = \{e_d\}$	$Y_1 = X_1^* = \mathcal{X}_d$	$G_1 = G_0 \cup \bar{F}_0 \cup F_1$
pas 2 :	$X_2 = Y_1^+$	$Y_2 = X_2^*$	$G_2 = G_1 \cup \bar{F}_1 \cup F_2$
------	------	------	---------
pas n :	$X_n = Y_{n-1}^+ = (X_{n-1}^*)^+$	$Y_n = X_n^* = (Y_{n-1}^+)^*$	$G_n = G_{n-1} \cup \bar{F}_{n-1} \cup F_n$

Par définition de ces suites, on a :

$$X_n = (X^*_{n-1})^+, \quad Y_n = (Y^+_{n-1})^*$$

D'après la propriété 2, on a donc :

Propriété 6 :

$$\forall n \geq 1, \; X_{n-1} \subseteq X_n, \; Y_{n-1} \subseteq Y_n$$

Par définition même de ces suites, il est clair qu'à chaque pas de l'algorithme X_n et Y_n correspondent respectivement à des sous-ensembles des couvertures de la classe dans E^i et E^ℓ, et que le graphe défini par $(X_n \cup Y_n, G_n)$ est connexe et contient e_d.

Les ensembles E^i et E^ℓ étant finis, on montre facilement, d'après la propriété 6, que les suites X_n et Y_n sont bornées, soit :

$$\exists_m > 1 \mid X_m = X_{m-1} \text{ ou } Y_m = Y_{m-1}$$

Remarque : Nous considérons séparément le cas limite où la classe ne comporte que l'élément de départ e_d. Dans ce cas, on a, dès le pas 1 de l'algorithme, $\mathcal{C}_d = \emptyset$ et il est évidemment inutile de poursuivre plus loin. Nous dirons alors que la classe est unitaire.

Propriété 7 - Condition d'arrêt de l'algorithme :

La condition $X_m = X_{m-1}$ ou $Y_m = Y_{m-1}$ est une condition d'arrêt de l'algorithme. Lorsqu'elle est vérifiée, la composante connexe de $\mathcal{C}$ recherchée est entièrement obtenue.

Démonstration :

Montrons que $X_m = X_{m-1} \Rightarrow Y_m = Y_{m-1}$:

on a $Y_m = X^*_m$ et $Y_{m-1} = X^*_{m-1}$

donc : $X_m = X_{m-1} \Rightarrow X^*_m = X^*_{m-1}$, d'après la propriété 1

et $Y_m = Y_{m-1}$.

Lorsque X_m est fermé pour l'application $X \to (X^*)^+$, Y_m est fermé pour l'application $Y \to (Y^+)^*$. On démontre de la même façon que $Y_m = Y_{m-1} \Rightarrow X_m = X_{m+1}$, et donc que lorsque Y_m est fermé, X_m l'est aussi.

La condition garantit donc la fermeture des deux couvertures au même pas de l'algorithme.

Il est par ailleurs simple de vérifier que $X_m = X_{m-1} \Rightarrow X_{m+1} = X_m$, et que $Y_m = Y_{m-1} \Rightarrow Y_{m+1} = Y_m$; les ensembles X_m et Y_m étant fermés, il est inutile de poursuivre l'exécution de l'algorithme. Tout pas supplémentaire n'apporterait aucun élément nouveau à la classe.

+++

Le coût de cet algorithme dépend essentiellement du nombre de coupes de $\mathcal{P}$ et $\bar{\mathcal{P}}$ qui sont à construire avant que la condition d'arrêt ne soit atteinte. Le coût élémentaire de la construction de chaque coupe dépend du nombre d'éléments qu'elle contient et des techniques d'accès considérées pour les fichiers à coupler. Notons c le coût moyen de la construction d'une coupe, qui est un invariant quel que soit l'algorithme de couplage considéré.

On peut remarquer que dans l'algorithme défini ci-dessus, le seul coût à évaluer est celui de la construction des couvertures X_n et Y_n ; les arcs sont tous donnés au cours de ces opérations.

Si nous notons $x_i = |X_i|$, $y_i = |Y_i|$, le coût de l'algorithme est donné pas à pas par :

pas 1 : c car $x_1 = 1$

pas 2 : c y_1 + c x_2

---

pas m : c y_{m-1} [+ c x_m]

selon que la condition $X_m = X_{m-1}$ est vérifiée, ou qu'une coupe supplémentaire doit être construite si l'algorithme s'arrête pour la condition $Y_m = Y_{m-1}$.

On a donc l'expression suivante du coût C_1 de cet algorithme :

$$C_1 = \sum_{j=1}^{m-1} c\,(x_j + y_j)$$

ou :

$$C_1 = \sum_{j=1}^{m-1} c\,(x_j + y_j) + c\,x_m$$

Cet algorithme est évidemment loin d'être optimal car à chaque pas, des coupes déjà effectuées aux pas précédents sont reconstruites. Si nous considérons par exemple la coupe de l'enregistrement de départ e_d, d'après la propriété 4, on a $X_1 \subseteq X_2 \subseteq \ldots \subseteq X_m$, et cette coupe est construite m-1 fois au moins.

2.5.1.c) Optimisation - Propriétés

Des améliorations très sensibles peuvent être apportées à l'algorithme précédent en utilisant certaines des propriétés énoncées dans les chapitres 2.4.2 et 2.4.3.

Considérons tout d'abord la propriété 12 de 2.4.3 :

$$\forall e_m \in E^i_\alpha,\ \forall e_n \in E^\ell_\beta,\ e_m \not\sim e_n \Rightarrow \alpha R \beta,\ \alpha \in \Pi^i,\ \beta \in \Pi^\ell$$

Un corollaire immédiat de cette propriété est qu'à la composante connexe de $\sim$ que l'on recherche, correspond un sous-ensemble connexe de R.

Notons $W \subseteq \Pi^i$, $Z \subseteq \Pi^\ell$ les couvertures de la classe dans Π^i et Π^ℓ définies respectivement par (cf. 2.1.3.j (g)) :

$$(c)\quad W = C^i(X) \text{ et } Z = C^\ell(Y).$$

Considérons alors la propriété 8 de 2.4.3 selon laquelle Π^i/γ et $\Pi^\ell/\bar{\gamma}$ sont des sup-demi-treillis ; la relation d'ordre $\preceq$ peut être utilisée pour sélectionner les éléments les plus productifs des couvertures W et Z.

Notons W' et Z' les sous-ensembles de W et Z contenant les représentants de toutes les classes de Π^i/γ et $\Pi^\ell/\bar{\gamma}$ qui couvrent les éléments de W et Z, soit :

$$(d)\quad W' = \{\alpha' \in W \mid (\alpha') \in \Pi^i/\gamma\} \text{ et } Z' = \{\beta' \in Z \mid (\beta') \in \Pi^\ell/\bar{\gamma}\}$$

W' et Z' sont respectivement des sous-ensembles ordonnés par $\preceq$ des sup-demi-treillis Π^i/γ et $\Pi^\ell/\bar{\gamma}$, et ils admettent donc chacun au moins un élément maximal.

Si nous notons $\mathcal{W}$ et $\mathcal{Z}$ les ensembles d'éléments maximaux de W' et Z', la propriété suivante découle immédiatement de la définition des sup-demi-treillis :

Propriété 8 - Couvertures de la classe dans Π^i et Π^ℓ :

La couverture W (resp. Z) de la classe dans Π^i (resp. Π^ℓ) peut être obtenue en considérant seulement les coupes de $\bar{R}$ (resp. R) relatives aux éléments de Z (resp. $\mathcal{W}$). Soit :

$W = Z^+$ et $Z = \mathcal{W}^*$ pour les coupes des relations $\bar{R}$ et R respectivement.

Propriété 9 - Corollaire :

Tout élément de W (resp. Z) contient dans sa coupe dans Π^ℓ (resp. Π^i) au moins un élément de Z (resp. $\mathcal{W}$).

Soit : $\forall \alpha_m \in W, \exists \beta \in Z \mid \beta \in R_m$

$\forall \beta_m \in Z, \exists \alpha \in \mathcal{W} \mid \alpha \in \bar{R}_m$

On démontre en effet que si cette propriété n'est pas vérifiée, alors la propriété 8 ci-dessus ne l'est pas non plus. Un autre corollaire est constitué par la propriété suivante :

Propriété 10 - Sous-ensembles minimaux pour la construction de W et Z :

Les ensembles $\mathcal{W}$ et Z sont minimaux pour la construction des couvertures W et Z de la classe dans Π^i et Π^ℓ.

On démontre en effet que si ces ensembles ne sont pas minimaux, certains des éléments de $\mathcal{W}$ ou Z ne sont pas des éléments maximaux des ensembles ordonnés W' et Z', ce qui est contraire à leur définition.

Notons $(W \cup Z, F)$ le sous-ensemble connexe de R correspondant à la classe, et F le sous-ensemble de F des arêtes unissant les éléments de $\mathcal{W}$ et Z. Nous pouvons énoncer la propriété suivante :

Propriété 11 - Connexité :

Le graphe défini par $(\mathcal{W} \cup Z, F)$ est un sous-graphe connexe de $(W \cup Z, F)$.

La démonstration s'effectue en prouvant que si $(\mathcal{W} \cup Z, F)$ n'est pas connexe, alors la classe $(W \cup Z, F)$ ne l'est pas non plus, ce qui est contraire à sa définition même.

Les propriétés 8, 9, 10, 11 énoncées ci-dessus peuvent être immédiatement transposées aux couvertures de la classe dans E^i et E^ℓ ; en effet, par définition même de W et Z (cf. (c)), la réciproque de la propriété 12 de 2.4.3 est vraie pour tous les éléments de W et Z :

$$\forall e_m \in E^i_\alpha \subseteq X,\ \forall e_n \in E^\ell_\beta \subseteq Y,\ e_m \not\asymp e_n \Rightarrow \alpha R \beta,\ \alpha \in W,\ \beta \in Z$$

et $\quad \forall \alpha \in Z,\ \forall \beta \in W,\ \alpha R \beta \Rightarrow e_m \not\asymp e_n,\ \forall e_m \in E^i_\alpha,\ \forall e_n \in E^\ell_\beta.$

Notons $X' = \bar{C}^i(W) \subseteq E^i$ et $Y' = \bar{C}^\ell(Z) \subseteq E^\ell$; on a évidemment $X' \subseteq X$ et $Y' \subseteq Y$ car $W \subseteq W$, $Z \subseteq Z$ et par définition $X = \bar{C}^i(W)$ et $Y = \bar{C}^\ell(Z)$.

La propriété 8 permet donc d'énoncer que la couverture X (resp. Y) de la classe dans E^i (resp. E^ℓ) peut être obtenue en considérant seulement les coupes de $\bar{\not\asymp}$ (resp. $\not\asymp$) pour les éléments de Y' (resp. X').

La propriété 11 de 2.4.3 qui dit que les partitions de E^i/C^i et E^i/γ' (resp. E^ℓ/C^ℓ et $E^\ell/\bar{\gamma}'$) sont hiérarchisées, permet de restreindre X' et Y' aux seuls représentants de E^i/C^i et E^ℓ/C^ℓ respectivement, sans modifier les coupes obtenues. Notons $\mathcal{X}$ et $\mathcal{Y}$ ces sous-ensembles de X' et Y'. On a donc la propriété suivante :

<u>Propriété 12 - Couvertures de la classe dans E^i et E^ℓ</u> :

> La couverture X (resp. Y) de la classe dans E^i (resp. E^ℓ) peut être obtenue en considérant seulement les coupes de $\bar{\not\asymp}$ (resp. $\not\asymp$) pour les éléments de $\mathcal{Y}$ (resp. $\mathcal{X}$). Soit : $X = \mathcal{Y}^+$, $Y = \mathcal{X}^*$.

Si on considère le sup-demi-treillis E^i/γ' (resp. $E^\ell/\bar{\gamma}'$), à la coupe X de la classe dans E^i (resp. Y dans E^ℓ) correspond un sous-ensemble X' (resp. Y') de représentants des éléments de ce sup-demi-treillis.

L'ensemble $\mathcal{X} \subseteq X$ (resp. $\mathcal{Y} \subseteq Y$) défini ci-dessus correspond au sous-ensemble des éléments maximaux de X' (resp. Y').

La propriété 9 devient :

<u>Propriété 13</u> :

> La coupe de tout élément de X (resp. Y) dans E^ℓ (resp. E^i) contient au moins un élément de $\mathcal{Y}$ (resp. $\mathcal{X}$).

Soit : $\forall e_m \in X, \exists e_n \in Y \mid e_n \in \gamma_m$

$\forall e_m \in Y, \exists e_n \in X \mid e_n \in \bar{\gamma}_m$

La propriété 10 devient :

Propriété 14 :

Les ensembles $X \subseteq X$ et $Y \subseteq Y$ sont minimaux pour la construction des couvertures X et Y de la classe.

Cette propriété est naturellement d'un grand intérêt pour la définition d'une limite inférieure du coût de tout algorithme de construction des classes de cohérence ; en ce qui concerne la construction des coupes de γ et $\bar{\gamma}$, il peut être évalué à :

$$C_{min} = c(x+y) \text{ où } x = |X|, \ y = |Y|.$$

Si nous notons $(X \cup Y, G)$ la composante connexe de γ que constitue une classe de cohérence, $G \subseteq G$ l'ensemble des arêtes unissant les éléments de X et Y, la propriété 9 devient :

Propriété 15 - Connexité :

Le graphe défini par $(X \cup Y, G)$ est un sous-graphe connexe de la classe de cohérence $(X \cup Y, G)$.

Cette propriété est également importante car elle garantit l'existence d'un chemin entre les noeuds définis par $X \cup Y$; l'ensemble des éléments maximaux peut donc être parcouru sans considérer les autres composants des couvertures. Si on considère de plus la propriété 14, on voit qu'il est possible de construire les couvertures de la classe en ne parcourant que le graphe $(X \cup Y, G)$.

L'utilisation des propriétés énoncées ci-dessus pour la définition d'une nouvelle version de l'algorithme de construction des classes nécessite quelques remarques et définitions préalables :

R1 : Les ensembles X et Y sont naturellement inconnus au départ. La propriété 13 garantit cependant qu'à chaque pas de l'algorithme 1, des éléments de X et de Y existent dans les coupes obtenues X_n et Y_n respectivement.

On peut donc utiliser la propriété 14 et limiter le nombre de coupes construites au pas suivant à ces seuls éléments maximaux. Pour cela, l'algorithme 1 doit être modifié par l'introduction d'une opération de sélection des éléments maximaux que nous représentons par les fonctions Max et $\bar{\text{M}}$ax ainsi définies :

(d) $\forall X \subseteq E^i$, Max(X) = X contient les éléments maximaux appartenant à X.

$\forall Y \subseteq E^\ell$, $\bar{\text{M}}$ax(Y) = Y contient les éléments maximaux appartenant à Y.

Nous donnerons une définition détaillée de ces fonctions en 2.5.1.e ; nous admettrons pour l'instant qu'elles vérifient les propriétés suivantes :

(e) $\forall X \subseteq E^i$, Max(X) = X = Max(X), Max($\emptyset$) = $\emptyset$, $\forall e_m \in E^i$, Max(e_m) = e_m.

$\forall Y \subseteq E$, $\bar{\text{M}}$ax(Y) = Y = $\bar{\text{M}}$ax(Y), $\bar{\text{M}}$ax($\emptyset$) = $\emptyset$, $\forall e_m \in E^\ell$, $\bar{\text{M}}$ax(e_m) = e_m.

R2 : Si l'algorithme ainsi modifié peut sélectionner à chaque pas les éléments de X et Y contenus dans les coupes obtenues, la propriété 15 garantit que d'un pas à l'autre de l'algorithme des éléments nouveaux de X et Y peuvent être trouvés (par définition, ces éléments étant maximaux, ils sont tous productifs les uns par rapport aux autres dans X ou dans Y). Ceci assure évidemment un parcours complet des éléments de X et Y, et donc la construction de toute la classe. Les propriétés 12 et 13 impliquent donc qu'un processus optimal de construction de la classe existe, à condition de ne construire qu'une seule fois la coupe de chaque élément de X et de Y.

R3 : Si l'enregistrement de départ est choisi arbitrairement dans E^i, rien ne permet d'affirmer qu'il appartient à X. Si cela n'est pas vérifié, la construction de la classe est plus coûteuse car elle ne s'effectue pas seulement par rapport aux éléments de X et de Y.

La propriété 13 permet cependant d'affirmer que si $e_d \notin X$, la coupe correspondante $\gamma_d = X_1^*$ construite au pas 1 contient au moins un élément de Y ; à partir du pas 2, on est donc ramené aux conditions optimales d'obtention des coupes de la classe.

R4 : On peut facilement vérifier que l'ensemble G' des arêtes de la classe obtenues en construisant les coupes des seuls éléments de X et de Y est un sous-ensemble de G. En effet, lorsque X et Y ont été parcourus, seules les arêtes G' dont les extrémités appartiennent à $X \times Y$ et $X \times Y$ ont été obtenues durant la construction des coupes. La classe étant un sous-ensemble de $X \times Y$, il reste à vérifier si certaines arêtes du graphe existent dans $(X-X) \times (Y-Y)$. Une opération supplémentaire doit donc être prévue en fin d'algorithme afin de compléter éventuellement G'. Nous notons cette opération de la façon suivante :

(f) $G = G' \cup ARC[(X-X), (Y-Y)]$

avec :

$ARC[\emptyset,(Y-Y)] = ARC[(X-X),\emptyset] = ARC[\emptyset,\emptyset] = \emptyset$.

Une définition plus détaillée de cette fonction est donnée en 2.5.1.e.

2.5.1.d) Version optimale de l'algorithme

Comme pour l'algorithme 1, nous définissons cette nouvelle version de manière récurrente. De nouvelles suites sont introduites : X'_m et Y'_n contiennent les éléments maximaux nouveaux extraits des coupes X_n et Y_n obtenues au pas n de l'algorithme. Les suites F'_n et $\bar{F}'_n$ représentent respectivement les ensembles d'arêtes obtenus lors de la construction des coupes X'^*_n et Y'^+_n. Comme défini précédemment, G'_n représente l'ensemble des arêtes de la classe obtenues jusqu'au pas n de l'algorithme inclus.

La définition de ces suites est la suivante :

Initialisation : $X'_0 = \emptyset$, $\mathcal{X}'_0 = \emptyset$ $\quad Y'_0 = \emptyset$, $\mathcal{Y}'_0 = \emptyset$ $\quad \bar{F}'_0 = \emptyset$, $G'_0 =$

pas 1 : $X'_1 = \{e_d\}$

$\mathcal{X}'_1 = \mathrm{Max}(X'_1) - \mathrm{Max}(X'_0)$ $\quad Y'_1 = X'^{*}_1 \cup Y'_0 = \mathcal{X}_d$ $\quad G'_1 = G'_0 \cup \bar{F}'_0 \cup$

$\mathcal{Y}'_1 = \bar{\mathrm{M}}\mathrm{ax}(Y'_1) - \bar{\mathrm{M}}\mathrm{ax}(Y'_0)$

pas 2 : $X'_2 = \mathcal{Y}'^{+}_1 \cup X'_1$

$\mathcal{X}'_2 = \mathrm{Max}(X'_2) - \mathrm{Max}(X'_1)$ $\quad Y'_2 = X'^{*}_2 \cup Y'_1$ $\quad G'_2 = G'_1 \cup \bar{F}'_1 \cup$

$\mathcal{Y}'_2 = \bar{\mathrm{M}}\mathrm{ax}(Y'_2) - \bar{\mathrm{M}}\mathrm{ax}(Y'_1)$

---

pas n : $X'_n = \mathcal{Y}'^{+}_{n-1} \cup X'_{n-1}$

$\mathcal{X}'_n = \mathrm{Max}(X'_n) - \mathrm{Max}(X'_{n-1})$ $\quad Y'_n = X'^{*}_n \cup Y'_{n-1}$ $\quad G'_n = G'_{n-1} \cup \bar{F}'_{n-}$

$\mathcal{Y}'_n = \bar{\mathrm{M}}\mathrm{ax}(Y'_n) - \bar{\mathrm{M}}\mathrm{ax}(Y'_{n-1})$ $\quad \cup F'_n$

Montrons que cet algorithme construit les mêmes couvertures que l'algorithme précédent ; pour cela, on note X'_n et Y'_n les coupes de l'algorithme 2 obtenues au pas n, et on montre que $\forall n \geq 1$, $X'_n = X_n$, $Y'_n = Y_n$, où X_n et Y_n sont les coupes obtenues au pas n par l'algorithme 1 (cf. 2.5.1.b).

Démonstration : On démontre cette propriété par récurrence :

pas 1 : $X'_1 = \{e_d\} = X_1$

$\mathcal{X}'_1 = \mathrm{Max}(X'_1) - \mathrm{Max}(X'_0) = \mathrm{Max}(X'_1)$ car $X'_0 = \emptyset$, et $\mathcal{X}'_1 = \mathrm{Max}(X_1)$ car $X_1 = X'_1$

d'où : $\mathrm{Max}(X_1) = e_d = \mathcal{X}'_1 = X_1$

$Y'_1 = X'^{*}_1 \cup Y'_0 = X'^{*}_1$ car $Y'_0 = \emptyset$, et $Y'_1 = X^{*}_1$ car $\mathcal{X}'_1 = X_1$, et donc $Y'_1 = Y_1 = X^{*}_1$.

On a : $\mathcal{Y}'_1 = \bar{\mathrm{M}}\mathrm{ax}(Y'_1) - \bar{\mathrm{M}}\mathrm{ax}(Y'_0) = \bar{\mathrm{M}}\mathrm{ax}(Y'_1)$ car $Y'_0 = \emptyset$, et $\mathcal{Y}_1 = \bar{\mathrm{M}}\mathrm{ax}(Y_1)$ car $Y_1 = Y'_1$. Donc $\mathcal{Y}_1 = \mathcal{Y}'_1$.

pas 2 : On a : $X'_2 = Y'^{+}_1 \cup X'_1 = Y'^{+}_1 \cup X_1$ car $X_1 = X'_1$, et puisque $Y'_1 = Y_1$:

$X'_2 = Y^{+}_1 \cup X_1$. D'après la propriété 5, $Y^{+}_1 = Y^{+}_1 = (X^{*}_1)^{+}$ et donc :

$X'_2 = (X^{*}_1)^{+} \cup X_1 = Y^{+}_1$ car $X_1 \subseteq (X^{*}_1)^{+} = Y^{+}_1$ et donc : $X'_2 = X_2 = (X^{*}_1)^{+}$.

Montrons que $Y'_2 = Y_2$:

On a : $X'_2 = X_2 - X_1$; deux cas sont possibles au départ :

(1) Si e_d n'est pas un élément maximal, on a : $X'_2 = X_2$, et par conséquent $Y'_2 = X^{*}_2 \cup Y_1$ puisque $X'_2 = X_2$ et $Y'_1 = Y_1$ et, d'après la propriété 5 :

$Y'_2 = X^{*}_2 \cup Y_1 = X^{*}_2$ car $Y_1 \subseteq X^{*}_2 = (Y^{+}_1)^{*}$, et donc $Y'_2 = Y_2$.

(2) Si e_d est un élément maximal de X_2, on a $X'_2 = X_2 - X_1$.

Donc $Y'_2 = (X_2 - X_1)^{*} \cup Y'_1 = (X_2 - X_1)^{*} \cup X^{*}_1$ puisque $Y'_1 = Y_1$ et d'après la propriété 5.

Montrons que $Y'_2 \subseteq Y_2$:

Par définition, $X_2 - X_1 \subseteq X_2$, et d'après la propriété 2 :

$(X_2 - X_1)^{*} \subseteq X^{*}_2$, et donc :

$Y'_2 \subseteq X^{*}_2 \cup X^{*}_1 = X^{*}_2$ puisque $X^{*}_1 \subseteq X_2 \Rightarrow X^{*}_1 \subseteq X^{*}_2$.

Donc, d'après la propriété 5, et puisque $X_2 = X'_2$:

$$Y'_2 \subseteq X^{*}_2 = Y_2.$$

Montrons que $Y'_2 \supseteq Y_2$:

$Y'_2 = (X_2 - X_1)^{*} \cup X^{*}_1$. La propriété 4 donne alors :

$(X_2 - X_1)^{*} \supseteq X^{*}_2 - X^{*}_1$ et donc :

$Y'_2 \supseteq (X^{*}_2 - X^{*}_1) \cup X^{*}_1 = X^{*}_2$, et $Y'_2 \supseteq Y_2 = X^{*}_2$

Donc, $Y'_2 = Y_2 = X^{*}_2$, et la proposition est vraie aux pas 1 et 2 de l'algorithme.

Si on suppose $Y'_n = Y_n$, $X'_n = X_n$ au pas n, on montre de la même façon que $X'_{n+1} = X_n$ et $Y'_{n+1} = Y_n$ (seul de cas (1) de la démonstration n'est évidemment plus à considérer).

+++

La seconde version de l'algorithme donne donc à chaque pas les mêmes couvertures que celles données par la précédente ; il en résulte qu'on a également la condition d'arrêt suivante :

$$\exists m \geq 1 \text{ tel que } X_m = X_{m-1} \text{ ou } Y_m = Y_{m-1} \text{ (condition 1).}$$

Cette condition pourrait donc être utilisée de la même façon que précédemment ; en fait, on démontre que l'on peut arrêter l'algorithme dès que $X'_n = \emptyset$ ou dès que $Y'_n = \emptyset$.

On a en effet :

$$X'_{m-1} = \emptyset \iff X_m = X_{m-1} \text{ et } Y_{m-1} = Y_{m-2}$$
$$Y'_{m-1} = \emptyset \iff X_m = X_{m-1} \text{ et } Y_m = Y_{m-1}.$$

Démonstration :

a) Montrons que $X'_{m-1} = \emptyset \Rightarrow X_m = X_{m-1}$ et $Y_{m-1} = Y_{m-2}$

$X'_{m-1} = \emptyset \Rightarrow X'^{*}_{m-1} = \emptyset$ et donc : $Y_{m-1} = X'^{*}_{m-1} \cup Y_{m-2} = Y_{m-2}$

Donc $X'_{m-1} = \emptyset \Rightarrow Y_{m-1} = Y_{m-2}$ et $Y'_{m-1} = Y_{m-1} - Y_{m-2} = \emptyset$

Par conséquent, $X_m = Y^{+}_{m-1} \cup X_{m-1} = X_{m-1}$.

Donc : $X'_{m-1} = \emptyset \Rightarrow Y_{m-1} = Y_{m-2}$ et $X_m = X_{m-1}$.

b) Démontrons l'implication inverse :

On a : $Y_{m-1} = X'^{*}_{m-1} \cup Y_{m-2}$ donc $Y_{m-1} = Y_{m-2} \Rightarrow X'^{*}_{m-1} \cup Y_{m-2} = Y_{m-2}$.

Trois cas sont logiquement possibles :

(1) $X'^{*}_{m-1} = \emptyset$:

Par conséquent : $X'_{m-1} = \emptyset$

(2) $Y_{m-2} = \emptyset$:

Par conséquent : $X'^{*}_{m-1} = \emptyset \Rightarrow X'_{m-1} = \emptyset$

(3) $X'^{*}_{m-1} \subseteq Y_{m-2}$:

On a : $Y_{m-2} = X^{*}_{m-2} = X^{*}_{m-2}$ d'après la proposition 5 ; donc d'après l'hypothèse : $X'^{*}_{m-1} \subseteq X^{*}_{m-2}$, et par définition de X'_{m-1} :

(4) $(X_{m-1}-X_{m-2})^* \subseteq X^*_{m-2}$

Or, $X_{m-2} \subseteq X_{m-1} \Rightarrow X_{m-2} \subseteq X_{m-1}$, et par conséquent

$(X_{m-1}-X_{m-2}) \not\subseteq X_{m-2}$

L'inclusion (a) ne peut donc être vérifiée car elle est contraire à la propriété 1.

Seules les possibilités (1) et (2) peuvent donc exister ; elles conduisent toutes deux à $X'_{m-1} = \emptyset$ et l'implication inverse est démontrée. La seconde propriété se démontre de manière analogue.

+++

On a alors la propriété suivante :

Propriété 16 - Condition d'arrêt de l'algorithme :

> La condition 2 définie par $X'_m = \emptyset$ ou $Y'_n = \emptyset$ est équivalente à la précédente condition définie par $X_n = X_{n-1}$ ou $Y_n = Y_{n-1}$ pour l'arrêt de l'algorithme.

En pratique, on évaluera la condition $X'_n = \emptyset$ dès que X'_n est construit, et la condition $Y'_n = \emptyset$ seulement si $X'_n = \emptyset$, puisque $X'_n = \emptyset \Rightarrow Y'_n = \emptyset$. On arrêtera l'algorithme dès que l'une ou l'autre de ces conditions élémentaires est vérifiée.

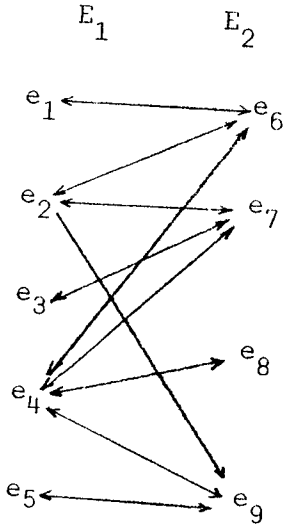
Par définition même des suites X'_n et Y'_n, on montre facilement que :

$$\forall n \geq 1,\ X'_n \cap X'_{n-1} = \emptyset,\ Y'_n \cap Y'_{n-1} = \emptyset$$

et donc que la construction des coupes de la classe s'effectue en considérant une fois et une seule les éléments maximaux de celles-ci. La propriété 15 garantit par ailleurs que tous les éléments maximaux sont accessibles par le processus décrit, et donc que toute la classe est effectivement construite.

La propriété 14 garantit alors que l'algorithme défini précédemment est optimal pour le nombre de constructions de coupes de $\mathcal{X}$ et $\bar{\mathcal{X}}$.

Exemple : Reprenons la classe considérée dans l'exemple de 2.4.3 :



- Supposons que $e_d = e_2$ on a :

pas 1 : $X_1 = \{e_2\}$

$X'_1 = \{e_2\}$ $\quad Y_1 = \{e_6,e_7,e_9\} = X'^*_1$

$y'_1 = \{e_6,e_7,e_9\} = y_1$

pas 2 : $X_2 = y'^+_1$ $\quad X_1 = \{e_1,e_2,e_3,e_4,e_5\}$

$X'_2 = X_2 - X_1 = \{e_4\} = X_2$

$Y_2 = X'^*_2 \cup Y_1 = \{e_6,e_7,e_8,e_9\}$

$y'_2 = y_2 - y_1 = \{e_6,e_7,e_9\} - \{e_6,e_7,e_9\} = \emptyset$

arrêt de l'algorithme.

On a bien obtenu les couvertures X et Y de la classe ; les éléments maximaux pour lesquels des coupes ont été construites sont notés X dans la figure.

En fin de traitement, on a $X-(X_2 \cup \{e_d\}) = \{e_1,e_3,e_5\}$ et $Y-y_2 = \{e_8\}$; la fonction ARC doit vérifier si les arcs (e_1,e_8), (e_3,e_8), (e_5,e_8) appartiennent à la classe ; cette vérification n'apporte rien de plus dans l'exemple traité.

+++

2.5.1.e) Recherche des éléments maximaux dans une coupe

Considérons la relation d'ordre large notée $\leq$ sur les classes de E^i/γ' et celles de $E^\ell/\bar{\gamma}'$:

$$\forall e_m, e_n \in E^i/\gamma',\ e_m \leq e_n \iff REC(e_m, e_n)$$
$$\forall e_m, e_n \in E^\ell/\bar{\gamma}',\ e_m \leq e_n \iff \overline{REC}(e_m, e_n).$$

Dans l'expression ci-dessus, les classes sont désignées par leur représentant ; étant donné un sous-ensemble X de E^i, notons X' un ensemble de représentants des éléments de E^i/γ' contenus dans X.

Par définition, $e_m \in E^i/\gamma'$ est élément maximal de X' si et seulement si :

$$\forall e_n \in X',\ e_m \neq e_n \iff \forall e_n \in X',\ \underline{\text{non}}\ REC(e_m, e_n)\ \underline{\text{ou}}\ e_m = e_n.$$

Soit :

(g) $\forall e_n \in X'-\{e_m\}$, $\underline{\text{non}}\ REC(e_m, e_n)$.

Les fonctions Max et $\bar{\text{Max}}$ utilisées dans l'algorithme optimal sont définies respectivement sur des sous-ensembles X de E^i et Y de E^ℓ ; comme on ne connaît pas a priori les ensembles de représentants X' et Y' correspondants, ces fonctions ne peuvent simplement se limiter au test (g) ci-dessus pour les éléments de X et Y ; elles doivent de plus sélectionner les éléments de X' et Y'. Si on note $(e_m) \in E^i/\gamma'$ la classe de représentant $e_m \in X$, vérifier que e_m est un élément maximal de X peut alors s'écrire :

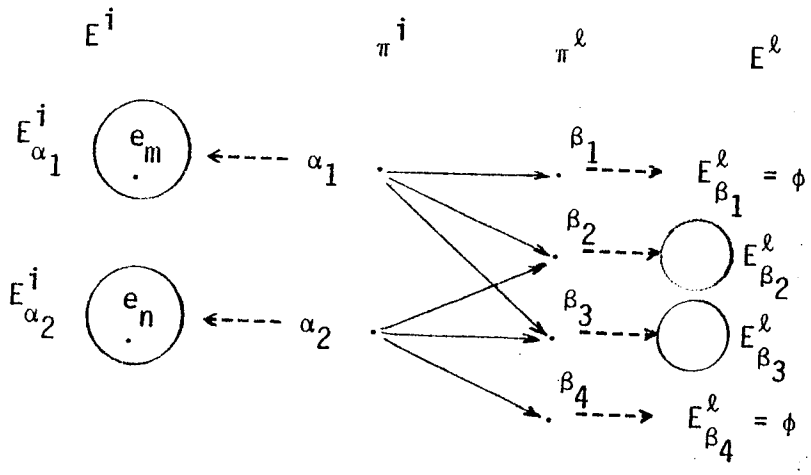
(g') $\forall e_n \in X-(e_m)$, $\underline{\text{non}}\ REC(e_m, e_n)$.

Pour des raisons de coût, on ne peut se permettre d'évaluer directement le prédicat $REC(e_m, e_n)$ qui implique la comparaison des coupes γ_m et γ_n, donc l'accès aux enregistrements correspondants. La propriété 12 de 2.4.3 permet de ramener l'évaluation de la productivité entre enregistrements à celle des éléments de π^i correspondants ; on déduit en effet de cette propriété que :

$$\forall e_m \in E^i_{\alpha_p},\ \forall e_n \in E^i_{\alpha_q},\ \underline{\text{non}}\ REC(e_m, e_n) \Rightarrow \underline{\text{non}}\ Rec(\alpha_p, \alpha_q).$$

La réciproque n'est pas toujours vérifiée ; ceci a pour conséquence que certains éléments de X peuvent être considérés à tort comme éléments maximaux.

Exemple :



On a : non $Rec(\alpha_1,\alpha_2)$ et non $Rec(\alpha_2,\alpha_1)$; e_m et e_n sont donc considérés comme éléments maximaux de X, alors que dans la réalité, $\gamma_m = \gamma_n = E^\ell_{\beta_2} \cup E^\ell_{\beta_3}$.

e_m et e_n appartiennent donc à une même classe de E^i/γ', et seul l'un des éléments de $X = E^i_{\alpha_1} \cup E^i_{\alpha_2}$ devrait être considéré comme élément aximal représentant cette classe.

+++

Nous considérons que ceci constitue un inconvénient moindre que la comparaison (et donc à la construction) systématique des coupes de γ et $\bar{\gamma}$.

La recherche des éléments maximaux de X est donc limitée à une comparaison deux à deux de ses éléments, soit en principe x(x-1) comparaisons si $x = |X|$. On peut notablement diminuer ce nombre en éliminant, dès qu'ils sont trouvés, tous les éléments non maximaux. La transposition de (g') aux éléments de π^i/γ donne :

$$\alpha_p \text{ maximal} \iff \forall \alpha_q \in W'-(\alpha_p),\ \text{non } Rec(\alpha_p,\alpha_q)$$

où W' note comme précédemment l'ensemble des classes de π^i/γ représentées dans X.

On en déduit donc :

$$\alpha_p \text{ non maximal} \iff \exists \alpha_q \in W'-(\alpha_p) \mid Rec(\alpha_p,\alpha_q)$$

que l'on peut écrire, par définition de π^i/γ :

(h) α_p non maximal $\Leftrightarrow \exists \alpha_q \in W' \mid Rec(\alpha_p,\alpha_q)$ et non $Rec(\alpha_q,\alpha_p)$

car cette condition ne peut être vraie que si α_p et α_q appartiennent à des classes différentes de π^i/γ.

Rappelons que l'évaluation des prédicats *Rec et* $\tilde{Rec}$ peut s'effectuer en utilisant la propriété 8 de 2.4.2, et plus particulièrement la propriété 9 de 2.4.3 relative à la productivité de la valeur indéterminée.

Une première définition de l'algorithme de sélection des éléments maximaux de X est donnée ci-contre ; on suppose ici que X est représenté par un vecteur de x éléments qui contiennent les références aux éléments de E^i qui le composent. La fonction Max prend comme valeur un vecteur du même type. Le vecteur de travail T a pour contenu initial X. Conventionnellement, les éléments supprimés dans un ensemble sont représentés par la valeur 0 dans les éléments correspondants du vecteur. La fonction ACC(m) accède directement à l'enregistrement d'adresse m ; elle prend pour valeur l'élément $\alpha_p \in \pi^i$ correspondant à cet enregistrement.

```
Fonction Max(X,x)
    T := COPIE(X,x) co transfert de contenu de X dans T ; ineffectif
                    si x = 0 co
    m := 1
(1) tantque m < x
      faire
          Sup1 := Sup2 := faux
(2)       si T(m) ≠ 0
          alors
              α_p = ACC(T(m))
(3)           s := m+1
(4)           tantque (non Sup1 ou Sup2) et s ≤ x
                faire
(5)               si T(s) ≠ 0
                  alors
                      α_q = ACC(T(s))
(6)                   Sup1 := Rec(α_p,α_q)
                      Sup2 := Rec(α_q,α_p)
(7)                   si Sup2
                      alors
                          T(s) := 0 co élimination α_q non maximum co
                      finsi
                  finsi
                  s := s+1
                finfaire
(8)           si Sup1 et non Sup2 co condition (h) co
                alors
                    T(m) := 0 co élimination α_p non maximum co
                finsi
          finsi
          m := m+1
(9)   finfaire
      Max := T
Finfonction
```

L'itération définie par (1) détermine la sélection de α_p dans le sous-vecteur de T donné par $m \in [1,x-1]$. La condition (2) limite ce parcours aux seuls éléments sous sous-vecteur non déjà éliminés (car non maximaux).

Une fois α_p obtenu, l'affectation (3) et l'itération (4) déterminent la sélection de α_q dans le sous-vecteur de T défini par $s \in [m+1,x]$. La condition (5) limite ce parcours aux seuls éléments du sous-vecteur non déjà éliminés. Pour tout couple (α_p,α_q) ainsi sélectionné, on évalue en (6) les prédicats $Rec(\alpha_p,\alpha_q)$ et $Rec(\alpha_q,\alpha_p)$. On élimine α_q si le prédicat $Rec(\alpha_q,\alpha_p)$ est vrai ; l'itération (4) compare donc α_p à tous les éléments α_q donnés par $s \in [m+1,x]$, et élimine tous les $\alpha_q \leq \alpha_p$. D'après la condition (4), ce processus s'arrête soit quand $q > x$ (tout le sous-vecteur a été parcouru), soit si la condition (h) est vérifiée, c'est-à-dire si $\exists s \in [m+1,x] \mid Rec(\alpha_p,\alpha_q)$ et non $Rec(\alpha_q,\alpha_p)$. α_p n'est donc pas maximal et il est inutile de poursuivre sa comparaison avec d'autres éléments ; il est éliminé en (8). Par opposition, si (4) se termine sur la condition $s > x$, $\exists s \in [m+1,x]$ tel que (h) soit vérifiée, et α_p est un élément maximum. Remarquons que si plusieurs éléments d'une même classe de π^i/γ sont présents dans X, dès que l'on accède au premier de ces éléments dans l'ordre du parcours séquentiel, tous les autres sont éliminés durant l'itération (4) correspondante ; α_p demeure alors le seul représentant de la classe. L'ensemble des éléments maximaux de X est donc donné par celui des éléments qui n'ont pas été éliminés en (7) ou en (8).

Remarquons que les propriétés (e) énoncées plus haut pour la fonction Max sont vérifiées par cet algorithme :

- si $X = \phi$, soit $x = 0$ on passe directement à (9) et $Max(X) = T = \phi$
- si $X = \{e_m\}$, soit $x = 1$, on passe directement à (9) et $Max(X) = T = \{e_m\}$
- si X ne contient que des éléments maximaux, aucun élément de T n'est éliminé en (7) ou en (8) et on a $Max(X) = T = X$.

La fonction $\bar{Max}$ qui sélectionne les éléments maximaux d'un sous-ensemble Y de E^ℓ est définie de manière analogue, en utilisant le prédicat $\bar{Rec}$.

Reprenons la définition des suites X'_n et Y'_n donnée en 2.5.1.d :

$$X'_n = \mathrm{Max}(X_n)-\mathrm{Max}(X_{n-1}) = X_n-X_{n-1}$$
$$Y'_n = \overline{\mathrm{Max}}(Y_n)-\overline{\mathrm{Max}}(Y_{n-1}) = Y_n-Y_{n-1}.$$

Il est possible de restreindre la recherche des éléments de X_n à un sous-ensemble de X_n ; on a en effet :

$X_n = (X_n-X_{n-1}) \cup X_{n-1}$, et donc : $\mathrm{Max}(X_n) = \mathrm{Max}((X_n-X_{n-1}) \cup X_{n-1})$.

Comme on a aussi $X_{n-1} \subseteq X_n$, on démontre que :

$$\mathrm{Max}((X_n-X_{n-1}) \cup X_{n-1}) = \mathrm{Max}((X_n-X_{n-1}) \cup \mathrm{Max}(X_{n-1})) = \mathrm{Max}((X_n-X_{n-1}) \cup X_{n-}$$

La recherche des éléments maximaux de X_n peut donc être limitée aux éléments de $(X_n-X_{n-1}) \cup X_{n-1}$. Par ailleurs, X_{n-1} étant un ensemble de μ éléments maximaux, il est inutile de comparer ces μ éléments entre eux ; il suffit donc de comparer les δ éléments de X_n-X_{n-1} aux $\mu+\delta$ éléments de $(X_n-X_{n-1}) \cup X_{n-1}$. L'algorithme précédent est donc modifié comme suit :

- on adopte une représentation vectorielle des ensembles analogue à celle définie précédemment, et on représente respectivement les ensembles X_n-X_{n-1} et X_{n-1} par les vecteurs R de δ éléments et S de μ éléments. La fonction MAX(R,δ,S,μ) utilise alors un vecteur de travail T de dimension $\delta+\mu$ dans lequel les vecteurs R et S sont concaténés en début de traitement ;
- pour que l'algorithme soit général, on a prévu le cas où $\mu = 0$, ce qui peut se produire au pas 1 de l'algorithme si $e_d \in E^{\ell}$;
- l'itération (2) est alors interrompue pour $m > \delta$ si $\mu \geq 1$ (tous les éléments de X_n-X_{n-1} sont comparés aux éléments de $(X_n-X_{n-1}) \cup X_{n-1}$), ou pour $m = \delta$ si $\mu = 0$ (T est alors égal à R, et la recherche des éléments maximaux doit s'arrêter après la comparaison du couple T (δ-1), T(δ)) ;
- l'itération (4) est interrompue pour $s > \delta+\mu$;
- l'algorithme de la fonction MAX est donc facilement déduit de celui de la fonction Max précédemment définie ; on a :

```
Fonction MAX(R,δ,S,μ)
    T := CONCAT(R,δ,S,μ)
    m := 1
    k := si μ ≥ 1 alors δ+1 sinon δ finsi
  tantque m < k
    faire
      Sup1 := Sup2 := faux
      si T(m) ≠ 0
        alors
          α_p := ACC(T(m))
          s := m+1
          tantque (non Sup1 ou Sup2) et s ≤ δ+μ
            faire
              si T(s) ≠ 0
                alors
                  α_q = ACC(T(s))
                  Sup1 := Rec(α_p,α_q)
                  Sup2 := Rec(α_q,α_p)
                  si Sup2
                    alors
                      T(s) := 0
                    finsi
                finsi
              s := s+1
            finfaire
          si Sup1 et non Sup2
            alors
              T(m) := 0
            finsi
        finsi
      m := m+1
    finfaire
    MAX := T
finfonction
```

On remarque que si $\mu = 0$, cet algorithme est équivalent à celui de la fonction Max.

Une fonction $\overline{MAX}$ peut être définie de manière analogue pour construire $y_n = \bar{M}ax(Y_n)$ à moindre coût. Nous écrirons donc désormais pour l'algorithme de couplage :

$$X'_n = MAX((X_n - X_{n-1}), X_{n-1}) - X_{n-1}$$
$$Y'_n = \overline{MAX}((Y_n - Y_{n-1}), Y_{n-1}) - Y_{n-1}$$

Remarque : Au pas 1 de l'algorithme, on a $X_1 = \{e_d\}$, $X_0 = \phi$, $X_0 = \phi$ *et* $X'_1 = MAX(X_1, \phi) = Max(X_1) = \{e_d\}$ *car alors* $\mu = 0$, $\delta = 1$.
De même, $Y_0 = Y_0 = \phi$ *et par conséquent :*
$\overline{MAX}(Y_1) = \bar{M}ax(Y_1)$ *car alors* $\mu = 0$, $\delta \geq 1$.

+++

Le coût de la fonction MAX en terme de nombre d'accès aux éléments de E^i est difficile à évaluer exactement ; tout dépend, pour des ensembles X_n, X_{n-1}, X_{n-1} donnés :

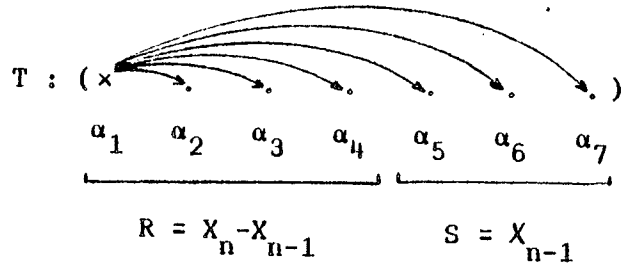
- du nombre d'éléments maximaux trouvés (et par conséquent du nombre d'enregistrements qui ont pu être éliminés en cours de comparaison) ;
- du nombre de monotonies de l'ordre $\leq$ rencontrées lors du processus de comparaison. Plus ces monotonies sont nombreuses, plus l'élimination des éléments non maximaux est rapide, et plus la construction de X_n est rapide.

Considérons des valeurs données de δ et μ ; notons $\rho \leq \delta$ le nombre d'éléments maximaux existants dans X'_n. On supposera $\rho \geq 1$ ($\rho = 0$ correspond à la condition d'arrêt de l'algorithme de couplage). Nous évaluons ci-dessous le coût de l'algorithme pour $\rho = 1$ et pour $\rho = \delta$. Nous ne retenons dans cette évaluation que le nombre d'accès aux éléments de E^i, qui en constitue la part prépondérante.

$\rho = 1$:

Le cas le plus favorable est celui où l'élément maximal est considéré le premier, tous les autres éléments de $X_n - X_{n-1}$ étant ≤ à celui-ci.

Exemple : $\delta = 4$, $\mu = 3$. L'élément maximum est noté ×.



avec $\alpha_2 \leq \alpha_1$, $\alpha_3 \leq \alpha_1$, $\alpha_4 \leq \alpha_1$.

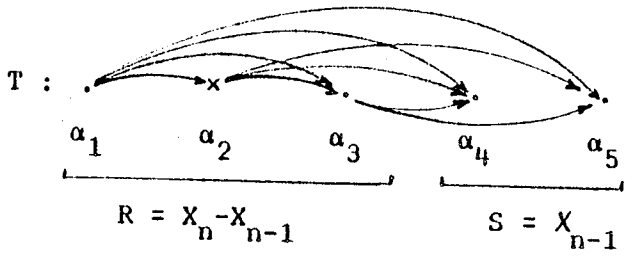
+++

Il est facile de voir que le coût minimum est alors :

$$C_{min} = \delta + \mu.$$

Le cas le plus défavorable est celui où les $\delta-1$ valeurs non maximales de $X_n - X_{n-1}$ sont éliminées par le dernier élément de X_{n-1}. La position de l'élément maximal de $X_n - X_{n-1}$ est alors indifférente ; il est non comparable avec ces $\delta-1$ valeurs.

Exemple : $\delta = 3$, $\mu = 2$



avec seulement : $\alpha_1 \leq \alpha_5$, $\alpha_3 \leq \alpha_5$

+++

On a alors le nombre maximal de comparaisons et donc d'accès :

$$C_{max} = \frac{\delta}{2}(\delta+1+2\mu) \text{ si } \mu \neq 0$$

et

$$C_{max} = \frac{\delta}{2}(\delta+1)-1 \text{ si } \mu = 0.$$

Le cas $\mu = 0$ ne peut se rencontrer qu'au pas 1 de l'algorithme.

$\underline{\rho = \delta}$:

Tous les éléments de $X_n - X_{n-1}$ sont maximaux ; on est dans le cas le plus défavorable où la sélection est ineffective et le coût maximum puisque toutes les comparaisons sont à effectuer.

On a donc :

(j) $C'_{max} = \frac{\delta}{2}(\delta+1+2\mu)$ si $\mu \neq 0$, $\frac{\delta}{2}(\delta+1)-1$ si $\mu = 0$.

On considèrera toujours le coût dans le cas général où $\mu \neq 0$.

2.5.1.f) - Phase finale de l'algorithme

La fonction ARC vérifie l'appartenance à la classe de tous les éléments de $(X_m - X_m) \times (Y_m - Y_m)$; le nombre correspondant d'accès aux enregistrements de X_m et Y_m dépend donc du nombre d'éléments maximaux trouvés, qui est imprévisible. Le seul fait évident est que ce coût diminue quand X_m et Y_m ont un cardinal croissant, pour être nul si $X_m = X_m$ ou $Y_m = Y_m$.
A priori, ce coût varie donc en sens opposé au coût des fonctions MAX et $\overline{MAX}$ qui croît lorsque le nombre d'éléments maximaux augmente.

Nous nous limiterons à étudier le coût de cette fonction dans quelques cas particuliers où l'évaluation est possible. Ces mêmes cas seront repris dans le paragraphe suivant où nous considérons le coût global de l'algorithme de couplage.

Nous considérons toujours une croissance uniforme des couvertures de la classe ; en supposant l'arrêt de l'algorithme de couplage pour la condition $X'_m = \phi$, on a :

$|X_n - X_{n-1}| = \delta_1$, $\forall n \geq 2$, $|X_1| = 1$, soit : $|X_m| = 1+(m-1)\delta_1$

$|Y_n - Y_{n-1}| = \delta_2$, $\forall n \geq 1$, $|Y_1| = \delta_2$, soit $|Y_m| = |Y_{m-1}| = (m-1)\delta_2$

Pour l'évaluation de $|X_m|$ et $|Y_m|$, nous considérons les mêmes cas que pour l'étude de la fonction MAX (on note ρ_n et ρ'_n le nombre d'éléments de X'_n et Y'_n).

$\rho'_n = \rho_n = 1$, $\forall n \geq 1$:

On a alors : $|X_m| = m-1$ et $|Y_m| = |Y_{m-1}| = m-1$.

Le nombre d'accès correspondant à la fonction ARC est alors :

(k) $C_{arc} = [1+(m-1)(\delta_2-1)] \times [1+(m-1)(\delta_1-1)]$

$\rho_n = \delta_1$, $\forall n \geq 2$; $\rho'_n = \delta_2$, $\forall n \geq 1$:

On a alors $X_m = X_m$, $Y_m = Y_m$, et le coût de la fonction ARC est nul.

2.5.1.g) Etude du coût global de l'algorithme

Nous comparons, dans ce qui suit, le coût de l'algorithme 1, présenté en 2.5.1.b), à celui de l'algorithme 2, en tenant compte des divers coûts élémentaires évoqués précédemment.

Cette comparaison théorique est limitée à quelques situations extrêmes particulièrement défavorables à l'algorithme 2.

Nous avons posé précédemment (cf. 2.5.1.b) que c représente le coût moyen de la construction d'une coupe de $\mathcal{P}$ ou de $\bar{\mathcal{P}}$; étant donné $e_m \in E^i$, considérons le coût c_m de la construction de $\mathcal{P}_m \subseteq E^\ell$.

En fonction des performances des techniques d'accès utilisées pour construire effectivement cette coupe, on peut dire que c_m est au moins égal à $|\mathcal{P}_m|$ accès. Si on considère ce coût minimal comme base d'évaluation pour toutes les coupes, et si on considère la construction d'une classe d'après l'algorithme 2, on montre que :

(ℓ) $\delta \leq c$

où δ est l'accroissement moyen des couvertures de la classe à chaque pas de l'algorithme, et c est le coût moyen de la construction des coupes de χ et $\bar{\chi}$ durant cette opération.

Par ailleurs, on a évidemment $\delta \geq 1$; nous restreindrons donc l'étude des différentes expressions de coût données ci-après, au domaine de variation de δ défini par :

(m) $1 \leq \delta \leq c$.

Dans tous les cas d'évaluation qui suivent, on a les hypothèses communes suivantes :

- croissance uniforme des couvertures X_m et Y_m de la classe :

$$\forall n > 1,\ |X_n - X_{n-1}| = \delta$$

$$\forall n \geq 1,\ |Y_n - Y_{n-1}| = \delta$$

- arrêt de l'algorithme 1 au pas m, pour la condition $Y_m = Y_{m-1}$, ce qui correspond à la condition d'arrêt $X'_m = \emptyset$ pour l'algorithme 2.

Cas 1 :

Nous supposons ici que $\delta = 1$; à chaque pas de l'algorithme, les couvertures de la classe s'accroissent d'un seul élément. Par définition de l'algorithme 2, il en résulte que cet élément est obligatoirement maximal, et que :

$$\delta_n = \delta'_n = 1,\ \forall n \geq 1.$$

Considérons l'algorithme 1 ; le coût correspondant pour l'arrêt avec la condition $Y_m = Y_{m-1}$ est donné par (cf. 2.5.1.b) :

$$C_1 = c\, x_m + c \sum_{i=1}^{m-1} (x_i + y_i)$$

avec :

$x_i = |X_i| = i$, $y_i = |Y_i| = i$, et donc :

$$\underline{C_1 = c\, m^2}$$

Considérons l'algorithme 2 ; au pas 1, le coût C (1) est égal à c car seule la coupe donnant Y_1 est construite, et le coût des fonctions MAX et $\overline{MAX}$ est nul car $\delta = 1$ et $\mu = 0$ pour ces deux fonctions.

Au pas n de l'algorithme, on a : $|X_n| = |Y_n| = n$, et $|X_{n-1}| = |Y_{n-1}| = n-1$ car tous les éléments des couvertures sont maximaux. Le coût des fonctions MAX et $\overline{MAX}$ est donc maximal ; il est donné d'après (j) par :

$$2 \times [\tfrac{1}{2}(1+1+2(n-1))] = 2n, \text{ pour } n \geq 2.$$

Par ailleurs, au pas n de l'algorithme on a $\rho'_n = |Y'_{n-1}| = \rho_n = |X'_n| = 1$, et on ne construit donc que deux coupes pour la création de X_n et Y_n. Le coût C(n) du pas n est donc donné par :

$$C(n) = 2c + 2n.$$

Enfin, si l'algorithme s'arrête pour la condition $X'_m = \emptyset$, le coût au pas m est donné par :

$$2+c \leq C(m) \leq m+c$$

car seule la coupe nécessaire à l'obtention de X_m est construite, et le coût de la fonction MAX est compris entre 2 et m selon l'élément de x_{m-1} qui provoque l'élimination de ce dernier élément. Considérons le cas le plus défavorable :

$$C(m) = c+m.$$

Le coût de la fonction ARC est nul dans ce cas, car $Y_m = Y_m$, et on a :

	$\lvert X_n \rvert$	$\lvert Y_n \rvert$	C(n)
pas 1 :	1	1	c
pas 2 :	2	2	2c+2
-----	----	----	----
pas m-1 :	m-1	m-1	2c+2(m-1)
pas m :	m		c+m

$$\text{Soit } C_2 = \sum_{n=1}^{m} c(n) = \underline{m^2 + 2c\,(m-1)}$$

Considérons $\Delta_1 = C_1 - C_2 = m^2(c-1) - 2mc + 2c$.

On montre facilement que pour $c > 2$, cette fonction est toujours positive, donc que l'algorithme 2 est plus performant, quel que soit m, dans cette configuration.

Cas 2 :

Nous supposons ici que $\delta > 1$, et qu'à chaque pas de l'algorithme, un seul élément maximal nouveau est trouvé :

$$\rho_n = \rho'_n = 1, \forall n \geq 1.$$

Considérons l'algorithme 1 :

	$\lvert X_n \rvert$	$\lvert Y_n \rvert$	C(n)
pas 1 :	1	δ	c
pas 2 :	$1+\delta$	2δ	$\delta c+(1+\delta)c$
----	----	----	----
pas m-1 :	$1+(m-2)\delta$	$(m-1)\delta$	$(m-2)\delta c+(1+(m-2)\delta)c$
pas m :	$1+(m-1)\delta$	$m\delta$	$(m-1)\delta c+(1+(m-1)\delta)c$

D'où le coût total au pas m :

$$\underline{C_1 = mc(1+(m-1)\delta)}$$

Considérons l'algorithme 2 ; au pas 1, le coût de la fonction $\overline{MAX}$ est seul à devoir être pris en compte pour la recherche des éléments maximaux ; d'après (j), il est égal à :

$$\frac{\delta}{2}(\delta+1)-1$$

et comme une seule coupe est construite pour obtenir Y_1, on a :

$$C(1) = c + \frac{\delta}{2}(\delta+1)-1.$$

Au pas n de l'algorithme, deux coupes seulement sont à construire car $\rho'_n = \rho_n = 1$; considérons le coût maximal pour les fonctions MAX et $\overline{MAX}$ donné par (j) :

$\frac{1}{2}\,\delta(\delta+1+2(n-1))$ pour chaque fonction

car $\mu = |X_{n-1}| = |Y_{n-1}| = n-1$ au pas n.

On a donc :

$$C(n) = 2c+\delta(\delta+1+2(n-1)).$$

De même que précédemment, lorsque la condition d'arrêt est atteinte au pas m, une seule coupe est construite, et le coût de la fonction MAX est seule à prendre en compte :

$$C(m) = c + \frac{\delta}{2}\,(\delta+1+2(m-1)).$$

On a donc :

	$\lvert X_n\rvert$	$\lvert Y_n\rvert$	C(n)
pas 1 :	1	δ	$c + \frac{\delta}{2}(\delta+1)-1$
pas 2 :	$1+\delta$	2δ	$2c+\delta(\delta+1+2)$
pas 3 :	$1+2\delta$	3δ	$2c+\delta(\delta+1+2\times 2)$
----	----	----	----
pas m-1 :	$1+(m-2)\delta$	$(m-1)\delta$	$2c+\delta(\delta+1+2(m-2))$
pas m :	$1+(m-1)\delta$		$c + \frac{\delta}{2}\,(\delta+1+2(m-1))$

Soit $C_2' = \sum_{n=1}^{m} C(n) = (m-1)(m\delta+\delta^2+2c)-1.$

Dans le cas considéré ici, le coût de la fonction ARC est non nul car $X_m - X_m \neq \emptyset$ et $Y_m - Y_m \neq \emptyset$. Ce coût est donné d'après (k) par :

$$C_{arc} = [1+(m-1)(\delta-1)]^2$$

et il vient :

$$C_2 = C_2' + C_{arc} = (m-1)[m(\delta^2-\delta+1)+4\delta+2c-3].$$

Considérons $\Delta_2 = C_1 - C_2$, on a :

$$C_1 = mc(1+(m-1)\delta) > (m-1)c+m\delta c(m-1) = C_1'$$

et donc :

$$\Delta_2 > C_1'-C_2 = -(m-1)[m(\delta^2-\delta(c+1)+1)+4\delta+c-3] = -(m-1)(mA+B).$$

Posons $y = mA+B$; pour tout $m > 1$, Δ_2 est donc positif pour les valeurs de m rendant y négatif, soit :

$$y < 0 \ \forall m > -\frac{B}{A}, \text{ si } A < 0$$

$$y < 0 \ \forall m < -\frac{B}{A}, \text{ si } A > 0.$$

On montre facilement que A est négatif pour toutes les valeurs de $\delta \in [1,c]$, et par conséquent que y est négatif pour les valeurs de $m > -\frac{B}{A}$.

L'étude de $-\frac{B}{A}$ montre que cette limite devient rapidement inférieure à 1 dès que c prend des valeurs croissantes ; on montre que $-\frac{B}{A}$ demeure inférieure ou égale à 1 pour toute valeur de $\delta \in [1,c-4]$.

Δ_2 est donc positif $\forall m > 1$ lorsque $\delta\in[1,c-4]$, donc pratiquement toujours puisque $1 \leq \delta \leq c$. Rappelons enfin que Δ_2 est évalué par défaut et que l'on a supposé un coût maximal pour MAX et $\overline{MAX}$ à chaque pas de l'algorithme.

Cas 3 :

On suppose ici le cas le plus défavorable pour l'algorithme 2 : $\delta > 1$ et tous les éléments de ces ensembles sont maximaux : $\rho_n = \rho_n' = \delta$, $\forall n$.

Le coût de l'algorithme 1, dans ce cas, est identique à celui évalué dans le cas précédent, soit :

$$C_1 = mc(1+(m-1)\delta).$$

Considérons l'algorithme 2 :

- Au pas 1 de l'algorithme, le coût est identique à celui évalué dans le cas précédent, soit :

$$C(1) = c + \frac{\delta}{2}(\ +1)-1 = c + \frac{\delta}{2}(\delta+1).$$

- Au pas $n \geq 2$ de l'algorithme, on a $|X_{n-1}| = 1+(n-2)\delta = |\mathcal{X}_{n-1}|$, et le coût de MAX est donc donné par :

$$\frac{\delta}{2}(\delta+1+2(1+(n-2)\delta)) = \frac{\delta}{2}(\delta+3+2(n-2)\delta)$$

On a également $|\mathcal{Y}_{n-1}| = (n-1)\delta = |Y_{n-1}|$, et le coût de la fonction $\overline{MAX}$ est donné par :

$$\frac{\delta}{2}(\delta+1+2(n-1)\delta).$$

Soit un coût global pour la recherche des éléments maximaux :

$$\delta(\delta+2+\delta(2n-3)).$$

- Au pas n, 2δ coupes sont construites pour l'obtention de X_n et Y_n, puisque tous les éléments sont maximaux ; on a donc :

$$C(n) = 2\delta c + \delta(\delta+2+\delta(2n-3)) = 2\delta c + \delta(2+\delta(2n-2)).$$

L'arrêt de l'algorithme au pas m pour la condition $\mathcal{X}'_m = \emptyset$ implique que seul X_m est construit, soit un coût de δc pour les coupes. Les δ éléments de $X_m - X_{m-1}$ n'étant pas maximaux sont éliminés par la fonction MAX ; on considère que ceci est effectué au coût maximum, et on a donc, puisque $|\mathcal{X}_{m-1}| = 1+(m-2)\delta$:

$$C(m) = \delta c + \frac{\delta}{2}(\delta+3+2(m-2)\delta).$$

Enfin, comme dans le cas 1, le coût de la fonction ARC est nul puisque $\mathcal{Y}_m = Y_m$.

On a donc :

	$\|X_n\|$	$\|Y_n\|$	C(n)
pas 1 :	1	δ	$c + \frac{\delta}{2}(\delta+1)$
pas 2 :	$1+\delta$	2δ	$2\delta c+\delta(2+2\delta)$
pas 3 :	$1+2\delta$	3δ	$2\delta c+\delta(2+4\delta)$
----	----	----	----
pas m-1 :	$1+(m-2)\delta$	$(m-1)\delta$	$2\delta c+\delta(2+2(m-2)\delta)$
pas m :	$1+(m-1)\delta$		$\delta c+ \frac{\delta}{2} (\delta+3+2(m-2)\delta)$

Soit $C_2 = \sum_{n=1}^{m} C(n) = (m-1)\ [\delta^2(m-1)+2\delta(c+1)]-\delta c+c$

et $\Delta_3 = C_2-C_1 = (m-1)\ [m\delta(c-\delta)+\delta^2-2\delta(c+1)+c]+\delta c$

soit :

$$\Delta_3 = m^2\delta(c-\delta)+m\ [2\delta^2-\delta(3c+2)+c] - [\delta^2-\delta(3c+2)+c] = m^2A+mB+C.$$

L'étude de cette famille de fonctions, dans les conditions données par :

$$c \geq 1$$

et

$$1 \leq \delta < c$$

montre que Δ_3 est positif quel que soit m si $1 \leq \delta \leq \frac{9c}{16}$ car dans ce cas, l'équation $\Delta_3 = 0$ n'admet pas de racine réelle.

Pour les valeurs supérieures de δ, il existe un intervalle de valeurs de m pour lequel Δ_3 est négatif ; l'algorithme 2 devient moins performant que l'algorithme 1 pour les grandes valeurs de δ.

Les configurations de données présentées ci-dessus (surtout les cas 1 et 3) sont très défavorables à l'algorithme 2 : coût maximal des fonctions MAX et $\overline{MAX}$ et, sauf pour le cas 2, non diminution du nombre de coupes à construire. Malgré cela, on voit que le coût de l'algorithme 2, exprimé en nombre d'accès aux éléments de la base, demeure inférieur à celui de l'algorithme 1 jusqu'à des valeurs élevées de δ.

2.5.2. Généralisation de l'algorithme de couplage

2.5.2.a) Définition - Notations

Nous considérons ici la généralisation de l'algorithme de couplage au traitement simultané de N fichiers $E^1,\ldots,E^N$. Un tel traitement est défini à partir d'un maximum de C_2^N relations de cohérence large notées $R^{i\ell} \subseteq \pi^i \times \pi^\ell$, auxquelles correspondent les relations entre enregistrements notées $\gamma^{i\ell} \subseteq E^i \times E^\ell$.

Considérons l'union des graphes bipartis correspondant à chacune de ces relations ; une classe de cohérence de $E = E^1+\ldots+E^N$ est définie comme une composante connexe de ce graphe. Elle comporte donc N couvertures notées $X^i \subseteq E^i$, et un ensemble d'arêtes $G = \cup G^{i\ell}$, où $G^{i\ell}$ est le sous-ensemble des arêtes du graphe de $\gamma^{i\ell}$, dont les extrémités appartiennent à X^i et X^ℓ.

A chacune des relations $R^{i\ell}$ correspond un couple de prédicats $\text{Réc}^{i\ell}$ et $\overline{\text{Réc}}^{i\ell}$ défini selon 2.4.2.h, et un couple de fonctions $\text{MAX}^{i\ell}$, $\overline{\text{MAX}}^{i\ell}$ de recherche des éléments maximaux relativement à l'ordre partiel $\preceq^{i\ell}$ induit par ces prédicats.

Comme précédemment, on note :

$$X_m^{i\ell} = \text{MAX}^{i\ell}((X_n^i - X_{n-1}^i),\ X_{n-1}^{i\ell}) \text{ pour } i < \ell \ ;\ i,\ell \in [1,N]$$

l'ensemble des éléments maximaux contenus dans X^i au pas n de l'algorithme, relativement à l'ordre $\preceq^{i\ell}$ induit par $\text{Réc}^{i\ell}$.

On note de même :

$$X_n^{i\ell} = \overline{\text{MAX}}^{i\ell}((X_n^\ell - X_{n-1}^\ell),\ X_{n-1}^{i\ell}) \text{ pour } i > \ell \ ;\ i,\ell \in [1,N]$$

l'ensemble des éléments maximaux contenus dans X^ℓ au pas n de l'algorithme relativement à l'ordre induit par $\overline{\text{Réc}}^{i\ell}$.

Comme précédemment, l'expression :

$$X'^{i\ell}_n = X_n^{i\ell} - X_{n-1}^{i\ell}$$

représente l'ensemble des éléments maximaux nouveaux obtenus dans X^i pour $\preceq^{i\ell}$, au pas n de l'algorithme.

Les applications précédemment notées * et + correspondant à la construction des coupes de $\mathcal{R}$ et $\bar{\mathcal{R}}$ sont à présent notées $x^{i\ell}$ avec la convention suivante :

(a) si $i > \ell$, $x^{i\ell} = *^{i\ell}$: construction des coupes de la relation directe $R^{i\ell}$

si $i < \ell$, $x^{i\ell} = +^{i\ell}$: construction des coupes de la relation inverse $\bar{R}^{i\ell}$.

Enfin, les ensembles d'arêtes obtenus lors de la construction des coupes ci-dessus sont notées (cf. définition de l'algorithme 2) :

(a') si $i > \ell$, $F^{i\ell} = F'^{i\ell}$, ensemble des arêtes obtenues par la relation directe $R^{i\ell}$

si $i < \ell$, $F^{i\ell} = \bar{F}'^{i\ell}$, ensemble des arêtes obtenues par la relation inverse $\bar{R}^{i\ell}$.

2.5.2.b) Définition récurrente de l'algorithme

L'algorithme de couplage de N fichiers est déduit de la définition de l'algorithme 2 ; partant d'un enregistrement e_d, chaque pas de l'algorithme est à présent constitué de N-1 sous-pas correspondant à la construction des coupes de X^k dans les N-1 autres fichiers notés X^ℓ.

Au pas 1 de l'algorithme, k est déterminé par le fichier E^i auquel appartient l'enregistrement de départ e_d : $k = i \in [1,N]$, et $X^k = \{e_d\}$ est la seule couverture non vide.

Aux pas suivants, k prend les valeurs successives k+1... de [1,N] selon une permutation circulaire définie au pas n par la fonction suivante :

(b) $k = (n+i-2) \text{ MOD}(N)+1 = f(n,i,N)$.

Exemple :

Si l'enregistrement de départ appartient à E^2 et que le processus de couplage est appliqué à N=4 fichiers, k prend les valeurs successives ci-dessous en fonction de n :

n =	1	2	3	4	5	6	7	...
k =	2	3	4	1	2	3	4	...

et on construira, dans l'ordre, les coupes de $X^2 \in E^2$, $X^3 \in E^3$, $X^4 \in E^4$, $X^1 \in E^1$, $X^2 \in E^2$, ...

+++

Au pas n, k étant donné par (b), on définit N-1 sous-pas correspondant à la construction des coupes de X^k dans les fichiers E^ℓ, où $\ell \in \{1,\ldots,N\}-\{k\}$. L'ordre selon lequel les coupes $X^\ell \subseteq E^\ell$ sont ainsi construites est indifférent ; comme pour les valeurs de k, nous considérons une fonction g d'énumération des valeurs de ℓ au sous-pas $m \in [1,N-1]$ définie par :

(c) $\ell = (k+m-1) \text{ MOD}(N)+1 = g(m,k,N)$.

Exemple : Pour k=3, N=4, ℓ prend les valeurs successives suivantes lors des N-1=3 sous-pas :

$m = 1, 2, 3$

$\ell = 4, 1, 2.$

Considérant $X^3 \subseteq E^3$, on construit donc successivement les coupes de X^3 dans E^4, E^1, E^2, soit X^4, X^1, X^2.

+++

Si on note comme précédemment G_n l'ensemble des arêtes de la classe obtenues au pas n de l'algorithme, et G_n^m ce même ensemble au sous-pas m, on a l'expression ci-contre de l'algorithme généralisé.

Comme précédemment, l'algorithme généralisé ne donne qu'un sous-ensemble G de l'ensemble des arêtes de la classe ; il doit être complété par une fonction ARC généralisée aux N fichiers, recherchant l'existence d'arêtes reliant les couples d'enregistrements appartenant aux ensembles produits :

(d) $(X^k - X^{k\ell}) \times (X^\ell - X^{\ell k})$, $\forall k,\ell \in [1,N]$, $k \neq \ell$.

Enfin, comme dans l'algorithme 2, la condition d'arrêt est atteinte lorsqu'il n'existe plus de couverture présentant des éléments maximaux nouveaux, soit :

(e) $\forall k,\ell \in [1,N]$, $k \neq \ell$, $X'^{k\ell} = \emptyset$.

Initialisation : $X_0^i = \{e_d\}$, $X_0^k = \emptyset$, $\forall k \in [1,N]$, $k \neq i$

$X_0^{k\ell} = \emptyset$, $\forall k,\ell \in [1,N]$, $k \neq \ell$

$G_0 = \emptyset$.

Pas 1 : $k = f(1,i,N) = i$

sous-pas 1 : $m=1$; $\ell = g(m,k,N)$

$X'^{k\ell}_1 = X_1^{k\ell} - X_0^{k\ell} = \{e_d\}$; $X_1^\ell = X_0^\ell \cup (X'^{k\ell})^{\times k\ell}$; $G_1^1 = G_0 \cup F^{k\ell}$

---

sous-pas N-1 : $m=N-1$; $\ell=g(m,k,N)$

$X'^{k\ell}_1 = X_1^{k\ell} - X_0^{k\ell} = \{e_d\}$; $X_1^\ell = X_0^\ell \cup (X'^{k\ell})^{\times k\ell}$; $G_1^{N-1} = G_1^{N-2} \cup F^{k\ell}$

Pas 2 : $k = f(2,i,N)$

sous-pas 1 : $m=1$; $\ell = g(m,k,N)$

$X'^{k\ell}_2 = X_2^{k\ell} - X_1^{k\ell}$; $X_2^\ell = X_1^\ell \cup (X'^{k\ell})^{\times k\ell}$; $G_2^1 = G_1^{N-1} \cup F^{k\ell}$

---

sous-pas N-1 : $m=N-1$; $\ell=g(m,k,N)$

$X'^{k\ell}_2 = X_2^{k\ell} - X_1^{k\ell}$; $X_2^\ell = X_1^\ell \cup (X'^{k\ell})^{\times k\ell}$; $G_2^{N-1} = G_2^{N-2} \cup F^{k\ell}$

Pas n : $k = f(n,i,N)$

sous-pas 1 : $m=1$; $\ell=g(m,k,N)$

$X'^{k\ell}_n = X_n^{k\ell} - X_{n-1}^{k\ell}$; $X_n^\ell = X_{n-1}^\ell \cup (X'^{k\ell})^{\times k\ell}$; $G_n^1 = G_{n-1}^{N-1} \cup F^{k\ell}$

---

sous-pas N-1 : $m=N-1$; $\ell=g(m,k,N)$

$X'^{k\ell}_n = X_n^{k\ell} - X_{n-1}^{k\ell}$; $X_n^\ell = X_{n-1}^\ell \cup (X'^{k\ell})^{\times k\ell}$; $G_n^{N-1} = G_n^{N-2} \cup F^{k\ell}$

2.5.2.c) Cas particulier - Relations entre caractéristiques synonymes

Nous considérons tout d'abord le cas particulier où une relation R^i est entièrement définie (cf. 2.4.2.b') à partir de relations R_j entre caractéristiques synonymes ; $\chi^{i\ell}$ peut alors être considéré comme une relation sur $E \times E$, avec $E = E^i + E^\ell$.

Une propriété de nature à simplifier l'algorithme de couplage est la symétrie de $R^{i\ell}$; on a alors $R^{i\ell} = \bar{R}^{i\ell}$ et, par conséquent, identité des applications $*^{i\ell}$ et $+^{i\ell}$, de même qu'il y a identité de définition entre les prédicats $\text{Réc}^{i\ell}$ et $\bar{\text{Réc}}^{i\ell}$, et les fonctions $\text{MAX}^{i\ell}$ et $\overline{\text{MAX}}^{i\ell}$ correspondantes.

Par définition de $R^{i\ell}$, et d'après la propriété 5 de 2.4.3, une condition suffisante pour que $R^{i\ell}$ soit symétrique est que toutes les relations R_j de départ, entrant dans sa définition, soient symétriques.

Les caractéristiques invariantes (cf. 2.1.3.d) constituent un cas particulier de caractéristiques synonymes ; les relations externes primitives R_j définies sur ce type de caractéristique étant l'égalité, les relations étendues correspondantes R_j définies selon 2.4.3.b' sont symétriques et réflexives (cf. propriété 3 de 2.4.3).

Dans le cas où une relation de cohérence $R^{i\ell}$ comporte dans sa définition une ou plusieurs relations entre caractéristiques invariantes, il est intéressant de considérer la définition suivante de cette relation :

$$R^{i\ell} = R'^{i\ell} \cap R''^{i\ell}$$

où $R'^{i\ell}$ regroupe toutes les relations définies entre caractéristiques invariantes, et $R''^{i\ell}$ les autres relations entrant dans la définition de $R^{i\ell}$.

Cette décomposition permet en particulier :

- de faciliter la construction des coupes de $\chi^{i\ell}$; dans la mesure où il est généralement plus aisé de définir des techniques d'accès rapide relativement à des relations simples (hash-code), $R'^{i\ell}$ permet une première sélection performante des éléments de la coupe, $R''^{i\ell}$ définissant une seconde sélection dans cet ensemble ;

- de réutiliser les mêmes fonctions d'accès rapide lorsque plusieurs relations R^{il} possèdent dans leur définition une même relation R_j entre caractéristiques invariantes.

2.5.3. Conclusion

L'algorithme proposé permet de construire des classes de cohérence dans un ensemble de N fichiers donnés, à partir de la spécification de l'ensemble des relations utiles pour chaque couple de fichier considéré.

Le processus proposé minimise effectivement le nombre de constructions des coupes aux éléments maximaux de chaque demi-treillis considéré ; la sélection de ces éléments maximaux par les prédicats *Rec* et $\overline{Rec}$ plutôt que par les prédicats *REC* et $\overline{REC}$ (cf. 2.5.2) constitue une approximation destinée à diminuer le coût de cette recherche, avec l'inconvénient mentionné plus haut de ne pas sélectionner toujours uniquement ces éléments maximaux. En ce sens, cette opération de sélection des éléments paraissant les plus indiqués pour une convergence rapide vers la solution peut être assimilée à une heuristique de sélection du chemin de coût minimal.

Un autre point fondamental dans la diminution du coût de l'algorithme est constitué par la construction des coupes des diverses relations de cohérence ; on peut résoudre ce problème par la sélection de critères d'accès les plus efficaces possible dans la base. Cet aspect est abordé dans le chapitre suivant où divers critères sont proposés pour l'évaluation du pouvoir discriminant d'un ensemble de caractéristiques.

CHAPITRE 2.6. Mesures dans le système d'informations

2.6. Mesures dans le système d'information

2.6.1. Introduction

Nous présentons dans ce chapitre divers outils d'évaluation destinés à résoudre deux problèmes précédemment évoqués :

- pour améliorer les performances de l'algorithme de couplage, on cherche à diminuer le coût moyen c de construction des coupes ; ceci peut être facilité par la définition de méthodes d'accès rapide relativement à des caractéristiques à fort pouvoir discriminant (cf. 2.5.2.c). Des critères d'évaluation du pouvoir discriminant d'un ensemble de caractéristiques sont définis en 2.6.2 ; ils permettent d'orienter la recherche d'un ensemble de critères d'accès efficace pour une application particulière, et d'évaluer l'incidence des modèles de variation (cf. 2.4.2.a) sur les performances du système (analyse de la perte d'information correspondant à ces modèles) ;
- une fois les classes de cohérence construites, il reste à lever les ambiguïtés qui se manifestent par des possibilités mutuellement exclusives dans chaque classe ; une méthode largement employée consiste à pondérer ces différentes possibilités et à sélectionner celles qui réalisent le meilleur score (cf. 2.4.4). Le chapitre 2.6.3 présente la définition d'un ensemble de critères de ressemblance permettant d'effectuer ces choix (phase de décision qui sera développée au chapitre 2.7).

La définition de l'ensemble de ces critères a pour fondement la théorie de l'information [B.ACD] ; l'utilisation qui en a été faite par C. PICARD dans la théorie des questionnaires [B.PIC] a servi de point de départ à la définition des critères d'évaluation du pouvoir discriminant. Plus tard, il est apparu que ce même outil théorique pouvait permettre la définition de critères de ressemblance fondés sur la notion de quantité d'information cohérente. Dans le cas général qui nous concerne ici, il n'est pas toujours simple d'évaluer exactement les divers critères proposés (grand volume d'information), aussi proposons-nous, chaque fois que cela est possible, des critères approximatifs dont l'évaluation est beaucoup plus simple.

Le choix de l'utilisation des critères théoriques ou approximatifs dépend surtout de la taille de l'application et du degré de précision recherché ; comme on le verra par la suite, les critères approximatifs permettent généralement des présélections à moindre coût ; les évaluations exactes ne sont opérées que sur les ensembles restreints d'informations ainsi déterminés.

2.6.2. Evaluation du pouvoir discriminant

Le pouvoir discriminant d'une caractéristique ou d'un ensemble de caractéristiques est défini ici comme une mesure dérivée de la notion d'entropie en théorie de l'information. Le critère proposé est un nombre réel compris entre 0 et 1 selon que la caractéristique n'est absolument pas discriminante (une seule valeur) ou discriminante (elle possède autant de valeurs qu'il y a d'enregistrements dans le fichier sur lequel elle est définie).

Les valeurs intermédiaires obtenues tiennent compte à la fois du nombre de valeurs de la caractéristique et de leur répartition dans le fichier.

2.6.2.a) Quantité d'information liée à une caractéristique

Etant donné une caractéristique C définie sur un fichier E de N éléments, et son ensemble V de M valeurs, considérons les fréquences f_v de chacune de ces valeurs dans le fichier ; soit :

$$f_v = \frac{n_v}{N} \text{ , où } n_v = |E_v|$$

Si nous considérons que ces quantités peuvent être assimilées à des probabilités (grand nombre de consultations aléatoires du fichier), nous pouvons utiliser la mesure de WIENER pour évaluer la quantité d'information donnée par la connaissance de chaque valeur de C [B.ACD].

(a) $\quad I(v) = -\log_2 p_v$, avec $0 < p_v \leq 1$.

La caractéristique inverse $\bar{C}$ déterminant une partition de E en M sous-ensembles, on peut considérer la connaissance de la valeur de C pour un

enregistrement de E comme une expérience à M éventualités discrètes, formant un système complet :

$$\sum_{v \in V} p_v = 1.$$

La définition de SHANNON de l'entropie [B.ACD] permet alors d'évaluer la quantité moyenne d'information liée à la connaissance d'une valeur de C :

(b) $$H(C) = \sum_{v \in V} - p_v \log_2 p_v.$$

Rappelons quelques propriétés fondamentales de la mesure de SHANNON :

(c) $H(C) \geq 0.$

(d) Si C est considéré comme une expérience à M issues, la valeur maximale de H(C) est obtenue si et seulement si les M éventualités sont équiprobables :

$$\forall v,\ p_v = \frac{1}{M}$$

la valeur maximale de H(C) est alors donnée par $H(C) = \log_2 M.$

Le choix de la base 2 pour les logarithmes est une convention fixant l'unité de mesure de l'entropie ; H(C) = 1 pour un système à deux éventualités équiprobables de probabilité 0.5. Nous écrirons désormais log pour alléger les notations.

Exemple : Considérons un fichier E, deux caractéristiques C_1 et C_2 définies sur E, ainsi que leurs distributions de valeurs données ci-dessous :

$C_1 : V_1 = \{a,b,c,d,e\}$ avec $p_a = p_b = p_c = p_d = p_e = 0.2$

$C_2 : V_2 = \{v,w,x,y,z\}$ avec $p_v = p_w = 0.2$, $p_x = 0.4$, $p_y = p_z = 0.1$

On obtient d'après (b) :

$$H(C_1) \log 5 = 2.232$$
$$H(C_2) = 2.122.$$

Les deux caractéristiques ont cinq valeurs ; C_1 donne le meilleur résultat car toutes ses valeurs sont équiprobables. Considérons à présent une caractéristique C_3 définie sur E et possédant quatre valeurs équiprobables :

C_3 : V_3 = {m,n,o,p} avec $p_m = p_n = p_o = p_p = 0.25$.

On a :

$$H(C_3) = \log 4 = 2.$$

L'entropie de C_3 est plus faible que celle de C_1 car elle possède moins de valeurs.

+++

2.6.2.b) Pouvoir discriminant d'une caractéristique

Une caractéristique C définie sur un fichier E de N éléments est discriminante si et seulement si elle possède N valeurs ; ces valeurs sont donc équiprobables avec la probabilité 1/N, et d'après (d), H(C) prend alors la valeur :

$$H(C) = \log N.$$

Nous définissons le pouvoir discriminant d'une caractéristique C relativement à son ensemble de définition E de N éléments par :

(e) $$P(C) = \frac{H(C)}{\log N}$$

avec $0 \leq P(C) \leq 1$.

La valeur maximale 1 n'est atteinte que si la caractéristique est discriminante; la valeur nulle n'est atteinte que pour le cas limite où la caractéristique ne possède qu'une seule valeur (de probabilité 1).

Exemple : Si nous reprenons les caractéristiques de l'exemple précédent, avec N = 1000, nous obtenons les valeurs suivantes du pouvoir discriminant :

$P(C_1) = 0.233$

$P(C_2) = 0.213$

$P(C_3) = 0.201$.

D'où il ressort que C_1 est la caractéristique la plus efficace pour la discrimination des éléments de E.

+++

Remarque : L'évaluation du pouvoir discriminant donnée par (e) est relative à un ensemble particulier ; par conséquent, $P(C)$ ne peut généralement pas être considérée comme une propriété intrinsèque de C, utilisable pour tout fichier comportant cette caractéristique dans sa description (caractéristiques synonymes). Dans ce cas, nous considérons la valeur moyenne de $H(C)$ dans ces différents fichiers.

2.6.2.c) Distribution des valeurs d'une caractéristique

Des considérations précédentes, on peut déduire la définition d'un critère caractérisant la distribution des valeurs d'une caractéristique :

(f) $$\mathcal{D}(C) = \frac{H(C)}{\log M}$$

avec

$0 < \mathcal{D}(C) \leq 1$, $M > 1$.

La valeur maximale 1 n'est évidemment atteinte que si la répartition des M valeurs de C est équiprobable (probabilité 1/M).

Exemple : Pour les caractéristiques de l'exemple précédent, on obtient :

$\mathcal{D}(C_1) = 1$

$\mathcal{D}(C_2) = 0{,}914$

$\mathcal{D}(C_3) = 1$.

On retrouve le fait que seules C_1 et C_3 ont une répartition équiprobable.

+++

D'après les définitions (e) et (f), les valeurs de $P(C)$ et $\mathcal{D}(C)$ sont liées :

(g)
- $P(C)$ et $\mathcal{D}(C)$ sont simultanément égaux à 1 si et seulement si C est discriminante.
- $P(C) = 1 \Rightarrow \mathcal{D}(C) = 1$.
- $\mathcal{D}(C) = 1 \Rightarrow P(C)$ prend sa valeur maximale log M / log N.

Pour juger de l'efficacité d'une caractéristique C en tant que critère d'accès au fichier E sur lequel elle est définie, il est préférable de considérer simultanément les valeurs de $P(C)$ et $\mathcal{D}(C)$; entre deux caractéristiques présentant des pouvoirs discriminants voisins, la plus efficace, en tant que critère d'accès, est celle qui présente la distribution de valeurs la plus uniforme (la plus forte valeur de $\mathcal{D}$). En effet, quelle que soit la technique d'accès rapide considérée pour une caractéristique (hash-code, fichier inverse, ...), les performances obtenues (place, temps d'accès) sont généralement meilleures avec des valeurs uniformément réparties.

A l'opposé, les faibles valeurs de $\mathcal{D}$ (et de P) sont obtenues pour des distributions fortement déséquilibrées des valeurs : valeurs à très forte ou très faible probabilités.

2.6.2.d) Quantité d'information liée à un ensemble de caractéristiques

Pour un ensemble D de n caractéristiques fichier définies sur E, on peut définir la quantité moyenne d'information liée à tout n-uplet de l'ensemble produit $\pi = V_1 \times \ldots \times V_n$.

Rappelons les propriétés suivantes de l'entropie relative à deux expériences notées A et B [B.TER]:

(h) A et B étant indépendantes, on a :
$H(A \cap B) = H(A) + H(B)$, où $H(A \cap B)$ mesure l'information moyenne obtenue pour les deux expériences considérées simultanément.

(i) A et B n'étant pas indépendantes, on a :
$H(A \cap B) = H(A) + H(B|A) = H(B) + H(A|B)$, où $H(B|A)$ est l'entropie de l'expérience B connaissant le résultat de l'expérience A.

On a, de plus, les inégalités suivantes :

(j) $H(A \cap B) \leq H(A) + H(B)$

(k) $H(A|B) \leq H(A)$

(l) $H(A|B) \geq 0$

(m) Les propriétés ci-dessus peuvent être généralisées à n expériences.

La quantité $H(A|B)$ est aussi appelée entropie de A conditionnée par B ; elle est définie par l'expression suivante pour un système complet :

(n) $\quad H(A\ B) = \sum_{b\in B} p_b\ H_b(A)$, avec $H_b(A) = \sum_{a\in A} -p(a|b)\ \log\ p(a|b)$

où $p(a|b) = \dfrac{p(a \cap b)}{p_b}$

L'expression $H(C_1 \cap C_2)$ peut être interprétée comme la quantité moyenne d'information apportée par les expériences combinées C_1 et C_2 ; d'après (i), elle est égale à l'entropie de C_1 augmentée de la quantité moyenne d'information donnée par C_2, le résultat de C_1 étant connu.

Dans les applications mettant en oeuvre de très gros fichiers, il peut être assez coûteux d'évaluer l'expression donnée par (n) du fait de la multitude de probabilités conditionnelles à considérer : les caractéristiques peuvent avoir des centaines ou des milliers de valeurs, et le nombre de combinaisons à évaluer peut être très grand.

Nous proposons ici deux évaluations du pouvoir discriminant pour un ensemble de caractéristiques : la première donne une approximation aisément calculable qui suppose que les caractéristiques sont indépendantes, la seconde donne la valeur exacte d'après la définition (i) de l'entropie :

(o) $\quad P'(C_1,C_2) = \dfrac{H(C_1)+H(C_2)}{\log N} = P(C_1)+P(C_2)$

(p) $\quad P(C_1,C_2) = \dfrac{H(C_1 \cap C_2)}{\log_2 N}$

On montre [B.CH1] les propriétés suivantes relatives à P et à P'.

Propriété 1 :

$0 \leq P(C_1,C_2) \leq 1$

La valeur maximale 1 n'est atteinte que si (C_1,C_2) est discriminant pour E.

Propriété 2 :

$P(C_1,C_2)$ satisfait la double inégalité suivante :
$\text{MAX}(P(C_1),\ P(C_2)) \leq P(C_1,C_2) \leq P'(C_1,C_2)$.

L'erreur systématique commise en considérant P' plutôt que P est égale à :

$$e = H(C_1) + H(C_2) - H(C_1 \cap C_2)$$

D'après (i), cette quantité peut s'écrire :

(q) $$e = H(C_1) - H(C_1|C_2) = H(C_2) - H(C_2|C_1).$$

Cette quantité est connue en théorie de l'information comme l'information mutuelle entre C_1 et C_2 ; on la note couramment $I(C_1,C_2)$. On a [B.TER] les propriétés suivantes de I :

(r) $$I(C_1,C_2) = I(C_2,C_1)$$

$$I(C_1,C_2) \geq 0$$

$I(C_1,C_2) = 0$ si et seulement si C_1 et C_2 sont indépendantes.

Nous verrons ultérieurement l'utilité de cette mesure pour l'évaluation de la redondance entre caractéristiques.

Propriété 3 :

Une condition nécessaire pour que (C_1,C_2) soit discriminant est que la valeur de $P'(C_1,C_2)$ soit supérieure ou égale à 1.

La démonstration de cette propriété est immédiate [B.CH1]; nous n'avons pas trouvé de condition nécessaire et suffisante relative à P' pour qu'un système de caractéristiques soit discriminant. S'il est pratiquement certain qu'une telle condition n'existe pas, il semble possible de rattacher l'expression de P' à une probabilité pour que l'ensemble de caractéristiques soit discriminant, mais nous n'avons pas pour l'instant de résultat significatif à ce niveau. Si une telle corrélation peut être établie, la signification du critère P' serait évidemment grandement améliorée ; dans l'état actuel des choses, le fait qu'un ensemble de caractéristiques vérifie la propriété 3 indique seulement qu'il existe une probabilité non nulle pour que ce groupe soit discriminant. En fait, ce qui est à analyser ici est l'interdépendance des diverses caractéristiques considérées.

Exemple : Considérons les deux caractéristiques C_1 et C_2 définies ci-dessous avec leur répartition de valeurs sur le même fichier E :

C_1 : V_1 = {a,b} avec $p_a = p_b = 0.5$

C_2 : V_2 = {v,w,x,y,z} avec $p_v = p_z = 0.125$, $p_w = p_x = p_y = 0.25$

Les deux répartitions données ci-dessous respectent ces contraintes, mais dans les deux cas, l'interdépendance entre ces répartitions n'est pas la même.

Cas 1 : C_1 et C_2 ne constituent pas un ensemble discriminant sur E :

. (a,x) . (a,x) . (a,y) . (a,y)
. (b,z) . (b,w) . (b,w) . (b,v)

E (N = 8)

On a :
$H(C_1) = 1$, $H(C_2) = 2.25$
$H(C_1)+H(C_2) = 3.25$
$P'(C_1,C_2) = 1.08$
$H(C_1 \cap C_2) = 2.25$
$P(C_1,C_2) = 0.75$
$I(C_1,C_2) = 1$

Cas 2 : C_1 et C_2 constituent un ensemble discriminant sur E :

. (a,x) . (a,y) . (a,w) . (a,z)
. (b,x) . (b,y) . (b,w) . (b,v)

E (N = 8)

On a toujours :
$H(C_1) = 1$, $H(C_2) = 2.25$
et $P'(C_1,C_2) = 1.08$
mais
$H(C_1 \cap C_2) = 3$ et $P(C_1,C_2) = 1$
avec $I(C_1,C_2) = 0.25$

Remarquons que dans le second cas, l'interdépendance entre C_1 et C_2 mesurée par I est plus faible que dans le premier cas.

+++

L'ensemble de ces définitions et propriétés peuvent être étendues à tout groupe D de p caractéristiques ; le pouvoir discriminant est alors définie par :

$$(s) \qquad P(C_1,C_2,\ldots,C_p) = \frac{H(C_1 \cap C_2 \cap \ldots \cap C_p)}{\log N}, \text{ avec } 0 \leq P'(C_1,C_2,\ldots,C_p) \leq 1$$

et la valeur approchée est définie par :

$$(t) \qquad P'(C_1,C_2,\ldots,C_p) = \frac{\sum_{i\in D} H(C_i)}{\log N} = \sum_{i\in D} P(C_i)$$

avec :

$$0 \leq P'(C_1,C_2,\ldots,C_p) \leq p$$

2.6.2.e) Evaluation de la redondance dans un ensemble de caractéristiques

La définition (q) de l'information mutuelle entre deux caractéristiques peut être utilisée pour évaluer la redondance existant entre ces deux informations ; là encore, nous pouvons considérer une valeur exacte et une valeur approchée plus aisément calculable.

Nous définissons la redondance entre C_1 et C_2 définie sur E par :

$$(u) \qquad R(C_1,C_2) = \frac{I(C_1,C_2)}{\log N} = P'(C_1,C_2) - P(C_1,C_2)$$

Cette quantité mesure la perte de pouvoir discriminant liée à la dépendance des répartitions des valeurs de C_1 et C_2.

On démontre facilement la propriété suivante :

Propriété 4 :

$$0 \leq R(C_1,C_2) \leq 1$$

La valeur minimale est atteinte lorsque C_1 et C_2 sont indépendantes, la valeur maximale est atteinte lorsque $\overline{C_1}$ et $\overline{C_2}$ déterminent une même partition de E.

Nous proposons le critère suivant pour l'évaluation approximative de R :

(v) $$R'(C_1,C_2) = \frac{H(C_1)+H(C_2)-\log N}{\log N} = P'(C_1,C_2)-1$$

On démontre que :

Propriété 5 :

R' est une approximation par défaut de R :

$$R'(C_1,C_2) \leq R(C_1,C_2)$$

et :

$$-1 \leq R'(C_1,C_2) \leq 1$$

La définition de ces critères peut être étendue à tout ensemble D de p caractéristiques ; on a alors :

(w) $$R(C_1,C_2,\ldots,C_p) = \frac{I(C_1,C_2,\ldots,C_p)}{\log N} = P'(C_1,C_2,\ldots,C_o)-P(C_1,C_2,\ldots,C_p)$$

avec :

$$0 \leq R(C_1,C_2,\ldots,C_p) \leq p-1$$

(x) $$R'(C_1,C_2,\ldots,C_p) = \frac{\sum_{C_i \in D} H(C_i)-\log N}{\log N} = P'(C_1,C_2,\ldots,C_p)-1$$

avec :

$$-1 \leq R'(C_1,C_2,\ldots,C_p) \leq p-1$$

2.6.2.f) Sélection des caractéristiques les plus efficaces

Etant donné un fichier E, sa description D, la sélection dans D des caractéristiques les plus efficaces pour la discrimination des éléments de E peut être envisagée de la façon suivante, compte tenu de ce qui précède :

a) Calcul du pouvoir discriminant $P(C_i)$ de tous les éléments de D ; ceci peut être effectué en un seul parcours de E en gérant des dictionnaires de valeurs des différentes caractéristiques.

b) Classifier les éléments de D par ordre décroissant de leur pouvoir discriminant et de leur critère de répartition $D(C_i)$.

c) Si aucune caractéristique n'a un pouvoir discriminant jugé suffisant, considérer des groupes de deux caractéristiques présentant à la fois un bon pouvoir discriminant et une bonne répartition de leurs valeurs. L'ensemble considéré est d'autant plus efficace que la redondance entre les caractéristiques étudiées est plus faible.

d) Si cela est nécessaire, on envisage des groupes de 3, ou plus, caractéristiques de D, en conservant les mêmes critères qu'en c).

Dans ce processus de sélection, l'usage des différents critères proposés plus haut est évident ; les valeurs approximatives guident les premiers choix, et les valeurs réelles ne sont calculées qu'à la demande, sur un nombre limité de cas, et seulement si ce degré de précision se justifie dans l'application considérée.

La recherche d'ensembles de caractéristiques de plus petit cardinal possible donnant la meilleure discrimination d'un fichier rejoint l'utilisation qui a été faite de la théorie de l'information dans la théorie des questionnaires ([B.PIC] et [B.TER]).

2.6.3. Proposition d'un critère de ressemblance:

2.6.3.a - Introduction:

Une classe de cohérence étant construite, il reste à sélectionner dans le graphe correspondant l'ensemble des couples d'enregistrements homologues. Les algorithmes présentés au chapitre 2.7 effectuent cette opération à partir d'une évaluation du degré de ressemblance de tous les couples d'enregistrements constituant les extrémités d'un arc de la classe (poids de l'arc). De nombreuses mesures de ce type ont été proposées en classification automatique ([B.DUC] et [B.LER]) notamment, ainsi que dans le domaine de la reconstitution automatique des familles ([B.SK2]); aucun de ces critères ne répond clairement à la question suivante: un degré

de ressemblance élevé correspond-t'il à une forte probabilité pour que deux enregistrements soient homologues?

Dans ce qui suit, nous tentons de répondre à cet objectif: l'expression des critères proposés découle directement de la probabilité pour que les deux enregistrements soient homologues, sachant qu'ils sont cohérents.

Les évaluations correspondantes sont bien entendu partiellement empiriques comme cela est presque toujours le cas dans ce type de mesure; nous recherchons en fait surtout un moyen de comparaison entre plusieurs arcs d'une même classe. Les définitions proposées sont assez simples et certainement susceptibles d'améliorations; comme dans le chapitre précédent, il est vraissemblablement possible de mieux utiliser les possibilités de la théorie de l'information pour affiner ces critères.

2.6.3.b - Critère global:

On peut définir, relativement à une relation simple, un critère de ressemblance global G, qui ne tient compte que du fait que deux enregistrements sont cohérents pour cette relation. Considérons l'expression de la probabilité conditionnelle $p(\sim|\not\sim)$ que deux enregistrements soient homologues, sachant qu'ils sont cohérents pour la relation considérée (indépendamment des valeurs particulières pour lesquelles cette cohérence est obtenue):

$$(y)\quad p(\sim|\not\sim) = p(\sim)p(\not\sim|\sim)/p(\not\sim) = p_1p_3/p_2$$

où:

- la probabilité p_1 que deux enregistrements soient homologues est inconnue, mais est une constante dans l'application; on peut donc ignorer ce facteur multiplicatif dans la comparaison de plusieurs arcs.

- la probabilité p_2 que deux enregistrements soient cohérents pour la relation considérée peut être évaluée à partir des répartitions des valeurs des caractéristiques concernées. Rappelons que ces répartitions sont également nécessaires pour l'évaluation des critères proposés au chapitre précédent.

- la probabilité p_3 que deux enregistrements soient cohérents pour la relation considérée, sachant qu'ils sont homologues, est théoriquement égale à 1 quelle que soit la relation. Toute différence par rapport à

cette valeur ne peut être due qu'à des erreurs ou des omissions affectant les caractéristiques concernées par la relation. On affecte empiriquement une valeur k, comprise entre 0 et 1, à cette probabilité en fonction de la qualité observée de ces informations.

En fonction de ces remarques, nous proposons la définition suivante du critère de ressemblance global associé à une relation:

$$G = \log_2 (1 + k/p(\not\sim))$$

où: $k, p(\not\sim) \in]0, 1]$

On vérifie facilement que $G \geq 0$; par convention, on attribue une valeur nulle à G chaque fois que la relation n'est pas vérifiée ou est non évaluable (l'un au moins des arguments a une valeur indéfinie).

L'avantage de ce type de critère est qu'il n'est évalué qu'une seule fois pour la relation considérée; inversement, sa précision est nécessairement limitée.

3.6.3.c - Critère fin:

On peut définir, pour toute relation élémentaire, un critère de ressemblance fin, tenant compte de l'élément particulier de la relation pour lequel la cohérence est constatée. La probabilité conditionnelle $p(\sim|\not\sim_j)$ que deux enregistrements soient homologues sachant qu'ils sont cohérents pour un élément particulier de la relation, s'écrit de manière analogue à (y):

$$(z)\ p(\sim|\not\sim) = p(\sim)p(\not\sim_j|\sim)/p(\not\sim_j) = p_1 p_3/p_2$$

où:

- comme dans (y), p1 est une constante pour l'application et peut être ignorée dans une évaluation destinée à comparer divers arcs.
- la probabilité p_2 que deux enregistrements soient cohérents pour un élément particulier de la relation peut être évaluée à partir des probabilités des valeurs concernées.
- la probabilité p_3 que deux enregistrements soient cohérents pour

l'élément particulier de la relation, sachant qu'ils sont homologues, doit être évaluée empiriquement; les deux possibilités suivantes sont proposées:

- utiliser des répartitions connues, issues d'expériences antérieures, si on les estime applicables au cas considéré. Dans des applications démographiques, on peut ainsi réutiliser des répartitions connues d'âge au mariage, au décès, si la population en cours de reconstitution a vécu dans des conditions estimées comparables.

- lorsqu'aucune répartition de référence n'est disponible, par exemple pour les noms patronymiques, on peut recourir à une estimation empirique fondée sur de nombreuses expériences: la cohérence pour des valeurs de faible probabilité est le plus souvent significative d'une forte probabilité d'homologie des enregistrements concernés. Diverses évaluations sont possibles pour traduire cette hypothèse; nous proposons la mesure empirique suivante, qui avantage les éléments de faible probabilité et pénalise ceux ayant une forte probabilité:

$$p(\gamma_j|\sim) \simeq (1 - p(\gamma_j))^m \qquad \text{où } m \in [0,1]$$

Le coefficient correcteur m est fixé pour chaque relation.

En fonction des remarques précédentes, nous proposons la définition suivante du critère de ressemblance fin:

$$F = \log_2(1 + p(\gamma_j|\sim)/p(\gamma_j)) \qquad F \geq 0$$

où les probabilités conditionnelles ont l'une des expressions données ci-dessus.

Ce type de critère est certainement plus efficace qu'un critère global, mais sont coût d'évaluation est plus élevé. Comme précédemment, on considère que la ressemblance est nulle lorsque la relation n'est pas évaluable.

2.6.3. d - Critère de ressemblance dans un ensemble de relations:

La mesure de ressemblance associée à une relation de cohérence est obtenue par sommation des critères choisis pour chacune des relations composantes.

Selon la nature et la qualité des informations utilisées, on peut définir pour chacune des relations considérées le type de critère paraissant le mieux adapté (global ou fin). Le choix des paramètres empiriques propres à chacun d'eux est effectué par expérimentation sur des ensembles de cas jugés significatifs; le contexte interactif des opérations de couplage et de jumelage est très précieux à cet égard (cf 3.3.5 et 3.3.6).

2.6.4. Conclusion

L'étude présentée dans ce chapitre est encore embryonnaire; de nombreux développements sont possibles sur le plan théorique en utilisant de manière plus approfondie les possibilités de la théorie de l'information.

De telles études sont indispensables dans le domaine d'application considéré, car de la qualité des critères choisis dépend directement celle des solutions construites par le système de reconstitution automatique.

CHAPITRE 2.7. - Sélection des solutions - Décision de jumelage :

2.7. Sélection des solutions - Décision de jumelage

2.7.1. Introduction

Considérons une classe de cohérence c construite lors du couplage de N fichiers $E^1,\ldots,E^n$; si les solutions de cohérence large ont été correctement définies (cf 2.4), cette classe contient un nombre fini de classes de la relation d'homologie (cf 2.1.3.a), avec la contrainte supplémentaire suivante (cf. exemple en 2.4.3.e) :

(a) Toute classe d'homologie appartenant à c est un sous-graphe de c.

Dans le cas général, la construction des classes d'homologie implique la vérification de contraintes supplémentaires que nous appellerons règles de compatibilité, liées à la nature des données traitées (par exemple l'unicité de la naissance, du décès dans la biographie d'une personne, la multiplicité possible ou non des mariages selon le contexte social, etc.).

On appelle solution de jumelage toute partition d'une classe de cohérence vérifiant les règles de compatibilité prédéfinies pour l'application, et la condition (a).

Un premier problème consiste donc à définir un algorithme permettant de construire les différentes solutions de jumelage (extraction), et un second problème consiste à trouver les solutions jugées les meilleures (sélection). La résolution de ces deux problèmes suppose d'une part une définition des règles de compatibilité, d'autre part une définition des critères d'évaluation de la qualité d'une solution. Il est naturellement possible d'obtenir plusieurs solutions de même qualité maximale à partir d'une classe; dans ce cas, il subsiste une ambiguïté qui ne peut être levée qu'arbitrairement (décision aléatoire), ou par l'utilisateur finale (décision forcée). En raison de la forte combinatoire liée à l'opération d'extraction des solutions, nous avons eu recours aux techniques classiques de l'intelligence artificielle [B.NIL] pour combiner les opérations d'extraction et de sélection. L'introduction de règles de compatibilité relativement souples, ainsi que la prise en compte d'un critère de qualité dans l'élaboration des solutions constituent l'essentiel des extensions que nous avons apportées à l'algorithme de sélection proposé par M. SKOLNICK dans sa thèse [B.SK1]; une amélioration de la méthode de sélection est suggérée en 2.7.5.d.

2.7.2. Définition des règles de compatibilité - Possibilité de jumelage

2.7.2.a) Classes de cohérence - Classes d'homologie.

La condition d'inclusion (a) est directement déduite de la propriété (1) donnée en 2.2.1., définissant la dépendance des notions d'homologie et de cohérence; rappelons que les diverses extensions des relations de cohérence analysées dans le chapitre 2.4 n'ont d'autre but que d'assurer que tout couple d'enregistrements homologues sont cohérents dans l'ensemble des relations définies, en dépit des erreurs et des omissions. Considérons une classe de cohérence c, construite lors du couplage de N fichiers $E^1,\ldots,E^n$; par construction, c est un graphe possédant les propriétés suivantes :

b) c'est un graphe M-parti, $M \leq N$ où M est le nombre de couvertures $X^i \subseteq E^i$ de la classe. Par définition, on appellera désormais trace d'une classe, l'ensemble $t \subseteq \{1,\ldots,N\}$ des indices de ses couvertures.

c) c'est un graphe connexe.

Considérons à présent une classe d'homologie $h \subseteq c$; h étant une classe d'équivalence, et d'après la condition a), on a les propriétés suivantes du graphe m-parti correspondant à la relation d'homologie restreinte à h :

d) h est un graphe m-parti, inclus dans c, avec $m \leq M$

e) h est un graphe complet

f) h est un graphe vérifiant l'ensemble des règles de compatibilité définies pour l'application (cf ci-après).

2.7.2.b) Définition des règles de compatibilité

Les règles évoquées dans l'introduction de ce chapitre et dans la propriété (f) ci-dessus), peuvent être considérées comme des contraintes relatives au degré des sommets de toute classe d'homologie. Dans le cas général on doit prévoir l'existence de contraintes particulières à tout couple de fichiers (de la même façon que l'on avait envisagé des relations de cohérence particulières à tout couple de fichiers). Nous avons considéré la définition suivante d'une règle de compatibilité r_{ij} entre deux fichiers E^i et E^j :

$$(g) r_{ij} : min_{ij} \leq degré_{ij}(e) \leq max_{ij}, \forall e \in X^i \subseteq E^i, \forall i,j \in [1..N], i \neq j$$

avec $min_{ij} \geq 0$, $max_{ij} \geq 1$

Pour tout graphe analysé, possédant une couverture X^i dans E^i, les règles sont interprétées de la façon suivante :

a) degré minimum

Tout sommet e du graphe appartenant à $X^i \subseteq E^i$, doit vérifier la condition suivante :

$degré_{ij}(e) \geq min_{ij}$, $\forall j \in [1..N]$, $i \neq j$

La valaur minimale 0 pour min_{ij} permet d'envisager des classes d'homologie qui ne sont pas systématiquement des graphes N-partis (une ou plusieurs couvertures de la classe peuvent être vides). Lorsque $min_{ij} = 0$ pour toutes les règles, les classes d'homologie peuvent être ces graphes vérifiant les conditions (d), (e), (f), pour tout m compris entre 1 et $M \leq N$.

b) degré maximum

Tout sommet e du graphe, appartenant à $X^i \subseteq E^i$, doit vérifier la condition suivante :

$degré_{ij}(e) \leq max_{ij}$, $\forall j \in [1..N]$, $j \neq i$

Cette contrainte permet de limiter la taille admissible des classes d'homologie.

La définition des constantes min et max pour tous les couples de fichiers, est un choix dépendant de la nature des informations à jumeler (cf exemple dans l'introduction); elles déterminent l'ordre minimal et maximal des classes d'homologie dans l'application considéré.

Pour une application particulière, les valeurs de min et max pour toute règle de compatibilité sont définies de la manière suivante :

c) définition des min

Elle s'effectue à partir d'une définition minimale des classes d'homologie dans les N fichiers traités. Cette définition obéit aux règles restrictives suivantes (cf exemple ci-après) :

- les classes sont des graphes m-partis complets (éventuellement réduits à un seul sommet);
- les classes sont disjointes (cf remarque ci-dessous);
- la définition des classes est complète : chacun des N fichiers fait partie de la définition d'une classe minimale (pour obtenir une définition de tous les min).

Les valeurs de min et max sont alors directement données par le degré des sommets dans le graphe N-parti constitué par l'union des classes minimales (s'il n'y a pas d'arc entre un sommet de E^i et un sommet de E^j, alors $min_{ij}=0$).

Remarque : On considère des classes disjointes pour obtenir une définition unique de min_{ij}, quel que soit le couple de fichiers E^i, E^j considéré. Il serait intéressant de pouvoir lever cette contrain pour permettre une définition plus souple des configurations minimales admises; cependant, nous n'avons pas démontré que dans cette hypothèse l'algorithme de construction des solutions, est correct.

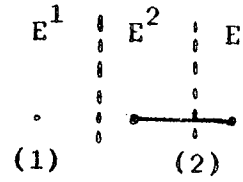
Les classes minimales étant disjointes, on a :

(h) $min_{ij} = 0 \Leftrightarrow min_{ji} = 0$

d) Définition des max

On considère, pour l'application envisagée, la taille maximale des couvertures d'une classe d'homologie dans les différents fichiers de départ. Les valeurs des max sont alors obtenues en considérant le degré de chaque sommet dans le graphe N-parti complet correspondant. Lorsque le nombre maximal d'éléments d'une couverture ne peut être prédéterminé, on attribue aux max correspondants une valeur notée vmax, majorant le cardinal des N fichiers.

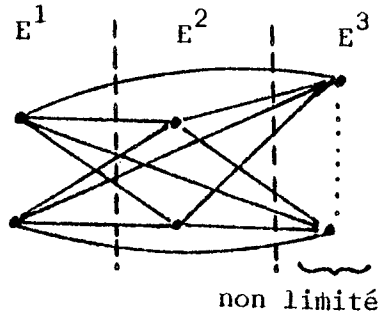
Exemple : Considérons trois fichiers E^1, E^2, E^3, et la définittion ci-dessous des classes minimales pour l'application :



On a donc : $min_{12}=0$; $min_{13}=0$ d'après (1)

et $min_{23}=min_{32}=1$, $min_{21}=min_{31}=0$ d'après (2).

Considérons la définition ci-dessous de la classe maximale admissible :



(3)

On a donc d'après (3) :

$max_{12}=max_{21}=2$

$max_{13}=vmax$, $max_{31}=2$

$max_{23}=vmax$, $max_{32}=2$

L'ensemble des règles de compatibilité correspondant à cet exemple peut être résumé par la matrice ci-dessous :

	r_{12}	r_{13}	r_{23}	r_{21}	r_{31}	r_{32}
min	0	0	1	0	0	1
max	2	vmax	vmax	2	2	2

2.7.2.c) Possibilité de jumelage

Dans le cas général, il peut exister plusieurs décompositions d'une classe de cohérence c en sous-graphes vérifiant les propriétés (d), (e), (f); pour envisager l'ensemble de ces décompositions, nous considérons une entité particulière appelée possibilité de jumelage, notée p. Si c est un graphe M-parti de trace T, on considère par définition comme possibilité de jumelage tout sous-graphe de c, de trace $t \subseteq T$, complet et d'ordre minimal compte tenu des règles de compatibilité. On peut résumer ainsi les différentes propriétés du graphe correspondant à une possibilité p de c, d'ordre ≥ 2 :

(i) p est un graphe m-parti, $1 \leq m \leq M$, inclus dans c.

(j) p est un graphe complet.

(k) la trace t de p est complète, compte tenu des règles de compatibilité :

$\forall i \in t, \forall j \in [1..N], i \neq j, min_{ij} \neq 0 \Rightarrow j \in t$

(l) Tous les sommets e de p vérifient les contraintes de degré minimales suivantes :

$\forall i,j \in [1,..,N], i \neq j, \forall e \in X^i$, $degré_{ij}(e)=d$, avec

(l1) $min_{ij} \neq 0 \Rightarrow d=min_{ij}$

(l2) $min_{ij} = 0 \Rightarrow d=MAX(1,dmax)$

où $d = MAX\ (min_{kj}) \geq 0$, $k \in t - \{j\}$

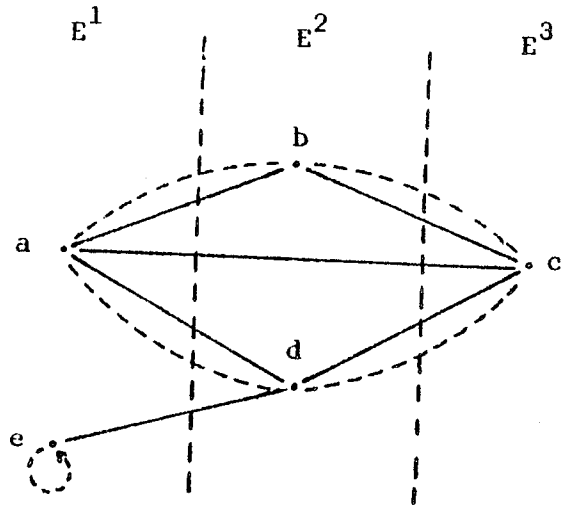
Les propriétés (11) et (12) garantissent que le graphe p est complet, et est d'ordre minimal compte tenu des règles de compatibilité. Le cas limite d'une possibilité d'ordre 1 ne peut exister dans E^i que si :

(m) $\forall j \in [1,..,N],\ j \neq i,\ \min_{ij} = 0$

Remarquons que si seul le cas (11) est rencontré dans une possibilité p, ce graphe correspond à un schéma de classe minimale, d'ordre ≥2, telles qu'elles ont été définies en 2.7.2.b).

L'exemple ci-dessous illustre la décomposition d'une classe de cohérence en possibilités de jumelage.

Exemple : reprenons la définition des règles de compatibilité données dans l'exemple précédent, et considérons la classe de cohérence ci-dessous :



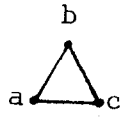
Comme dans les exemples précédents, les arcs en pointillés représentent le graphe de la relation d'homologie (la réalité à reconstituer); les classes d'homologie sont donc ici (a,b,c,d) et (e).

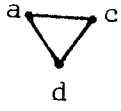
Les possibilités de jumelage sont données ci-dessous, compte tenu des règles de compatibilité :

Possibilités

ordre 1
P_1 : a. $t_1=\{1\}$
P_2 : e. $t_2=\{1\}$

ordre 2
P_3 : b — c $t_3=\{2,3\}$
P_4 : d — c $t_4=\{2,3\}$

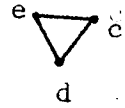
ordre 3
P_5 : 

$t_5=\{1,2,3\}$
P_6 : 

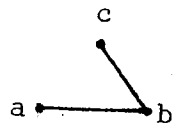
$t_6=\{1,2,3\}$

Exemples de non-possibilités

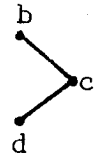
a — b car $degré_{23}(b)=0<min_2$



car non complet (l'ar
ec n'existe pas dans
classe)



car non complet



car $degré_{32}(c)=2>min_3$

etc.

+++

Etant donné une classe de cohérence c, on engendre à partir du graphe correspondant l'ensemble P des possibilités qu'elle contient, c'est-à-dire tous les graphes m-partis, s'ils existent, avec $1\leq m\leq M$ vérifiant les propriétés (i), (j), (k), (l). Comme nous le montrons dans le chapitre suivant, P peut servir de base à la construction de toutes les solutions de jumelage.

2.7.3. Solutions de jumelage - Propriétés

2.7.3.a) Définition

Une solution de jumelage est constituée par toute partition d'une classe de cohérence, dont les éléments vérifient les propriétés (d), (e), (f).

Pour une classe de cohérence c, M-partie, d'ordre m, on peut donc énoncer les propriétés suivantes de toute solution de jumelage s correspondante :

(n) s est un graphe m-parti, $m \leq M$, d'ordre $n' \leq n$, inclus dans c;

(o) toutes les composantes connexes de s vérifient les propriétés (e), (f).

Dans ce qui suit, nous allons montrer que toute solution de jumelage dans une classe de cohérence, peut être obtenue à partir de l'ensemble des possibilités de jumelage correspondantes; cette méthode de construction des solutions rejoint des techniques utilisées en documentation automatique [B.GAI

2.7.3.b) Construction des solutions de jumelage

Par définition des possibilités (i), toute union arbitraire de ces graphes constitue un graphe inclu dans la classe de cohérence d'origine; la propriété (n) est donc toujours vérifiée quelle que soit la combinaison de possibilités envisagée. Remarquons que les inégalités $m \leq M$, $n' \leq n$ de la propriété (k) sont liées au fait que certains sommets de la classe de départ peuvent ne vérifier aucune des règles de compatibilité; par définition des possibilités (cf propriétés (k) et (l)), ces sommets ne figurent dans aucune possibilité de jumelage.

On démontre facilement la propriété suivante :

Propriété 1

Toute possibilité de jumelage, ou toute union de possibilités de jumelage constitue une solution de jumelage.

La propriété suivante définit, avec la propriété 1 ci-dessus, une méthodologie de construction des solutions de jumelage à partir des possibilités :

Propriété 2 :

Etant donné un sous-graphe g d'une classe de cohérence, vérifiant les conditions (d), (e), (f), s'il n'est pas lui-même une possibilité de jumelage, il est décomposable en un nombre fini de possibilités de jumelage ayant la même trace que g.

Démonstration : Considérons le cas où le graphe g donné n'est pas une possiblité; soit t sa trace et n son ordre. g vérifie les propriétés (d), (e), (f); par conséquent :

(1) g est un graphe m-parti, avec m=|t|

(2) g est un graphe complet

(3) g vérifie toutes les règles de compatibilité.

Si g n'est pas une possibilité, c'est qu'il n'est pas d'ordre minimal, compte tenu de sa trace (une des couvertures contient au moins un sommet de trop pour que g satisfasse strictement aux contraintes minimales de degré exprimées par (1)).

Considérons le cas où g ne comporte qu'un seul sommet redondant $e \in X^i$; la démonstration qui suit se généralise facilement à un nombre à un nombre quelconque de sommets redondants.

Eclatons le graphe g en deux sous-graphes g_1 et g_2 tels que :

- g_1 est le sous-graphe de g obtenu en restreignant X^i à {e}.
- g_2 est le sous-graphe de g obtenu en restreignant X^i à X^i-{e}.

Il est clair que g_1 et g_2 sont des graphes m-parti complets de trace t; par ailleurs si e est le seul élément redondant de g, g_1 et g_2 sont d'ordre minimal, compte tenu de t et des règles de compatibilité. g_1 et g_2 sont donc des possibilités de trace t.

+++

Il résulte de cette propriété que toute composante connexe de trace t d'une solution de jumelage peut être obtenue par l'union d'un nombre fini de possibilités ayant la même trace t.

Toute solution de jumelage peut donc être obtenue en ne considérant que des combinaisons de possibilités de jumelage. Dans un processus de constructi des solutions par unions successives de possibilités, un facteur de simplification intéressant consiste à connaître dans quelles conditions chaque union effectuée vérifie les conditions (d), (e), (f). La propriété ci-dessous est, à ce titre, intéressante dans le cas de l'union de graphes m-partis complets; elle ne peut cependant pas être utilisée comme seul critère dans la construction des solutions, car dans le processus d'unions successives, une composante peut être temporairement non complète.

Propriété 3 :

Une condition nécessaire et suffisante pour que l'union de deux graphes distincts g_1 et g_2 m-partis complets, constitue un graphe m-parti complet, est que :

- **g_1 et g_2 aient des traces identiques**
- **g_1 et g_2 aient m-1 couvertures identiques.**

Les trois propriétés précédentes nous permettent de définir des règles de combinaison des possibilités entre elles, ou de possibilités avec des graphes déjà obtenus, garantissant l'obtention d'une solution de jumelage.

Règle 1 : Tout couple de graphes disjoints est acceptable, d'après la propriété 1.

Règle 2 : On accepte toute combinaison de graphes connexes vérifiant les propriétés suivantes :

(2-1) ils ont une trace identique t

(2-2) ils ont m-1 couvertures identiques (m=|t|)

(2-3) tous les sommets de ces m-1 couvertures vérifient les contraintes maximales de degré définies par les règles de compatibilité.

Exemple : Dans l'exemple du chapitre précédent : la "bonne" solution est donnée par l'union de P_5, P_6,P_2; d'autres solutions sont données par P_1,P_2,P_3; P_1,P_2,P_4; P_1,P_2,P_5,P_6 etc.

2.7.4. Evaluation du poids d'une solution

2.7.4.a) Introduction

Dans le chapitre 2.6.3, nous avons proposé diverses pondérations des arcs composant une classe de cohérence; ces mesures vont nous permettre de sélectionner, parmi toutes les solutions de jumelage possibles, celles qui paraissent les "meilleures". Il est important de noter que, là comme ailleurs, la qualité d'une solution ne dépend pas de critères absolus, admis par tous les utilisateurs; on peut au moins considérer deux définitions courantes (mais relativement opposées) d'une "bonne" solution :

(1) C'est une solution qui permet le jumelage d'un maximum d'enregistrements de la classe

(2) C'est une solution qui n'autorise que les jumelages les plus sûrs.

On voit que ces deux points de vue ne peuvent guère être conciliés que si les données de départ sont suffisamment fiables pour qu'un maximum de jumelages autorisés soient les plus sûrs possibles ! (c'est-à-dire en fait dans les applications ne posant pas de problèmes).

La solution que nous proposons a pour point de départ les **constatations** suivantes :

- Si les données sont de mauvaise qualité, il en résulte à la fois une grande ambiguïté (donc des classes de cohérence d'ordre élevé), et une valeur moyenne faible du poids des arcs. Dans un tel contexte choisir les solutions jumelant le maximum d'enregistrements est assez hasardeux quant à la qualité des résultats.

- Si les données sont de bonne qualité, les effets sont opposés : classes de cohérence d'ordre faible, valeur élevée du poids moyen des arcs. Dans cette situation, on peut sans grand risque opter pour le jumelage d'un maximum d'éléments.

Il nous a donc paru plus naturel de ne pas choisir a priori entre les définitions (1) et (2), et d'essayer de s'orienter automatiquement vers l'une ou l'autre selon les cas de figure posé par chaque classe de cohérence. L'orientation du processus de sélection des solutions en fonction de ces deux interprétations de leur qualité peut se traduire comme suit :

(3) Un jumelage correspondant à (1) consiste à rechercher de préférence des solutions fortement connexes (le poids moyen des arcs pouvant alors être relativement faible).

(4) Un jumelage correspondant à (2) consiste à rechercher des solutions ne retenant de préférence que des arcs présentant un poids moyen élevé (la connexité obtenue pouvant alors être faible).

La méthode d'évaluation du poids d'une solution que nous proposons ici, tend à favoriser les solutions fortement connexes lorsque la qualité moyenne des possibilités est élevée, et à favoriser les solutions faiblement connexes lorsque cette possibilité moyenne est faible.

2.7.4.b) Définition d'un critère de qualité

Considérons l'ensemble P de possibilités extraites d'une classe de cohérence donnée; soit m le poids moyen des arcs dans cet ensemble. La qualité globale q des éléments de P est déterminée par rapport à une valeur de référence Q fixée pour chaque application :

(p) $q = Q/m$

La valeur de Q est fixée par l'utilisateur; elle correspond à un poids moyen des arcs dans tout ensemble de possibilités tel que :

- Si $m \geq Q$, la qualité moyenne de l'ensemble des possibilités est estimée bonne, et on peut s'orienter vers des solutions fortement connexes. On a alors $q \leq 1$.
- Si $m < Q$, la qualité moyenne de l'ensemble des possibilités est jugée insuffisante, et on s'oriente vers des solutions faiblement connexes, ne reteanant que les arcs présentant la meilleure qualité. On a alors $q > 1$.

La valeur de Q dans une application particulière est déterminée par l'expérience : essais successifs de sélection de solutions dans un ensemble de classes jugé représentatif.

2.7.4.c) Définition du poids d'une possibilité de jumelage

Etant donné une possibilité $p \in P$, composée d'un ensemble d'arcs a_i de poids notés $pds(a_i)$, on définit le poids de p par :

$$(q) \quad pds(p) = \frac{1}{n^q} \sum_{a_i \in p} pds(a_i) = \frac{y_i}{n^q}$$

où $n \geq 1$ est le nombre d'arcs composant p, et q est le critère de qualité défini selon (p) pour P.

Par définition, une possibilité d'ordre 1 (n=0) a un poids nul.

Considérons les propriétés essentielles de cette mesure :

a) Possibilité de bonne qualité moyenne

Cela correspond à une valeur de $q \leq 1$,; lorsque q tend vers 0, la définition ci-dessus de poids d'une possibilité tend à se réduire à la somme des poids des arcs composants. On rejoint alors la définition du poids d'une possibilité, donnée par M. SKOLNICK dans sa thèse. L'évaluation tend alors à accorder le plus fort poids aux possibilités d'ordre élevés, et donc, aux contraintes de compatibilité près, à favoriser les solutions fortement connexes (cf. 2.7.4.e)).

Dans le cas particulier où q=1, le poids d'une possibilité est égal au poids moyen des arcs composants.

b) Possibilité de mauvaise qualité moyenne

Cela correspond à une valeur de q > 1; lorsque q devient grand par rapport à 1, les possibilités d'ordre élevé ont leur poids considérablement réduit et on fournira plutôt les solutions faiblement connexes, construites à partir de possibilités d'ordre peu élevé (cf 2.7.4.e)).

2.7.4.d) Incidence du facteur de qualité dans la sélection des solutions

Considérons le cas général où les éléments de P ne sont pas tous mutuellement compatibles. L'existence simultanée de solutions fortement et faiblement connexes dépend principalement du degré de connexité entre elles des différentes possibilités, et des incompatibilités qui peuvent en résulter (traces différentes, contraintes de max non respectées). Si on considère deux possibilités non compatibles, elles ne peuvent qu'appartenir à des solutions distinctes; si de plus leurs degrés sont très différents, les solutions produites pourront présenter des caractères de connexité très différents.

On note S l'espace des solutions de jumelage obtenues à partir d'un ensemble P de possibilités. Les solutions recherchées sont les éléments de poids maximal de S.

Considérons un ensemble P dont deux éléments p_1 et p_2 sont tels que :

- p_1 et p_2 sont incompatibles.
- p_1 et p_2 sont respectivement compatibles avec l'ensemble $P' = P - \{p_1, p_2\}$ des possibilités restantes.
- tous les éléments de P' sont compatibles entre eux.

Soit s_1 la solution constituée par $p_1 \cup P'$, s_2 la solution constituée par $p_2 \cup P'$, et $q = Q/m$ la qualité moyenne des arcs de P. D'après (r), on a :

$$pds(s_1) = pds(p_1) + x$$

$$pds(s_2) = pds(p_2) + x$$

où $x = \sum_{p_i \in P'} pds(p_i)$

Les éléments de P' étant communs aux deux solutions, leur poids, quel que soit le facteur de qualité global q, est le même dans les deux solutions.

La solution de plus fort poids correspondra donc à celle qui contient, de p_1 et p_2, la possibilité de plus fort poids; considérons l'inégalité :

$pds(p_1) \geq pds(p_2)$

ou, d'après (q) :

(t) $y_1 \geq \left(\frac{n_1}{n_2}\right)^q y_2$

Considérons les conditions de réalisation de cette inégalité en fonction des valeurs du facteur de qualité q :

a) Possibilités de bonne qualité ($0 < q \leq 1$)

Lorsque n_1/n_2 est peu différent de 0, (t) se réduit pratiquement à $y_1 \geq y_2$, et s_1 est la meilleure solution si la somme des poids des arcs constituant p_1 est supérieure à la quantité correspondante dans p_2.

Lorsque p_1 et p_2 sont d'ordres différents, le facteur de qualité favorise d'autant plus les possibilités d'ordre élevé, qu'il a une valeur plus proche de 0; (t) peut en effet s'écrire :

(u) $m_1 \geq \left(\frac{n_1}{n_2}\right)^{q-1} \times m_2$, où $m_1 = y_1/n_1$, $m_2 = y_2/n_2$ sont les poids moyens des arcs de p_1 et p_2 respectivement.

Supposons que $n_1 < n_2$ (p_1 est d'ordre plus faible que p_2); on vérifie facilement que, pour $0 < q \leq 1$, (t) n'est vérifiée que si $m_1 \geq km_2$, avec $k \geq 1$. La valeur maximale de k est atteinte pour q tendant vers 0. Le cas limite où $k = 1$ correspond à $q = 1$ (le poids moyen des arcs de P est égal à la valeur de référence Q); (t) se réduit alors à la comparaison directe des poids moyens m_1 et m_2.

b) Possibilités de mauvaise qualité ($q > 1$)

Reprenons l'inégalité (u) ci-dessus, avec $n_1 > n_2$; on montre facilement que pour $q > 1$, elle se réduit à :

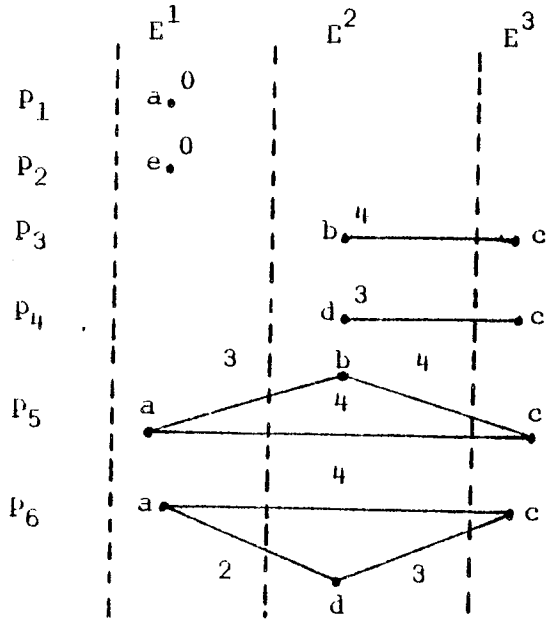
$m_1 \geq km_2$, avec $k < 1$ et que k est une fonction décroissante de q.

Lorsque la qualité moyenne des possibilités est faible, les possibilités d'ordre faible sont donc avantagées par rapport aux possibilités d'ordre élevé.

Si nous revenons à la comparaison des solutions s_1 et s_2, on voit donc que le choix entre ces deux solutions dépend fortement de q; pour des valeurs de m_1 et m_2 données, et pour $n_1 < n_2$:

- Si $0 < q \leq 1$, la solution s_2 sera favorisée, et on s'orientera donc vers une solution fortement connexe ($n_2 > n_1$).
- Si $q > 1$, la solution s_1 sera favorisée, et on s'orientera donc vers une solution faiblement connexe.

Exemple : Reprenons l'exemple donné en 2.7.2.c), avec les poids ci-dessous des arcs composant les diverses possiblités :



- Les arcs constituant P sont :
  ab, bc, ac, ad, dc
- Leur poids moyen est :
  $\bar{m} = 3,20$

Considérons la sélection de la meilleure solution dans les deux cas de figure suivants :

a) Les arcs sont de bonne qualité

Soit, par exemple, une valeur de Q = 2,5; le facteur de qualité q est alors égal à Q/m, soit : q = 0,78.

On a alors les poids ci-dessous des possibilités, d'après (q) :

p_1	p_2	p_3	p_4	p_5	p_6
0	0	4	3	4,67	3,82

La solution la meilleure est alors constituée par (p_5, p_6) avec un poids égal à $pds(p_5) + pds(p_6) = 8,49$. C'est une solution fortement connexe.

b) Les arcs sont de mauvaise qualité

Soit, par exemple, une valeur de Q = 4; le facteur de qualité q est alors :

$q = 4/3,20 = 1,25$

On a alors les poids ci-dessous des possibilités, d'après (q) :

p_1	p_2	p_3	p_4	p_5	p_6
0	0	4	3	2,79	2,28

Pour cette valeur du facteur de qualité, la meilleure solution est constituée par (p_3, p_4), avec un poids égal à 7. Remarquons que, dans ce cas, la solution la plus fortement connexe (p_5, p_6) n'a qu'un poids égal à 5,07.

2.7.5. Algorithmes de sélection des meilleures solutions

2.7.5.a) Présentation générale

A partir d'un ensemble P de possibilités muni de la mesure définie précédemment, il s'agit de construire l'ensemble des solutions de poids maximal.

Une première difficulté réside dans la combinatoire généralement élevée du processus de sélection des possibilités pouvant constituer une solution. L'algorithme présenté par M. SKOLNICK dans sa thèse utilise certaines techniques de l'intelligence artificielle pour résoudre ce problème [B.SK1]; après un essai d'implémentation, les premières améliorations apportées ont consisté à étendre l'algorithme à des règles de compatibilité plus générales (cf 2.7.2.b)) et à reconsidérer l'évaluation des poids des possibilités et de là, celles des solutions (cf 2.7.4).

Ces modifications visaient tout d'abord à une amélioration des résultats produits, indépendamment de la stratégie de construction des solutions. Sur le plan de l'utilisation des techniques de l'intelligence artificielle, il est probable que des améliorations importantes peuvent être apportées à l'algorithme initial, notamment par le passage à une méthode de type branch and bound (cf 2.7.5.d)). Les deux types d'algorithmes présentés plus loin utilisent cependant la même méthode d'estimation des solutions en cours de construction (fonction heuristique); ils diffèrent surtout par la stratégie de sélection des solutions à dériver.

2.7.5.b) Définition d'une fonction d'estimation

Considérons un exemple P de possibilités, et pdstotal la somme des poids de ses éléments. D'après (s), toute solution s est telle que pds(s)≤pdstotal; considérons l'état noté s_n de s après la prise en compte de n possibilités constituant un sous-ensemble P' ⊆ P (solution partielle). On a :

$$pds(s_n) \leq \sum_{p \in P'} pds(p) = a$$

Notons $P'' \subseteq P'$ l'ensemble des possibilités compatibles constituant s_n; on a :

$$pds(s_n) = \sum_{p \in P''} pds(p) = b$$

La différence a - b = c représente le poids des possibilités rejetées lors de la construction de s_n; le poids maximal de toute solution s dérivée de s_n à partir des possibilités restantes dans P-P' est donc =

(u) $lim(s_n) = pds(s_n) + pds(P-P') = pds(s_n) + reste$

soit encore

(v) $lim(s_n) = pdstotal - c$

Le poids d'une solution dérivée de s_n ne peut atteindre cette valeur maximale que si l'ensemble des possibilités restantes dans P-P' sont compatibles avec les possibilités constituant $s_n(P'')$. L'expression (u) est la fonction d'évaluation utilisée par M. SKOLNICK, que nous reprenons au chapitre suivant; l'expression (v) est utilisée dans l'algorithme de type branch and bound présenté en 2.7.5.d) ([B.THE]).

Considérons à présent deux solutions partielles s_i et s_j obtenues au pas n de l'algorithme, et soit pdsmax le poids maximal des solutions (partielles ou complètes) obtenues à ce stade; on peut limiter l'expansion de l'arbre de dérivation des solutions en éliminant les solutions partielles qui ne peuvent en aucun cas conduire à des solutions maximales. L'élimination de s_i, par exemple, de l'arbre de dérivation au pas n+1 de l'algorithme, correspond à la vérification de la condition suivante :

(w) $lim(s_i) < pdsmax$

La condition (w) est celle utilisée dans l'algorithme proposé par M.SKOLNICK; elle garantit l'obtention des solutions de poids maximal en fin de traitement Naturellement, l'algorithme convergera d'autant plus rapidement vers ces solutions, qu'on aura pu éliminer plus tôt les solutions partielles n'ayant aucune chance de produire une solution maximale; la condition (w) est donc

critique pour les performances de l'algorithme, et on retrouve la nécessité pour la fonction d'évaluation de délivrer le plus petit majorant possible.

La fonction définie par (u) ou (v) est très facilement calculable, mais n'est pas très performante car elle ne tient pas compte des incompatibilités existantes parmi les possibilités restant à traiter. D'autres fonctions heuristiques tenant compte de ceci sont envisageables, mais il n'est pas évident que leur coût d'évaluation ne compense pas le gain de performance obtenu en limitant le nombre de solutions partielles considérées.

En tout état de cause, pour rendre (w) aussi efficace que possible, il faut que les possibilités soient considérées par ordre de poids décroissant; s'il se trouve que les possibilités de plus fort poids ont un fort degré de compatibilité, pdsmax prendra rapidement des valeurs élevées, et la condition (w) sera effective plus tôt. Cet ordre de considération des possibilités dans le processus de sélection des solutions est un élément essentiel de l'heuristique mise en oeuvre dans les deux algorithmes proposés ci-dessous.

2.7.5.c) Premier algorithme de sélection

Nous présentons brièvement l'algorithme proposé par M. SKOLNICK, [B.SK2] qui a été implémenté avec les extensions mentionnées plus haut.

Partant d'un espace de solutions vide S_0, au pas 1 de l'algorithme on construit une solution s_1 à partir de la possibilité de plus fort poids p_1; pdsmax prend pour valeur $pds(p_1)$ et $S_1=s_1$.

Après le pas n-1 de l'algorithme, on a un espace S_{n-1} de solutions partielles obtenues à partir du traitement n-1 premières possibilités; toutes ces solutions ont leur poids effectif majoré par pdsmax.

Au pas n de l'algorithme, on construit l'espace de solutions S_n de la façon suivante :

a) S_n est considéré, au départ, comme égal à S_{n-1}
On sélectionne dans P la possibilité p_n dans l'ordre décroissant des poids.

b) Pour chaque élément s de S_{n-1} on effectue le traitement suivant :
- b1 : si lim(s) < pdsmax, alors s est retirée de S_n (condition (w))
- b2 : si lim(s) ≥ pdsmax, alors on effectue la vérification de compatibilité entre s et p_n; si ce test est positif, une solution s' constituée par s ∪ p_n est ajoutée à S_n, avec un poids égal à pds(s) + pds(p_n).
Si pds(s) > pdsmax, alors pdsmax prend cette nouvelle valeur.

c) Si, après la phase b, on a lim(p_n) ≥ pdsmax, alors il peut exister des solutions maximales dérivées de p_n, et un élément de S_n est construit à partir de cette seule possibilité.

Puis on passe au pas n+1 avec la possibilité p_{n+1}, etc. L'algorithme s'arrête lorsque toutes les possibilités de P ont été traitées.

Remarque 1 : Durant la phase b) de la construction de S_n, il peut subsister des solutions telles que lim(s) < pdsmax car pdsmax est modifié dynamiquement durant ces opérations. Elles sont en fait supprimées au pas n+1 (évaluation de (w) en b1); on a préféré cette solution à celle, plus coûteuse consistant en fin de phase b), à éliminer toutes les solutions de S_n ne vérifiant pas (w) avec la nouvelle valeur de pdsmax. Cette opération est néanmoins nécessaire en fin d'algorithme, mais n'est exécutée qu'une fois sur un nombre minimal de solutions.

Remarque 2 : Si toutes les possibilités de départ sont incompatibles entre elles, l'espace final des solutions contiendra des éléments réduits à une seule possibilité de poids maximal.

Remarque 3 : Si les m possibilités de départ ont un poids nul (réduites à un seul sommet), l'algorithme donne comme résultat l'ensemble complet des parties de P (à la solution vide près). Toute combinaison de ces possibilités est en effet une solution (elles sont toutes disjointes deux à deux), de poids nul, et pdsmax est constamment nul. Il en résulte que (w) n'est jamais vérifiée, et qu'aucune solution partielle n'est éliminée.

Si cette configuration particulière est rencontrée, l'algorithme n'est pas appliqué et la solution proposée est égale à P. On décèle très simplement cette situation en remarquant que pdstotal a alors une valeur nulle.

Remarque 4 : Les solutions recherchées doivent être des graphes dont toutes les composantes connexes sont des sous-graphes n-partis complets; la propriété 3 de 2.7.3.b) permet de contrôler dans quelles conditions l'union d'une solution partielle $s \in S_{n-1}$ et de la possibilité p_n constitue un graphe vérifiant ces propriétés. Au pas n de l'algorithme, l'espace S_{n-1} des solutions est donc en fait partitionné en deux sous-ensembles S'_{n-1} et S''_{n-1} contenant respectivement les solutions vérifiant la propriété 3 et celles qui ne la vérifient pas. Lors de la construction de S_n, il peut y avoir génération de solutions partielles dans S'_n ou S''_n selon que la propriété 3 est vérifiée ou non; pdsmax est évalué relativement à S'_n, et, en fin de traitement, ses éléments de poids égal à pdsmax sont seuls retenus (ceux éventuellement existants de S''_n ne sont pas des solutions conformes à la définition donnée en 2.7.3.a)).

Selon la terminologie classique en intelligence artificielle, cet algorithme s'apparente à une méthode de recherche de type "breadth first" [B.NIL], la condition (w) intervenant pour limiter l'expression horizontale de l'arbre de recherche ("pruning"). Bien que relativement simple à implémenter, ses performances sont limitées et il reste à trouver des heuristiques plus efficaces. Un point d'amélioration important paraît le recours à un algorithme de type "branch and bound" susceptible de dériver moins de solutions partielles.

2.7.5.d) Second algorithme de sélection

Nous ne donnons ici que les principes essentiels relatifs à une implémentation possible d'un algorithme de type "branch and bound", pour résoudre le problème de la sélection des solutions de jumelage. L'intérêt principal de cette méthode, par rapport à la précédente, est qu'elle permet de limiter plus fortement, dans le cas général, le nombre de solutions partielles dérivées. Cet avantage tient au fait que, à un pas quelconque de l'algorithme, la solution la plus prometteuse est la seule à être traitée. Lorsqu'il s'avère que la solution en cours n'est plus dérivable (toutes les variables sont traitées) ou ne possède plus la meilleure estimation (bound), il y a branchement arrière (backtracking) vers un état antérieur plus prometteur, et réitération du processus.

En ce qui concerne notre problème particulier, on peut facilement vérifier que les conditions d'applicabilité de cette méthode présentées par A. THESEN dans [B.THE], sont vérifiées. Une différence notable toutefois réside dans les objectifs recherchés : nous désirons obtenir non pas une solution mais l'ensemble des solutions de jumelage, de poids maximal; ceci nécessite quelques adaptations de l'algorithme général.

Dans l'algorithme proposé ici, toute solution est représentée par un ensembl de quatre variables :

- lim : sa valeur correspond à la fonction d'évaluation pour la solution (cf ci-après)
- pds : poids effectif de la solution
- num : rang de la dernière possibilité traitée pour construire la solution (cf ci-après)
- setcomp : ensemble des composantes connexes constituant s.

Les choix d'implémentation de l'algorithme sont les suivants (cf B.THE) :

a) Ordre de sélection des possibilités

Comme dans l'algorithme précédent (et pour les mêmes raisons), les possibilit sont classées et traitées dans l'ordre décroissant de leur poids. Le rang de la dernière possibilité traitée est enregistré dans la description de la solution (num).

b) Traitement des possibilités - fonction d'évaluation (bound)

La fonction d'évaluation d'une solution s est celle donnée par (v) en 2.7.5.b)

lim(s) = pdstotal - c

où c est le poids total des possibilités rejetées (car non compatibles) durant la construction de s. Dans l'algorithme proposé, la solution traitée à un pas quelconque est toujours celle qui présente la valeur maximale de lim(s) (cf backtracking en c)). Notons s_n l'état de la solution s après le traitement des n premières possibilités, et considérons le pas n+1 où l'on tente de dériver s_{n+1} à partir de s_n et p_{n+1}; trois cas sont possibles :

b1) p_{n+1} n'existe pas
Toutes les possibilités ont été traitées pour dériver cette solution; elle est de poids maximal (cf ci-dessous) pdsmax. S'il reste des solutions partielles s telles que lim(s) ≥ pdsmax, il y a retour en arrière pour les traiter.

b2) p_{n+1} est compatible avec s_n :
Dans ce cas, s_{n+1} peut être dérivée de s_n et p_{n+1}, et s_{n+1} prend pour valeur :

- $lim(s_{n+1}) = lim(s_n)$: s_{n+1} reste la solution à dériver en priorité
- $pds(s_{n+1}) = pds(s_n) + pds(p_{n+1})$, avec modification éventuelle de pdsmax.
- $num(s_{n+1}) = num(s_n) + 1$: passage à la possibilité suivante.
- $setcomp(s_{n+1}) = setcomp(s_n) \cup p_{n+1}$: modification de l'ensemble des composantes connexes de s_n.

On considère ensuite l'éventualité de dériver une solution s_{n+1} sans p_{n+1}; cette solution partielle s'_{n+1} n'est créée que si elle peut conduire à une solution maximale, c'est-à-dire si :

$$lim(s'_{n+1}) = pdstotal - (c+pds(p_{n+1})) = lim(s_n) - pds(p_{n+1}) \geq pdsm$$

Si cette condition est vérifiée, s'_{n+1} est enregistrée avec les valeurs suivantes des variables descriptives :

$$lim(s'_{n+1}) = lim(s_n) - pds(p_{n+1})$$

$$pds(s'_{n+1}) = pds(s_n)$$

$$num(s'_{n+1}) = num(s_{n+1}) + 1$$

$$setcomp(s'_{n+1}) = setcomp(s_n)$$

b3) p_{n+1} et s_n ne sont pas compatibles :
Dans ce cas, s_{n+1} prend les mêmes valeurs que celles décrites ci-dessus pour s'_{n+1}. La fonction d'évaluation prenant alors une valeur moindre que celle que possédait s_n, il y a backtrack si s_{n+1} n'est plus la solution partielle la plus prometteuse, et s_{n+1} est éliminée si $lim(s_{n+1}) < pdsmax$.

c) Retour arrière

Lorsqu'il y a retour arrière (cf cas b1, b3 ci-dessus), la solution partielle considérée doit être celle jugée comme étant la plus prometteuse parmi les solutions pendantes; on envisage donc d'organiser l'ensemble des solutions sous forme de liste chaînée, triée par ordre décroissant des valeurs de la variable lim. Chaque fois qu'une solution est créée (b2), ou supprimée (cf ci-dessous), cette liste doit être mise à jour.

Lors de l'opération de backtrack, on ne considère que des solutions s telles que $lim(s) \geq pdsmax$; si aucune solution pendante ne vérifie cette propriété, ou s'il ne reste plus de solution à dériver, l'algorithme est terminé.

On voit donc que la condition (w) d'élimination des solutions garantit l'obtention de solutions de poids maximal. Les remarques 1, 2, 3, 4 faites pour l'algorithme précédent sont toujours valables, en particulier la remarque 4 à propos de notre contrainte supplémentaire d'obtention de solutions complètes. Cette contrainte est malheuresuement de nature à provoquer la dérivation de plus de solutions partielles (pdsmax est évalué restrictivement par rapport au sous-ensemble des solutions de jumelage), d'où l'importance de rechercher une méthode de sélection aussi efficace que possible.

2.7.6. Conclusion

Notre étude a tout d'abord porté sur une généralisation des contraintes de compatibilité, puis sur la caractérisation formelle des entités manipulées (possibilités, solutions) à partir de la notion de classe de cohérence et de classe d'homologie (le résultat du couplage, la réalité à reconstituer). Puis nous avons proposé une méthode d'évaluation du poids des possibilités et des solutions permettant la prise en compte d'un critère de qualité moyen pour s'orienter automatiquement vers des solutions fortement ou faiblement connexes. Ces propositions visent donc essentiellement une plus grande généralité de la méthode, ainsi qu'une amélioration de la qualité des résultats produits. Sur le plan algorihtmique, nous nous sommes tout d'abord contentés d'adapter l'algorithme de M. SKOLNICK à ces extensions, pour à présent nous préoccuper de rechercher des heuristiques et des méthodes de sélection plus efficaces. Comme souvent en pareil cas, seule l'expérimentation en vraie grandeur pourra indiquer quelle est l'implémentation globalement la plus performante (en espace mémoire et en temps d'exécution).

3.- PRÉSENTATION DES LOGICIELS RÉALISÉS

3.1. Objectifs Généraux

Au delà de leurs fonctions respectives, l'ensemble des logiciels présentés dans ce chapitre répondent à un certain nombre d'objectifs communs qui ont largement déterminé leur complexité au stade de la réalisation, mais aussi leurs modalités d'utilisation ainsi que leurs possibilités d'applications.

Dans le chapitre I, nous avons énuméré les principaux domaines d'application actuels d'un logiciel de reconstitution automatique de population ; tous concernent les Sciences Humaines et par conséquent des utilisateurs généralement non informaticiens. Dans ces disciplines, le recours croissant à l'informatique s'explique bien sûr par les possibilités de traitement rapide de masses d'informations très volumineuses, mais de plus en plus également par l'ouverture de nouveaux types d'investigations totalement inenvisageables par des processus manuels. Ce dernier point est certainement le plus fondamental dans la mesure où l'informatique apparaît alors comme un outil indispensable de progrès dans les disciplines concernées. Il n'en reste pas moins que cette évolution est souvent entravée par la forte dépendance dans laquelle se trouve souvent l'utilisateur vis à vis de l'informaticien ; cette dépendance se manifeste au niveau de la conception des logiciels (difficulté de communication, difficulté d'aboutir à des spécifications cohérentes et complètes), mais aussi lors de leur exploitation. Toute obtention d'un résultat implique le recours à l'informaticien seul apte à mettre en oeuvre le programme.

Un objectif essentiel que nous nous sommes fixé est donc de réduire cette dépendance en permettant une mise en oeuvre directe du logiciel par l'utilisateur final, tout au moins pour les phases principales d'une application au cours desquelles sa compétence est irremplaçable.

Ceci se traduit par la conception de logiciels conversationnels, caractérisés par des commandes de haut niveau exprimées dans un langage quasi naturel, et couvrant l'ensemble des fonctions-clé du processus [B.CH3]

Par ailleurs, nous avons également mentionné au chapitre 1 la multiplicité des tentatives de définition de logiciels autour de telle ou telle application ; outre la dispersion d'énergie et de moyens que cela implique, ces logiciels sont bien souvent trop limités (car élaborés trop sommairement), et donc relativement peu satisfaisants pour l'utilisateur qui doit encore fournir un important travail de validation et d'affinement des résultats fournis par la machine. Enfin, ils sont souvent trop liés à l'application de départ pour être réutilisables pour d'autres applications même très voisines.

L'étude générale développée au chapitre 2 nous a permis, dans une large mesure, de nous orienter vers la réalisation d'outils généraux, facilement adaptables à un large éventail d'applications. Cette généralité est assurée par une paramétrisation poussée, combinée à la recherche d'algorithmes et de structures de données aussi universels que possible [B.CH3].

Trois logiciels ont ainsi été réalisés, au moins au stade de prototypes :

- un logiciel de génération de programmes de contrôle des informations source
- un logiciel de transduction phonétique des noms : PIAFPHO.
- un logiciel de constitution d'une base de données démographiques : MERCURE.

Ces trois produits ont été réalisés séparément, dans des langages différents ; l'idéal serait qu'ils puissent être tous intégrés à MERCURE. Cela n'a pas été effectué pour des raisons techniques, notamment la non

disponibilité d'un langage de programmation pouvant efficacement prendre en compte des problèmes aussi éloignés que l'analyse de chaînes de caractères et la gestion d'une base de données. Indépendamment de ce problème, il est toutefois intéressant de pouvoir en disposer séparément, dans la mesure où leur généralité permet d'envisager leur utilisation dans un tout autre contexte que celui d'une application de reconstitution automatique de population. L'objet de notre étude concernant essentiellement le couplage d'informations floues, nous présentons de manière plus détaillée le logiciel MERCURE (en particulier au niveau algorithmique). Les deux autres ne seront présentés que de manière plus externe, au travers de leurs spécifications générales et leurs moyens de mise en oeuvre.

3.2. Eléments d'un formalisme algorithmique de type ensembliste :

3.2.1. Nécessité, intérêt d'un tel formalisme

Notre problème consiste à donner une présentation aussi claire que possible du système d'informations complexe mis en oeuvre par MERCURE, ainsi que des algorithmes de base utilisés lors des principales phases de traitement. Une telle présentation ne peut-être envisagée au travers des langages spécialisés du SGBD SOCRATE [*B.ABR*] qui a été utilisé pour l'implémentation de ce logiciel, en raison de leur caractère très spécifique. Par ailleurs, nous recherchons une présentation de haut niveau allégée d'un certain nombre d'informations d'intérêt secondaire relatives au choix d'implémentation, telles que les structures physiques et les méthodes d'accès correspondantes. Un langage tel que SETL [*B.MES*] répond, dans son principe, à ces préoccupations et nous en avons largement repris les possibilités, notamment en ce qui concerne les manipulations d'ensemble et de tuples

Il nous a cependant paru nécessaire d'introduire une large extension déclarative, inspirée des remarques de R. ENGMANN [B.ENG], et de recourir notamment aux définitions de types. Cette extension présente, à notre sens, le double avantage de permettre une définition claire des structures logiques manipulées, et de permettre, dans une certaine mesure , la spécification de contraintes d'intégrité entre ces entités. Ce dernier point permet de dégager la présentation des différents algorithmes de l'expression des traitements de vérification correspondants, et par là de les rendre plus aisément compréhensibles. Le mode de définition de la base de données conserve le caractère hierarchique-réseau [B.DAT] propre à SOCRATE ; la définition des traitements est largement inspirée du formalisme bien connu propre à PASCAL [B.PAS].
Remarquons que le recours à un modèle relationnel [B.COD], bien qu'en principe possible, n'a pas été retenu ici pour la raison essentielle qu'il aurait par trop déformé les algorithmes effectivement implémentés que nous cherchons à présenter. Enfin, il doit être clair que notre objectif n'est pas ici de proposer un langage algorithmique complet ; une telle tâche constitue en soi une recherche fondamentale totalement hors de notre sujet. Nous nous sommes limités à la définition des éléments strictement nécessaires à une présentation de haut niveau de MERCURE.

3.2.2. **Définition des données - Intégrité** :

On distingue deux classes de variables représentant l'ensemble des informations utilisées :

- Les variables locales aux procédures et aux fonctions ; elles sont définies comme en PASCAL dans une section déclarative spéciale (var)
- Les variables décrivant la base de données et sa structure logique ; elles sont définies hors du contexte de toute procédure ou fonction, et constituent l'équivalent de la définition de la structure d'une base en SOCRATE. Les types définis à ce niveau ont une portée globale.

Les déclarations correspondent au formalisme classique de PASCAL :

identificateur : *type*

Il est possible d'introduire, au stade déclaratif, des contraintes de dépendance entre les valeurs de plusieurs variables ; elles correspondent à des actions déclenchées (triggers) du type de celles existant dans SEQUEL [*B.ABL*]. Ces contraintes, définies statiquement, sont vérifiées et maintenues implicitement lors de toute modification des variables concernées. De nombreux types de contraintes peuvent être imaginés ; nous nous sommes limités à celui qui nous est le plus utile : les relations d'inclusion entre ensembles de même type :

exemple : id-ensemble 1, id-ensemble 2 : type ensemble
telque id-ensemble 1 $\subseteq$ id-ensemble 2

doit être interprêté comme suit :

- id-ensemble 1 désigne un sous-ensemble de id-ensemble 2
- Toute modification (ajout, suppression d'élément) de l'un de ces ensembles conserve cette propriété.

+++

Naturellement, les algorithmes correspondants ont été implémentés dans MERCURE ; la notation introduite ici permet simplement de considérer leur exécution comme implicite.

3.2.3. Définition des types

3.2.3. a) Introduction :

Les définitions de types peuvent-être introduites dans la définition générale de la base, ou au niveau d'une procédure ou d'une fonction.

Comme dans PASCAL, les définitions de type peuvent-être effectuées soit explicitement par une clause type, soit implicitement lors de la déclaration des variables. D'une manière générale, nous considérons qu'un type définit la propriété générique des éléments d'un ensemble référentiel particulier. Lorsque cela clarifie l'expression de la propriété, on note x tout élément du référentiel :

- *type nom-type = propriété (x) ; le référentiel est donné par {x| propriété (x) }*

Lorsque le referentiel ne peut-être formellement défini par une propriété il est défini par énumération de ses éléments :

- *type nom-type = x ∈ {éléments} ; le référentiel est donné par { éléments}*

Les types scolaires de base sont définis par les mots réservés entier, réel, booléen, chaîne ; la valeur notée 'u' représente la valeur indéterminée pour tous les types scalaires. Le type chaîne définit l'appartenance à un référentiel C^*, ou C est un ensemble fini de caractères.

La définition de sous-types scalaires est possible à partir des types de base ou d'autres types scalaires prédéfinis, par l'expression de conditions restrictives :

- exemple : type t1 = entier, $0 \leqslant x \leqslant 100$

Le référentiel est l'ensemble des entiers compris entre 1 et 100

type t2 = t1, x mod (2) = 0

Le référentiel est l'ensemble des entiers pairs compris entre 1 et 100

+++

Les types composites introduisent des propriétés particulières de structuration logique des éléments du référentiel ; on considère deux structures principales : les tuples et les ensembles.

3.2.3. b) Les tuples :

Comme dans SETL, les tuples définissent des suites ordonnées, avec répétition possible, d'éléments appartenant à des types composants. La valeur indéterminée correspondant à ce type est également notée nult (tuple de longueur nulle). Le référentiel défini par un type tuple est le produit cartésien des référentiels correspondant aux types composants.

Par analogie avec les notions classiques, nous distinguons deux classes de tuples :

- les structures
- les suites.

a) Les structures

Tous les champs composants sont identifiés, ils peuvent-être de types différents. Une structure est définie par une liste de composants placés entre parenthèses.

exemple : type t1 = (x : entier, y : booleen)

Le référentiel est le produit cartésien entier x booléen

+++

b) Les suites

Tous les champs composants sont de même type ; une suite est définie par le type commun de ses éléments, placé entre crochets. Le nombre de composants de la suite peut-être fixe ou variable ; dans le premier cas, la dimension est placée en exposant, et dans le second cas le symbole '*' est placé en exposant. Le nombre de dimensions n'est pas limité.

exemple = type t1 = [entier]^{3}

Le référentiel est le produit cartésien entier^{3}

Type t2 = [booléen] *

Le référentiel est le produit cartésien booléen^{n}, $\forall n \geq o$.

+++

Remarque : suivant les remarques de HOARE [B.HOA], KUSTER [B.KOS] et ROZEN [B.ROZ], il nous a paru intéressant d'autoriser la définition de types tuples récursifs ; ne nous préoccupant pas ici d'implémentation, cette possibilité est de nature à simplifier la définition des structures de données hierarchisées (biographies, familles, généalogies) ou réseau prépondérantes dans MERCURE, ainsi que la présentation des algorithmes qui les utilisent. Seules des définitions incohérentes telles que :

type t = $(x : t)$

sont interdites [B.KOS].

3.2.3. c) Le type ensemble :

Il est défini en plaçant entre accolades le type commun des éléments de l'ensemble ; le référentiel est alors l'ensemble des parties du référentiel correspondant à ce type commun.

exemple : type t1 = {entier}
Le référentiel de t1 est l'ensemble des parties de entier

Une variable de ce type a donc pour valeur possible toute partie de l'ensemble des entiers.

+++

On note $\emptyset$ l'ensemble vide ; cette valeur correspond à la valeur indéterminée pour le type ensemble. On peut définir des ensembles à partir de n'importe quel type de composant, et en particulier des ensembles d'ensembles :

exemple : type t1 = {{entier}}
var x : t1

x est un ensemble d'ensembles d'entiers.

+++

3.2.3. d) Union de types :

L'union de types est définie par l'opérateur ou placée devant une liste de propriétés correspondant à autant de types opérandes.

exemple : type t1 = ou (entier, booleen)

Le référentiel de t1 est l'union des référentiels correspondant à entier et booleen.

var x : t1, x peut prendre toute valeur entière ou booléenne.

+++

Inversement, il est souvent indispensable de connaitre le type composant t auquel appartient la valeur de la variable à un instant donné ; nous supposons que cela peut-être effectué grâce à la fonction logique suivante :

type (x : t)

qui prend la valeur vrai si et seulement si la valeur de x est du type t. Cette fonction peut-être utilisée pour la vérification de tout type.

3.2.4. **Procédures et fonctions** :

Fonctions et procédures sont représentées de manière similaire à PASCAL ; nous précisons cependant les conventions suivantes afin d'éviter toute erreur ou ambiguité dans leur interprétation :

Il n'y a pas de notion de variable globale ; toutes les variables non locales utilisées par une procédure ou une fonction, doivent figurer dans la liste des paramètres formels. Cette convention, bien qu'entraînant parfois une certaine lourdeur (quand il y a beaucoup de paramètres), permet toutefois une vérification rapide et sûre de la bonne utilisation des variables.

Les paramètres d'une procédure dont la valeur peut être modifiée lors de son exécution, sont soulignés.

Aucun paramètre d'une fonction ne peut être modifié au cours de son évaluation.

3.2.5. Les opérations élémentaires sur les variables :

3.2.5. a) Opérations sur les suites

Affectation :

Lorsqu'une suite de longueur variable est placée en partie gauche d'une affectation, sa longueur peut-être modifiée par cette opération. Dans le cas d'une affectation globale, elle prend pour longueur celle du tuple placé en partie droite.
Si une affectation porte sur un composant non existant de la suite (de rang supérieur à sa longueur courante), la longueur de la suite est étendue au rang du nouveau composant, et les positions intermédiaires éventuelles sont affectées à la valeur indéterminée correspondant au type de composants.

- longueur d'une suite :

La fonctions long (id-suite) délivre la longueur effective d'une suite (rang du dernier élément ayant une valeur définie). (0 si suite = nult)

- accès au premier élément défini d'une suite :

La fonction premier (id-suite) délivre le premier élément de valeur définie de la suite, ou la valeur indéterminée correspondant au type commun des composants si un tel élément n'existe pas.

- accès à l'élément défini suivant d'une suite :

La fonction suivant (id ϵ id-suite) prend pour valeur le composant suivant, de valeur définie, par rapport à celui représenté par id. Si cet élément n'existe pas (suite vide, fin de suite...), suivant prend la valeur indéterminée, correspondant au type commun des composants.

3.2.5.b) Opérations sur les ensembles

- affectation :

On reprend ici la plupart des possibilités offertes par SETL, notamment les générateurs d'ensembles ; nous nous limitons toutefois à la forme simple suivante :

id-ensemble : = {*id-élém.* ϵ *id-ensemble* | *prédicat*}

Les opérations classiques d'union, d'intersection, de différence d'ensembles sont autorisées.

- extraction aléatoire d'un élément

La fonction elem (id-ensemble) prend pour valeur un élément arbitraire de l'ensemble cité en paramètre, ou la valeur indéterminée correspondant au type des éléments, si l'ensemble est vide. On peut également extraire un élément, s'il existe, vérifiant une certaine propriété :

- elem (id-ensemble | prédicat)

- appartenance d'un élément à un ensemble :

L'expression logique id-elem ϵ id-ensemble prend la valeur vrai si et seulement si il existe un élément ayant pour valeur id-elem dans id-ensemble.

- vérification de l'existence et extraction d'un élément dans un ensemble :

La fonction existe (id-elem ϵ id-ensemble | prédicat) combine les deux opérations précédentes de vérification d'existence et d'extraction ; elle prend la valeur vrai si il existe un élément quelconque de l'ensemble vérifiant le prédicat, et cet élément est retourné dans la variable id-élém. Sinon la valeur prise est faux et id-élém à la valeur indéterminée correspondant au type des éléments de l'ensemble.

- ajout d'un élément dans un ensemble :

La procédure aj (id-elem ϵ id-ensemble) ajoute l'élément représenté par id-elem à l'ensemble.

- suppression d'un élément dans un ensemble :

La procédure supp (id-elem ∈ id-ensemble) supprime, s'il existe, l'élément représenté par id-elem dans l'ensemble.

- transfert d'un élément d'un ensemble dans un autre :

La procédure tr (id-elem ∈ id-ensemble-1 dans id-ensemble-2) transfère un élément, s'il existe, de ensemble-1 (où il est supprimé) dans ensemble-2 (où il est ajouté).

Ces trois dernières opérations, ainsi que l'affectation, supposent la vérification des contraintes éventuelles d'inclusion entre les ensembles concernés.

3.2.6. Instructions de choix

3.2.6 a) Incidence de la valeur indéterminée :

L'incidence de la valeur indéterminée dans l'évaluation des expressions logiques est résolue de manière classique (SEQUEL SOCRATE) par la définition d'une logique à trois valeurs : vrai, faux, u . On considère que toute expression élémentaire dont au moins un argument est indéfini, prend la valeur logique u. Pour les expressions composées, on considère une définition étendue des opérateurs logiques classiques, donnée par les tables de vérité suivantes :

ET	vrai	faux	u
vrai	vrai	faux	u
faux	faux	faux	faux
u	u	faux	u

OU	vrai	faux	u
vrai	vrai	vrai	vrai
faux	vrai	faux	u
u	vrai	u	u

NON	
vrai	faux
faux	vrai
u	u

3.2.6. b) Choix

Nous reprenons ici la notation algorithmique utilisée par J. COURTIN et J. VOIRON [B.COUV], inspirée de ALGOL 68 :

Si condition alors séquence-d'action finsi
Si condition alors séquence-d'action sinon séquence-d'action finsi

Dans le contexte de la logique à trois valeurs, si condition prend la valeur logique indéterminée, il y a branchement à l'instruction suivant le finsi.

3.2.6. c) Cas

Une règle analogue est appliquée au cas : pour chacune des conditions mutuellement exclusives déterminant l'exécution des différentes séquences d'actions :

Cas variable dans valeur-1 : séquence-1... valeur-n : séquence-n fincas

3.2.7. Itération

L'instruction pourtout a pour fonction la définition d'énumération d'ensembles (ou de sous-ensemble lorsqu'elle est assortie d'une condition filtrante); par analogie avec SOCRATE, on l'a étendue au parcours des éléments d'une suite.

3.2.7. a) Application à un ensemble :

L'énumération des éléments de l'ensemble s'effectue dans un ordre aléatoire ; seuls les éléments vérifiant la condition filtrante éventuellement spécifiée font l'objet de la séquence de traitement :

pourtout ident-élém ∈ ident-ensemble [| filtre] faire séquence finfair

Prouver que l'itération s'arrête revient à démontrer que l'ensemble énuméré est fini ; remarquons que cet ensemble peut-être modifié durant l'énumération. Si nous désignons par X l'ensemble, nous considérons qu'il est en permanence composé de deux sous-ensembles disjoints notés $\bar{X}$ et $\bar{\bar{X}}$ qui représentent respectivement les éléments déjà énumérés et ceux qui ne le sont pas encore (au départ, $\bar{X} = \emptyset$, $\bar{\bar{X}} = X$; à la sortie de l'itération, $\bar{X} = X$, $\bar{\bar{X}} = \emptyset$). Il est obligatoire de faire référence à ces sous-ensembles lors d'opérations d'ajout d'éléments dans X , en cours d'énumération :

- Si on ajoute les éléments dans $\bar{X}$, ils ne pourront être reconsidérés da l'énumération et il suffit de démontrer que X est fini.
- Si on ajoute les éléments dans $\bar{\bar{X}}$, ils seront considérés dans l'énumération, et il faut prouver que, au départ, X est fini, et que ces ajouts sont en nombre fini.

Dans tous les cas, l'instruction d'ajout aj (x ∈ x) en cours d'énumération de l'ensemble X est incohérente.

3.2.7. b) Application à une suite :

L'ordre d'énumération des éléments correspond à leur ordre d'occurence, de gauche à droite, dans la suite. Seuls les éléments de valeur définie sont considérés :

pourtout ident-élém. ∈ ident-suite [| filtre] faire séquence actions finfaire

Contrairement au cas précédent de l'énumération d'ensembles, il est interdit de modifier le nombre d'éléments de la suite (si elle est de dimension variable bien sûr) au cours d'un parcours défini par pourtout.

3.2.7. c) Sortie :

On peut interrompre une énumération pourtout par l'exécution de l'instruction sortie. Cette commande provoque le branchement à l'instruction suivant le finfaire correspondant à l'itération interrompue.

3.2.7. d) Fonction rang :

Au cours de l'énumération d'une suite, il est possible d'obtenir le rang de l'élément en cours de traitement par la fonction :

rang (id-elem ∈ id-suite)

3.2.7. e) Instruction tantque :

En plus de l'instruction d'énumération pourtout présentée ci-dessus, on dispose de l'instruction classique d'itération :

tantque condition faire séquence d'actions finfaire

Lorsque condition prend la valeur indéfinie, il y a branchement à l'instruction suivant finfaire.

CHAPITRE 3.3 Présentation du logiciel MERCURE

3.3. Présentation du logiciel MERCURE

3.3.1. Introduction

Le logiciel MERCURE a été conçu pour permettre le couplage et le jumelage interactifs d'informations floues, dans le cadre d'applications de reconstitution automatique de populations (cf. 3.4.). Les aspects méthodologiques mis en oeuvre sont ceux qui ont été développés dans le chapitre II ; au niveau de réalisation actuel, il doit être considéré comme un prototype ayant servi à divers essais d'implémentation. Les évaluations effectuées à partir de ce prototype ont essentiellement porté sur la fonctionnalité des commandes proposées, et sur les diverses possibilités d'implémentation des algorithmes proposés au chapitre II. Sur ce dernier point, l'utilisation du système de gestion de base de données SOCRATE [B.ABR] s'est avérée extrêmement efficace.

Le projet MERCURE a suivi l'évolution des différentes versions de ce système, depuis les premiers prototypes développés sous la direction de J.R. ABRIAL dans le cadre du Laboratoire d'Informatique et de Mathématiques Appliquées de Grenoble (Laboratoire IMAG), jusqu'aux versions industrielles disponibles par la suite au Centre Interuniversitaire de Calcul de Grenoble.

La présentation qui suit concerne tout d'abord les caractéristiques générales de MERCURE, ainsi que l'organisation de ses différentes fonctions. Les chapitres 3.3.3. à 3.3.6. concernent les modalités particulières de mise en oeuvre des principales fonctions ; les chapitres 3.3.7. à 3.3.9. présentent les principaux algorithmes utilisés, conformément au formalisme défini dans le chapitre 3.2.

Enfin, nous résumons en 3.3.10. les modalités générales d'implémentation ainsi que les restrictions d'utilisation du logiciel.

3.3.2. Les caractéristiques générales de MERCURE

En tant qu'outil de reconstitution automatique de population, MERCURE présente un caractère de généralité qui est fondé sur deux traits essentiels :

- La mise en oeuvre d'une méthodologie générale de couplage, de jumelage des informations de départ, telle qu'elle a été définie au chapitre II.
- L'adaptabilité du logiciel à toute application particulière ; cette caractéristique particulière a été obtenue par la définition d'un noyau commun à toutes les applications, et celle d'outils de définition des paramètres propres à chaque application (le modèle d'une application)

Par ailleurs, toutes les commandes de MERCURE, et en particulier celles permettant de définir ou modifier les différents paramètres, sont exécutables en mode conversationnel. On a donc la possibilité de mettre au point ce modèle de manière interactive, en évaluant sa définition sur autant d'exemples qu'on le désire ; l'ensemble de ces opérations peuvent-être effectuées sans affecter les données de départ.

MERCURE est également un logiciel de gestion de base de données ; cela est indispensable compte-tenu de la complexité des structures manipulées (biographies, familles, généalogies), et de la multiplicité des fichiers de base impliqués dans ces diverses structures (cf. 3.4.).

L'intégration de l'ensemble des informations dans une base de données unique permet en outre leur exploitation à différents niveaux (cf. 3.4.), et simplifie considérablement les investigations opérées sur la population reconstituée.

Enfin, comme nous nous le sommes fixé au chapitre 3.1., l'ensemble des fonctions d'exploitation de MERCURE sont accessibles à des utilisateurs non-informaticien, par le biais de commandes de haut niveau exprimées selon un langage quasi naturel. L'activation de MERCURE, s'effectue par la commande :

MERCURE nom-application ? ; on a alors accès à quatre environnements de travail :

a) La définition modification des paramètres :

- La commande PAR permet d'accéder à l'environnement "paramètres" qui regroupe l'ensemble des commandes de définition/modification/visualisation des paramètres constituant le modèle de l'application (cf. 3.3.3.).

b) Le chargement des données dans la base :

- La commande DON permet d'accéder à l'environnement "données" qui regroupe l'ensemble des commandes liées au chargement des données source dans la base (cf. 3.3.4.).

c) Les opérations de couplage :

- La commande COU permet d'accéder à l'environnement "couplage" qui regroupe l'ensemble des commandes permettant le couplage des fichiers de départ (cf. 3.3.5.).

d) Les opérations de jumelage :

- La commande JUM permet d'accéder à l'environnement "jumelage" qui

regroupe l'ensemble des commandes permettant le jumelage des classes de cohérence constituées durant la phase de couplage (cf. 3.3.6.).

3.3.3. Les paramètres de définition d'une application :

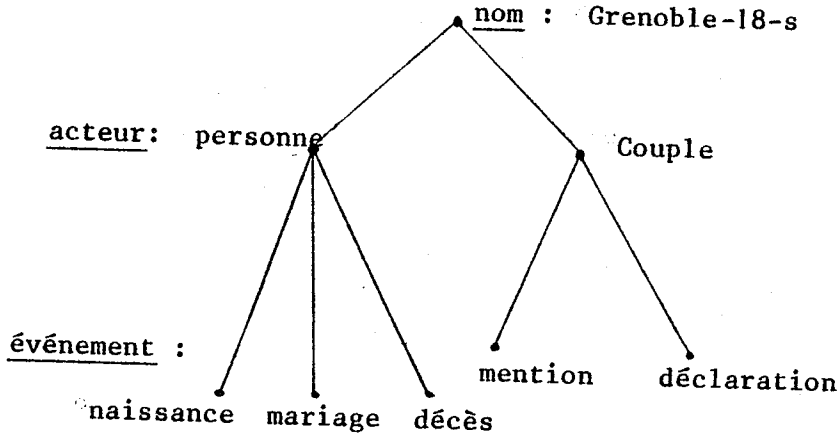
3.3.3. a) Les paramètres de type "donnée" :

La définition du modèle relatif à une application comporte la spécification de quatre types de paramètres correspondant à des données enregistrées dans la base :

a) La structure logique de l'application :

La définition de ce paramètre peut-être assimilée à une déclaration ; à son plus haut niveau, toute application est définie par une arborescence du type donné par l'exemple ci-dessous (cf. 3.4.) :

Exemple :



Qui est interprêtée ainsi : l'application de nom 'Grenoble-18-s' comporte le traitement de deux types d'acteurs : les personnes et les couples ;

les acteurs de types "personne" sont répartis sur trois fichiers source : les naissances, les mariages, les décès. Les acteurs de type "couple" sont répartis sur deux fichiers source : les mentions de couples et les déclarations de couples (cf. 3.4.).

+++

La structure correspondante dans la base fait partie du noyau de MERCURE ; elle donne en fait une définition plus complète de l'application, que nous préciserons plus loin. On peut considérer, à ce niveau, qu'elle est définie par la déclaration suivante :

(a) var APPLICATION : (NOM :chaîne ; V-TACT : [t-act]^2)

avec :

(b) type t-act = (TYPE : chaîne ; V-TEVEN : [t-even]^N; <suite-tact >)
t-even = (TYPE : chaîne ; <suite-t-even >)

<suite-tact> et <suite-teven> seront détaillés plus loin ; N représente le nombre de fichiers à coupler pour le type d'acteur considéré. Le traitement de ces paramètres est défini en 3.3.4. a.

b) La structure des données de départ :

Pour permettre le chargement de la base, on doit définir trois paramètres :

. la description des fichiers source (produits par le logiciel de contrôle (cf. 3.5.).

. la description des fichiers "acteur" et "évenement" de la base ; ces deux fichiers correspondent à un éclatement des informations source en

deux catégories d'informations :

- un enregistrement de type "acteur" regroupant toutes les informations logiquement invariantes (cf. 2.1.3.d).
- un enregistrement de type "événement" regroupant toutes les informations non invariantes.

. la définition de dictionnaires de valeurs des différentes caractéristiqu

Chaque dictionnaire correspond à une classe de caractéristiques synonymes (cf. 2.1.3.b) ; il permet une définition unique de l'ensemble des valeurs de ces caractéristiques, et la définition éventuelle de modèles de variati (cf. 2.4.2.a et 3.3.5.a ci-après).

Exemple : Soit un fichier source d'actes de naissance, dont les éléments sont du type :

Type : acte-naiss = (CODE, NOM, PRENOM, NOM-MERE : chaîne ; DATE : date ; LIEU : chaîne)

- Lors du chargement de ce fichier dans la base, on définit :
  - le type acteur = (CODE, NUM, PRENOM, NOM-M : chaîne) regroupant les caractéristiques logiquement invariantes
  - le type naissance = (DATE : date ; LIEU : chaîne) regroupant les caractéristiques non invariantes.
- Tout enregistrement du fichier source est éclaté en deux enregistrements de type acteur et naissance respectivement ; cette restructuration des données de départ peut-être représentée par le type acte-naissance suivan

type acte-naissance : (PERSONNE : acteur; EVENEMENT : naissance)

Par ailleurs, il faut définir les dictionnaires de valeurs correspondant aux caractéristiques : NOM et NOM-M qui sont synonymes

PRENOM

DATE

LIEU

Voir la remarque ci-dessous pour la caractéristique CODE.

+++

Remarque importante :

Dans la version actuelle de MERCURE, le type acteur est standard ; la caractéristique CODE correspond à une information discriminante fournie lors de la saisie, et utilisée par la suite pour accéder sans ambiguité à tout acteur particulier.

c) Les paramètres de couplage :

Ils consistent en la définition du processus de couplage des différents fichiers correspondant à un type d'acteur particulier. On peut considérer deux classes de paramètres de couplage :

. la définition des relations de cohérence large (cf. 2.4.3.) et celle des relations élémentaires entre caractéristiques, qui les composent (cf. 2.1.3.e). Suivant la remarque faite en 2.5.2.c, toute relation de cohérence large R^{ij} définie entre les fichiers E^i et E^j, est décomposée

en deux parties :

$$R^{ij} = R'^{ij} \cap R''^{ij}$$

ou R'^{ij} regroupe toutes les relations définies entre caractéristiques invariantes (donc entre enregistrements de type "acteur"), et R''^{ij} regroupe les relations définies entre caractéristiques non invariantes (donc entre enregistrements de type "évènement").

Dans MERCURE, toute relation de cohérence large, appelée aussi relation de couplage, est définie selon ce principe de décomposition. Les procédures d'évaluation sont engendrées par des langages spécialisés.

A partir de ces outils, on définit toute relation de couplage comme une suite de relations externes élémentaires ; chacune d'elle porte un nom, et correspond à un ensemble d'informations facilement accessibles appelées paramètres d'évaluation de la relation (cf. 3.3.5.a). Toute relation élémentaire possède un statut qui permet d'activer, de supprimer son évaluation dans la relation de couplage (pour juger des effets produits sur ce processus) sans recompiler la procédure correspondante. Une troisième possibilité de statut est la neutralisation ; la relation continue d'être évaluée mais ne fait plus logiquement partie de la relation de couplage (elle n'est plus filtrante). Si elle est vérifiée, le poids qui lui est associé est ajouté à celui des autres relations élémentaires, sinon elle n'a aucun effet. Ce type de statut est utile dans le cas de relations entre caractéristiques non-invariantes ; il permet de les utiliser comme renforcement de la mesure de cohérence.

D'autres paramètres d'évaluation sont accessibles directement au niveau des relations élémentaires, comme la définition (empirique ou non) de mesures globales, la définition d'intervalles de variations pour des données numériques (dates par exemple), etc...

Certaines relations élémentaires peuvent-être des modèles de variation (cf. 2.4.2.a) ; pour des raisons de coût, nous nous sommes limités à la définition d'un maximum de deux modèles de variation indépendants (cf. 2.4.2.d) pour une relation externe donnée.

La définition des modèles de variation est developpée en 3.3.5.a.

- La définition des critères d'évaluation propres à chaque relation élémentaire :

  . Pour chaque relation élémentaire, il est possible de définir l'un des deux types de mesure suivants (cf. 2.6.3.) :

    - une mesure globale : le poids affecté à la relation est indépendant des valeurs particulières pour lesquelles la cohérence est constatée.
    - une mesure fine : le poids affecté à la relation dépend des valeurs particulières pour lesquelles la cohérence est constatée.

- La définition de ces paramètres est présentée au chapitre 3.3.5.a.

d) Les paramètres de Jumelage :

Ils se réduisent à la définition, pour un type d'acteur donné :

- du facteur de qualité Q (cf. 2.7.4.b)
- des règles de compatibilité (cf. 2.7.2.b), qui sont représentées par une matrice carrée MATCOMP correspondant au type :

(c) type mat = [(MIN, MAX : entier)]^{N,N}

Remarque : la dimension N x N correspond au nombre maximal de fichiers à coupler pour un type d'acteur donné, défini par (b)

Nous pouvons donc compléter la définition (b) du type t-act :

(d) type t-act = (TYPE : chaîne ; V-TEVEN : [t-even]^{N} ; Q : réel ;
MATCOMP : mat ; <suite-t-act>)

La définition de ces paramètres est présentée au chapitre 3.3.6.a.

3.3.3. b) Les paramètres de type "procédure" :

En plus des données mentionnées au chapitre précédent, la définition du modèle d'une application comporte celle d'un certain nombre de procédures non standard qui correspondent aux fonctions suivantes :

a) - chargement de la base à partir des fichiers source

b) - édition de tout ou partie de ces données

c) - définition des caractéristiques d'accès rapide pour la phase de couplage

d) - définition des relations de couplage entre caractéristiques invariantes.

e) - définition des relations de couplage entre caractéristiques non invariantes.

Les procédures d) et e) sont destinées à prendre en compte les données définies au niveau des relations de couplage et des mesures correspondant (cf. paramètres de couplage dans le chapitre précédent). Un point importa à noter ici, est que la définition de ces procédures est effectuée à l'ai de langages de haut niveau spécialement conçus à cet effet (cf. 3.3.4.a et 3.3.5. a) ; comme on le verra plus loin, cette tâche est ainsi extrême-ment simplifiée.

3.3.3. c) Remarques générales relatives à l'existence des deux types de paramètres :

La répartition des paramètres en deux types d'informations (des données enregistrées dans la base et des procédures) correspond à la recherche d'un certain équilibre entre la commodité de définition/modification des paramètres, et les performances des programmes d'exploitation. Toute donnée enregistrée est facilement modifiable lors de la mise au point du modèle ; par contre sa consultation permanente lors des traitements peut être coûteuse.

Inversement, la programmation directe d'une logique de traitement conduit à de meilleures performances à l'exécution, mais toute modification nécessite une recompilation de la procédure concernée. Dans la mesure où la phase de mise au point du modèle est généralement délicate et nécessite de nombreuses rectifications successives, nous avons voulu lui assurer une certaine souplesse en conservant, sous forme de données, les paramètres susceptibles d'être le plus souvent modifiés. Ceci est important si l'on désire rendre la définition du modèle accessible à un non informaticien.

3.3.4. Initialisation d'une application

L'ensemble des opérations qui suivent doivent être effectuées par un informaticien.

3.3.4. a) Définition de la structure des fichiers et de l'application

La définition des fichiers source ainsi que celle des fichiers "événement" doit être effectuée directement en langage SOCRATE (cf. 3.3.10).

Rappelons (cf. remarque en 3.3.3.a) que le type "acteur" est standard pour toute application de reconstitution automatique de population :

(e) type acteur = ou (act-personne, act-couple)
avec

(f) type act-personne = (CODE : chaîne; D_PERSONNE : personne ;
PARENTS : couple ;

UNION : {(CONJOINT : act-personne ; COUPLE-D : act-couple ;
EVEN : mariage)};

BIOGRAPHIE : {événement})

Le type "personne" correspond à la description particulière adoptée dans l'application considérée.

Le type "évenement" correspond à l'union de tous les types d'événements traités pour les personnes, sauf les mariages qui sont considérés séparément (EVEN).

(g) type act-couple = (CODE : chaîne ; EPOUX, EPOUSE : act-personne ;
ENFANTS : {act-personne} ;
BIOGRAPHIE : {événement })

On trouvera également au chapitre 3.4. l'utilisation des variables de ce type dans le processus de reconstitution de la population.

A cette phase de définition de **fichiers** doit succéder la définition de la structure logique de l'application (création de la structure arborescente définie par (a)) ; cette opération s'effectue par le biais des commandes suivantes, dans l'environnement de travail "paramètres" :

D-APL nom application ? * affectation du nom de l'application *
D-TACT type-acteur ? * définition du niveau type acteur *
D-TEVEN type-even DE type-acteur ? * définition des types d'événements relatifs à un type d'acteur donné *

Exemple - Création de la structure arborescente définie par la figure 1 en 3.3.3. a :

D-APL GRENOBLE-18-S?
D-TACT PERSONNE ?
D-TEVEN NAISSANCE DE PERSONNE ?
D-TEVEN MARIAGE DE PERSONNE ?
D-TEVEN DECES DE PERSONNE ?
D-TACT COUPLE ?
D-TEVEN MENTION DE COUPLE ?
D-TEVEN DECLARATION DE COUPLE ?

+++

Il est possible d'obtenir l'édition de l'état de toute ou partie de l'application grâce aux commandes suivantes :

I-APL ? * édite l'arborescence définissant l'application, ainsi que l'état correspondant des données par type d'acteur et par type d'événement ;

- nombre total d'acteurs par type
- taux de couplage pour le type considéré

- nombre de classes de cohérence construites
- effectif de chaque fichier événement

La commande :

I-ACT type-acteur ?

restreint cette édition au type d'acteur fourni en paramètre

La commande :

I-EVEN type-événement DE type-acteur ?

restreint cette édition à un type d'événement relatif au type d'acteur fournis en paramètre.

Les commandes de suppression de tel ou tel élément de la définition de l'application existent également ; elles ne sont utilisées qu'en cas extrême car elles s'accompagnent obligatoirement de la perte de tout ou partie des informations source :

SD-TACT type-acteur PW mot-de-passe ? * supprime les données relatives au type d'acteur fourni en paramètre *

S-TACT type-acteur ? * supprime "type-acteur" de la définition de l'application ; elle est ineffective si des données de ce type ont été chargées (sécurité) *

SD-TEVEN type-événement DE type-acteur PW mot-de-passe ? * restreint le traitement de SD-TACT à un type d'événement particulier *

S-TEVEN type-événement DE type-acteur ? * restreint le traitement de S-TACT à un type d'événement particulier, avec les mêmes conditions de sécurité *.

3.3.4. b) Structure des dictionnaires de valeurs

Chaque dictionnaire correspond à une classe de caractéristiques synonymes (cf. 3.3.3.a), leur structure est standard et est définie par le type suivant :

(h) type dictionnaire = (NOM : chaîne ; TABSYN : [ID-CAR : chaîne ; NB-VAL : entier)] ^{2,N,2} ; NB-VAL : entier ; SET-VAL : { valeur })

où : NOM est le nom du dictionnaire

- TABSYN est un tableau à trois dimensions :

  - dimension 1 : rang du type d'acteur considéré dans APPLICATION.V-TACT (cf.(a) en 3.3.3.a), soit I $\in$ [1,2]
  - dimension 2 : rang du type d'événement considéré dans V-TAC[I].V-TEVEN (cf.(b) en 3.3.3.a) soit J $\in$[1,N.].
  - dimension 3 : numéro d'ordre de la caractéristique élémentaire dont l'identificateur est ID-CAR, dans le fichier défini par I et J ; soit K, qui a été limité ici à l'intervalle [1,2]. Cela implique que, dans un fichier source donné, toute classe de synonymie comporte au maximum deux éléments ; cette valeur peut-être modifiée pour une application particulière.

- Le triplet (I,J,K) définit le niveau de synonymie de toute caractéristique appartenant au dictionnaire. Pour chacune d'elles, NB-VAL contient le nombre de valeurs de la caractéristique.

- SET-VAL représente l'ensemble commun des valeurs de la classe de caractéristiques synonymes.

- Les valeurs de tout dictionnaire sont enregistées dans un fichier dont les éléments sont du type :

(i) type valeur = (VAL : ou (chaine, entier) ; MES-CARD :[(MESURE :réel
CARD : entier)]^{2,N,2} ;
<suite-valeur>)

ou : - VAL est la valeur de la caractéristique, limitée aux types chaîne et entier.

- MES-CARD [I,J,K].MESURE : donne, pour la valeur considérée de la caractéristique correspondant au niveau de synonymie (I, J,K), la mesure de WIENER (cf. 2.6.2.) correspondant à sa fréquence dans le fichier défini par (I,J)

- MES-CARD [I,J,K].CARD donne pour la valeur considérée de la caractéristique correspondant au niveau de synonymie(I, J, K), le nombre d'acteurs du fichier défini par (I, J) possédant cette valeur de la caractéristique, soit $N_{I,J}$

- On a donc :

$$\text{MES-CARD [I,J,K].MESURE} = -\log_2 (\text{MES-CARD [I,J,K].CARD}/N_{I,J})$$

- <suite-valeur> réunit des informations relatives à d'éventuels modèles de variation définis sur l'ensemble des valeurs.

3.3.4. c) Définition - Edition des dictionnaires :

Le noyau de MERCURE comporte la définition d'un ensemble de dictionnaires, soit :

(j) var SET-DICT :{ dictionnaire}

Les commandes qui suivent sont relatives à la création ou l'édition des éléments de cet ensemble.

a) Création d'un dictionnaire :

Elle s'effectue par la commande :

D-DICT nom-dictionnaire ?

où <nom-dictionnaire> est une chaine alphanumérique de longueur ≤16, qui définit un nom commun à tous les identificateurs de caractéristiques appartenant à la classe de synonymie. Cette commande provoque l'engagement d'un dialogue au cours duquel on demande à l'utilisateur de définir l'ensemble des éléments de la classe des synonymie par le triplet :

- type d'acteur
- type d'événement
- nom de la caractéristique

Pour un dictionnaire x de SET-DICT, chacun de ces triplets permet l'affectation de x.TAB-SYN (cf. (h)). L'ensemble des valeurs est engendré lors du chargement des données (cf. 3.3.4. d ci-après).

b) Edition des dictionnaires :

La commande :

I-DICT nom dictionnaire ?

provoque l'édition de la définition du dictionnaire dont le nom est donné en paramètre, soit :

- les triplets type-acteur, type événement, identificateur définissant chaque élément de la classe de synonymie, ainsi que le nombre de valeurs enregistrées pour chacune d'elles.

- le nombre total de valeurs contenues dans le dictionnaire.

c) Edition des valeurs des caractéristiques :

La commande :

I-VAL nom-dictionnaire ?

permet l'édition de l'ensemble des valeurs du dictionnaire ; chaque valeur est affichée, suivie de celle du tableau MES-CARD correspondant (cf (i))

La commande :

IR-VAL nom dictionnaire ?

permet l'édition des valeurs d'un élément particulier de la classe de synonymie ; un dialogue est engagé permettant de spécifier le triplet type acteur, type-événement, identificateur définissant la caractéristique concernée.

D'autres commandes de travail sont disponibles que nous ne détaillons pas ici.

d) Suppression d'un dictionnaire :

Cette opération a été prévue en cas d'erreur ou d'oubli dans la définition d'un dictionnaire ; elle n'est effective que si le dictionnaire n'a pas encore été chargé (ensemble des valeurs vide).

S-DICT nom dictionnaire ?

3.3.4. d) Chargement de la base :

Cette opération nécessite la définition préalable de procédures de chargement adaptées au format des données source. Dans l'état actuel de MERCURE, seule une saisie conversationnelle des données a été implémentée (prototype) ; la définition d'un langage analogue à celui présenté ci-dessous, pour un chargement par lots ne poserait pas de grande difficulté.

a) Langage de définition des procédures de chargement

On définit deux procédures :

- la procédure de chargement d'un acteur de type "personne".
- la procédure de chargement d'un enregistrement de type "événement".

Saisie des acteurs - syntaxe :

Saisie-acteur : {SAIS-ACT type-acteur {caract-acteur} $^{+}$ FI}$^{+}$

caract-acteur : SVAL-ACT nom-dictionnaire nom-caractéristique

Saisie des événements - syntaxe :

Saisie-événement : {saisie-type-événement}$^{+}$

Saisie-type-événement : SAIS-EVEN type-even {caract-even} $^{+}$ FI

Caract-even : SVAL-EV nom-dictionnaire nom-caractéristique

Exemple : Supposons la définition suivante des types "personne", "naissance", "mariage", décès :

```
type  personne = (NOM, PRENOM, NOM-MERE : chaîne)
      naissance = (DATE-N : date ; LIEU-N : chaîne)
      mariage   = (DATE-M : date ; LIEU-M : chaîne ; PROFESSION
                   chaîne)
      décès     = (DATE-D : date ; LIEU-D : chaîne ; METIER :
                   chaîne)
```

On aura défini les dictionnaires :

NOM : regroupant NOM et NOM-MERE pour les actes de naissance, mariage, décès

PRENOM : regroupant les caractéristiques PRENOM des trois types d'actes

DATE : regroupant DATE-N, DATE-M, DATE-D

LIEU : regroupant LIEU-N, LIEU-M, LIEU-D

METIER : regroupant METIER et PROFESSION

La saisie conversationnelle de ces données sera réalisée à partir des deux procédures définies de la manière suivante :

- Saisie des acteurs "personne" :

```
SAISACT  PERSONNE
         SVAL-ACT NOM NOM
         SVAL-ACT PRENOM PRENOM
         SVAL-ACT NOM NOM-MERE

FI
```

- Saisie des événements :

```
SAIS-EVEN  NAISSANCE
           SVAL-EV  DATE  DATE-N
           SVAL-EV  LIEU  LIEU-N
```

```
FI
SAIS-EVEN MARIAGE
        SVAL-EV  DATE  DATE-M
        SVAL-EV  LIEU LIEU-M
        SVAL-EV  METIER PROFESSION

FI
SAISEVEN DECES
        SVAL-EV  DATE DATE-D
        SVAL-EV  LIEU LIEU-D
        SVAL-EV  METIER METIER

FI
```

+++

b) Commandes de chargement de la base :

Dans le contexte de travail "données", on lance la saisie interactive des données par la commande :

E-ACTE type-événement DE type-acteur ?

Le dialogue est alors engagé, et on entre, au terminal, les différentes caractéristiques correspondant à la définition du type d'acteur et du type d'événement mentionné.

On termine la saisie en répondant "N" à la demande de création d'acteur.

Exemple : entrée des actes de naissance :

E-ACTE NAISSANCE DE PERSONNE ?

+++

- Remarques : La saisie de la caractéristique CODE (cf (f)) obligatoire pour tout type d'acteur, est générée automatiquement

Les données saisies ont souvent une structure plus complexe que celles données en exemple auparavant ; un exemple en est donné par les actes de mariage regroupant des informations relatives aux deux conjoints. La saisie des données doit alors s'accompagner de l'établissement de liens généalogiques (dans le cas du mariage, la relation "conjoint" cf (f)) entre plusieurs personnes.

La commande :

E-COUPLE type-événement DE type-acteur ?

a été définie pour la saisie des couples.

Un développement important de MERCURE consisterait à définir un langage de saisie pouvant traiter des structures plus complexes (familles, recensements...).

3.3.4. e) Edition des données enregistrées dans la base :

De manière analogue à la définition des procédures de saisie, il est possible de définir des procédures d'édition des données enregistrées ; nous ne détaillons pas ici le langage et les commandes correspondant à cette fonction.

3.3.5. Les commandes relatives au couplage des informations :

3.3.5. a) La définition des relations de couplage :

a) Les paramètres d'évaluation des relations externes :

. Toute relation externe est munie d'un nom et d'un certain nombre de paramètres d'évaluation (cf. 3.3.3.a) ; on peut définir comme suit ces informations :

(k) type relation : (NOM : chaîne, STATUT : {A,S, N}, MESURE-G : réel ; BIINF, BSUP ; réel ; MODELE, MODELE-C : modèle ; BORNE : réel ; <suite-relation>)

Un ensemble de relations de ce type est défini dans le noyau de MERCURE :

var SET-REL : {relation}

La plupart des commandes d'accès à ces paramètres sont très simples :

D-REL nom-rel ? * crée une relation de nom "nom-rel" *
I-REL nom-rel ? * édite le contenu des paramètres d'évaluation *
A-REL nom-rel ? * affecte le statut de la relation à "active" (A) *
S-REL nom-rel ? * affecte le statut de la relation à "supprimée" (S) *
N-REL nom-rel ? * affecte le statut de la relation à "neutralisée"(N) *

MES-G nom-rel = reel ? * affecte la mesure globale à réel ≥ 0 *

INTERVALLE nom-rel ENTRE binf ET bsup ? * affecte BIINF et BSUP *
BORNE nom-rel = reel ? * affecte BORNE *

d'autres, comme celles liées à l'utilisation des modèles de variation sont plus complexes, nous ne les présentons pas ici car elles en sont encore au stade expérimental.

b) Les langages de spécification des relations de couplage :

Quatre procédures sont à définir (cf. 3.3.8 b) :

- Les relations de couplage entre acteurs, par type d'acteur.
- Les relations de couplage entre évènements, pour un type d'acteur.
- La sélection des critères d'accès rapide pour la construction des coupes.

La syntaxe est la suivante pour les relations entre acteurs :

compar-act : {COMPARAISON-ACT type-acteur {comp-act} $^{+}$ FI}$^{+}$

comp-act : EQ-ACT nom-rel DE nom-caract |

* modèle * EQM-ACT nom-rel DE nom-caract PAR nom-car-red |

* modèles indépendants * EQMC-ACT nom-rel DE nom-caract PAR nom-car-red-1 ET-PAR nom-car-red-2

. Exemple :

```
COMPARAISON-ACT PERSONNE

        EQM-ACT R1 DE NOM PAR NOM-R1
        EQ-ACT R2 DE PRENOM

FI
```

Pour les acteurs de type "personne", la relation de couplage entre caractéristiques invariantes est définie par l'égalité des noms via un modèle spécifié dans R1, la variable comparée étant NOM-R1 (codification des noms d'après le modèle cf.3.3.5.b) et l'égalité du prénom dont les paramètres d'évaluation sont accessibles dans R2 (cf. chapitre précédent).

+++

- Pour la comparaison des événements on a :

compar-év : {COMPARAISON-EV type-even ET type-even {comp-ev }$^{+}$ FI }$^{+}$

comp-ev : EQ-EV nom-rel DE nom-car |

* modèle * EQM-EV nom-rel DE nom-car PAR nom-red |

* modèles indépendants * EQMC-EV nom-rel DE nom-car PAR nom-red-1 ET-PAR nom-red-2

TEST nom-rel nom-car opération comparaison BORNE

Opération : DIFFERENCE |SOMME| PRODUIT | RAPPORT

Comparaison : INFEG |SUPEG | INF | SUP | EG | COMPRIS

Exemple :

COMPARAISON-EV NAISSANCE ET MARIAGE
EQM-EV R3 DE LIEU PAR LIEUR
TEST R4 DATE DIFFERENCE COMPRIS BORNE

FI

Pour les événements NAISSANCE et MARIAGE, il y a comparaison des lieux via un modèle (défini dans R3) ; la variable comparée directement est LIEUR (cf. 3.3.5.4.). Il y a également comparaison des dates (en années) par vérification de l'âge au mariage dans un intervalle donné dans R4 (cf. commande INTERVALLE).

+++

Pour les mesures, on se limite ici à indiquer la liste des relations élémentaires pour lesquelles on désire que l'évaluation soit effectuée (le mode d'évaluation est entièrement spécifié au niveau des relations $R1...R_n$)

bloc-mesure : {EVALUATION type-even ET type-even {mesure }$^{+}$ FI }$^{+}$

mesure : MESURE nom-rel

Enfin, la procédure de définition des critères d'accès rapide est entièrement définie par la simple clause :

SELECTION nom-car-1 OU nom-car-2

Exemple :

SELECTION PRENOM OU NOM

Le critère d'accès rapide primaire est le PRENOM ; si sa valeur est indéfinie pour un acteur donné, la recherche est lancée à partir du NOM ; si les deux caractéristiques sont indéfinies, la recherche se déroule séquentiellement.

+++

La commande SELECTION permet de prendre en compte les évaluations qui ont pu être faites grâce aux méthodes d'évaluation du pouvoir discriminant définies en 2.6.

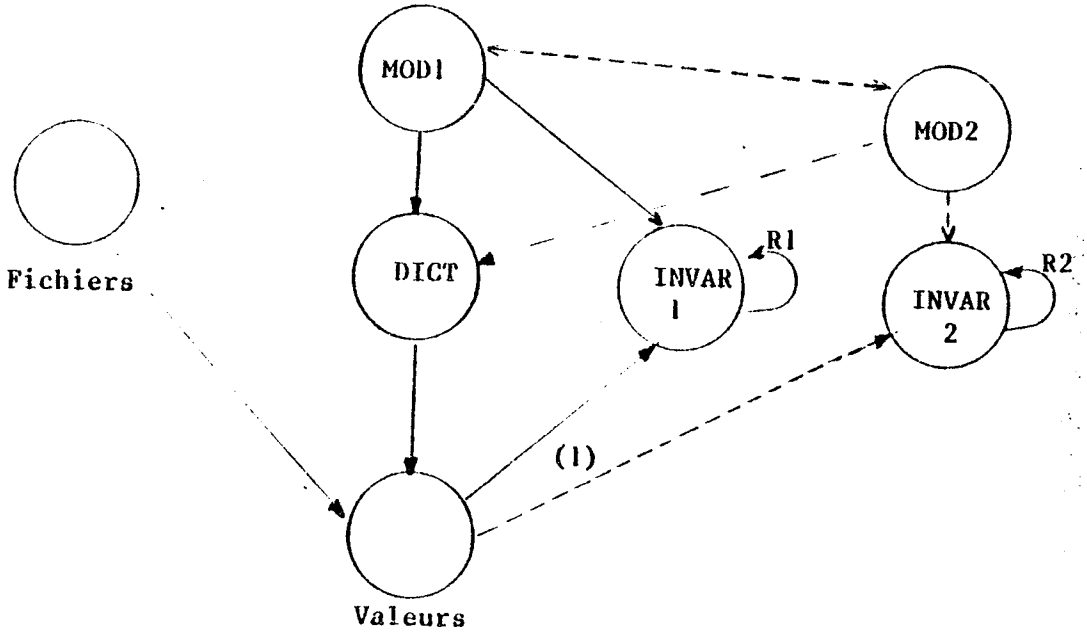
3.3.5.b) Les modèles de variation :

Les modèles définis au chapitre 2.4.2. sont séduisants par leur généralité, mais posent par là même des problèmes d'implémentation importants en raison de leur taille et surtout de leur grande diversité. Notre idée est en fait d'extrapoler cette notion vers une modélisation plus large des variations, qu'elles soient accidentelles, comme cela était notre propos au départ,

ou naturelles. On pourrait ainsi incorporer à l'application des modèles de mortalité, de nuptialité etc... qui seraient pris en compte au moment de l'évaluation des arcs. Dans l'état actuel des choses, nos réalisations sont plus modestes ; nous nous sommes limités à des modèles hiérarchisés, essentiellement pour pouvoir traiter les variations nominatives en exploitant les résultats de PIAFPHO (cf. 3.6.).

On se souvient que toutes les valeurs des caractéristiques sont stockées dans des dictionnaires ; un modèle est défini comme une hiérarchie de partitions sur l'ensemble de ses valeurs. Chaque élément de chaque niveau de partition est représenté par un invariant (dans le cas de PIAFPHO les chaînes de transduction phonétique correspondant aux trois niveaux produits).

On peut représenter ainsi le système d'information qui est implémenté :



Les arcs en pointillés représentent les liens supplémentaires qui sont établis lorsque le modèle indépendant MOD 2 est également défini sur DICT ; R1 et R2 représentent les hiérarchies définies respectivement sur les

ensembles INVAR 1 et INVAR 2 propres à chacun des modèles. Un fait lié à cette implémentation, est que l'utilisation des modèles doit-être prévue lors de la définition de la structure des fichiers acteur et événement ; les comparaisons doivent pouvoir s'effectuer directement entre les valeurs d'invariants substituées aux valeurs originelles, et il faut donc prévoir des caractéristiques réservées à cet usage dans les fichiers concernés (cf. exemples dans le chapitre précédent lors de la définition de relations de couplage impliquant l'utilisation de modèles). Cette liaison supplémentaire entre les fichiers et les invariants limite les possibilités de modification interactive des modèles, car il faut chaque fois repercuter les conséquences sur les fichiers de base concernés.

Nous ne donnons pas ici les commandes correspondant à la gestion des modèles ; elles sont nombreuses tant il est vrai que ce système constitue à lui seul une base de données dont l'intégrité est délicate à maintenir.

3.3.5. c) Les commandes de couplage :

L'environnement de travail "couplage" réunit l'ensemble des commandes correspondant à cette fonction ; l'algorithme de construction des classes de cohérence est celui défini en 2.5.2.b et dont le détail est donné en 3.3.7. Les commandes disponibles sont les suivantes :

D-CLASSE code-acteur DE type-acteur ?

Elle construit la classe de cohérence relative à l'acteur de type indiqué, désigné par son code. La classe est enregistrée dans la base et est disponible pour toute consultation ou traitement ultérieur (Jumelage).

Cette commande affiche le nombre d'acteurs composant la classe obtenue, et son numéro d'ordre.

I-CLASSE num-classe DE type-acteur MODE mode-édition SEUIL reel ?

Elle édite le contenu de la classe de cohérence dont le numéro d'ordre est donné en paramètre. Deux modes d'édition sont possibles selon la valeur attribuée au paramètre "mode-édition" :

- Mode "L" :

  Le contenu de la classe est édité sous forme de liste divisée verticalement en autant de colonnes que la classe comporte de couvertures. Sous chaque code d'acteur appartenant à une couverture donnée, figurent la liste des codes des enregistrements qui lui sont cohérents dans les autres couvertures, ainsi que le poids de l'arc correspondant (cf. exemple). Ce mode d'édition est à choisir pour les classes de taille importante (plus de 30 éléments).

- Mode "G" :

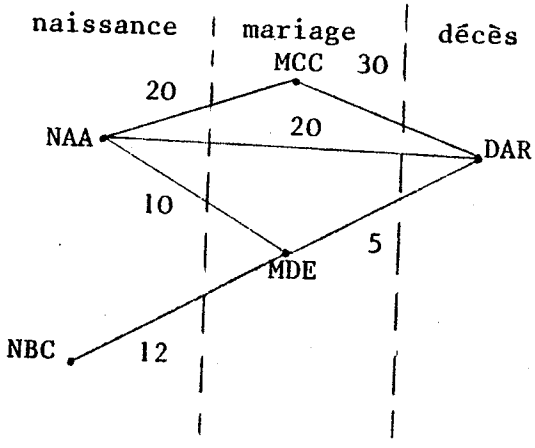
  Le contenu de la classe est édité sous forme de graphe, le mode de représentation choisi étant une matrice triangulaire. Les couvertures de la classe sont visualisées, ainsi que le code des acteurs composants, l'intersection d'une ligne et d'une colonne comporte le poids de l'arc correspondant à ces deux acteurs.

Le paramètre "seuil" permet de n'afficher que les poids supérieurs ou égaux à la valeur indiquée ; les autres sont remplacés à l'édition, par le symbole '<' . Cette option permet de faire ressortir, quel que soit le mode d'édition les arcs de poids élevé.

Exemple : Considérons le couplage de trois fichiers relatifs aux naissances, mariages, décès d'une population, et un enregistrement de code "NAA" relatif à un acte de naissance. La commande :

D-CLASSE NAA DE PERSONNE ?

construit la classe de cohérence de l'acteur indiqué dans les trois fichiers d'événements relatifs au type d'acteur "personne". Soit le graphe ci-dessous de la classe obtenue, où les acteurs de type "personne" sont représentés par leur code :



La commande de couplage produit le message :

<< CLASSE 1 - NOMBRE D'ENREGISTREMENTS : 5 >>

La visualisation de cette classe peut-être effectuée selon les deux modes d'édition suivants :

I-CLASSE 1 DE PERSONNE MODE L SEUIL 15 ?

qui produit l'édition en mode "liste" du graphe correspondant à la classe :

CLASSE :NAA

NAISSANCE	MARIAGE	DECES
NAA	MCC	DAR
MCC 20	NAA 20	MCC 30
MDE <	DAR 30	MDE <
DAR 20	MDE	NAA 20
NBC	NAA <	
MDE <	NBC <	
	DAR <	

I-CLASSE 1 DE PERSONNE MODE G SEUIL 15 ?

qui produit l'édition en mode "graphe" de la classe :

	DECES	MARIAGE	
	DAR	MCC	MDE
NAISSANCE			
NAA	20	20	<
NBC			<
MARIAGE			
MCC	30		
MDE	<		

+++

La commande :

S-CLASSE num-classe DE type-acteur ?

permet de supprimer la classe dont le numéro d'ordre est donné en paramètre.

Les mêmes fonctions peuvent être exécutées sur l'ensemble des fichiers :

D-COUPLAGE type-acteur ?

provoque la construction de l'ensemble des classes de cohérence pour le type d'acteur indiqué, et conformément au modèle défini pour cette catégorie d'informations. Pour chaque classe construite, il y a affichage de son numéro d'ordre et du nombre de ses composants ; en fin de traitement il y a édition :

- du nombre total de classes,
- du nombre total de classes unitaires (réduites à un seul acteur)

La commande :

I-COUPLAGE type-acteur MODE mode-édition SEUIL seuil ?

provoque l'édition, selon le mode choisi, de l'ensemble des classes de cohérence ; si le mode choisi est "G" (graphe), et que la taille d'une classe est trop importante pour ce mode d'édition, elle est automatiquement éditée sous forme de liste.

La commande :

S-COUPLAGE type-acteur PW mot-de passe ?

permet de supprimer l'ensemble des classes de cohérence produites pour le type d'acteur indiqué ; cette commande implique la fourniture d'un mot de passe. Elle est principalement utilisée en préambule du couplage définitif des fichiers (une fois le modèle jugé satisfaisant), pour s'assurer qu'aucune des classes construites lors de la phase d'essais ne subsiste dans la base.

3.3.6. Ensemble des fonctions liées au jumelage des informations :

3.3.6. a) Définition - modification des paramètres de jumelage :

Nous avons un (cf. 3.3.3.a) que ces paramètres sont au nombre de deux :

- Le facteur de qualité défini pour le type d'acteur
- La matrice représentant l'ensemble des règles de compatibilité

On peut affecter la valeur de ces paramètres par les commandes suivantes de l'environnement "paramètre" :

D-QUAL type-acteur A réel ? * affecte le facteur de qualité relatif au type d'acteur indiqué, à la valeur indiquée par 'reel'*

I-QUAL type-acteur ? * affiche le facteur de qualité de 'type-acteur' *

La commande :

D-MATCOMP type-acteur ?

engage un dialogue avec l'utilisateur, lui demandant de fixer les valeurs de MIN et de MAX pour toutes les relations de couplage définies entre les fichiers correspondant à 'type-acteur'.

La commande :

M-MATCOMP type-acteur ?

effectue un traitement analogue, mais c'est l'utilisateur qui spécifie l'élément de la matrice à modifier en indiquant le nom des deux fichiers concernés.

3.3.6. b) Commandes de jumelage :

Les commandes de l'environnement "jumelage" mettent en oeuvre les algorithmes de sélection et de jumelage définis en 3.3.8. et 3.3.9.; on peut considérer deux classes de commandes :

a) Les commandes d'évaluation :

qui se limitent à la sélection des solutions de jumelage à partir d'une ou plusieurs classes de cohérence. Ces commandes permettent surtout de valider la partie 'mesure' du modèle de l'application, en observant :

- l'incidence des types de mesure affectées aux différentes relations élémentaires.
- l'incidence du facteur de qualité,

sur le processus de construction des solutions.

Les commandes disponibles à ce niveau sont :

D-SOL num-classe DE type-acteur ?

Cette commande construit les solutions de jumelage de poids maximal contenues dans une classe de cohérence d'un type d'acteur donné. La classe est désignée par son numéro d'ordre "num-classe". Le résultat de l'opération est édité sous la forme suivante :

- poids de la solution
- poids moyen des arcs dans l'ensemble des possibilités
- Pour chaque solution : édition des noeuds appartenant à chaque composante connexe (le code acteur seul est affiché).

La même fonction peut-être appliquée à un sous-ensemble de classes pour des essais plus systématiques ; nous avons choisi comme critère d'extraction de tels sous-ensembles la taille des classes :

EXTR-CLASSE type-acteur TAILLE entier ?

Cette fonction extrait dans un fichier de travail l'ensemble des classes de cohérence d'un type d'acteur donné, ayant un nombre de sommets supérieur ou égal à entier (elle affiche le nombre de classes correspondant à ce critère).

D-SOL ENSEMBLE DE type-acteur ?

effectue le traitement de la commande D-SOL précédente sur un ensemble de classes produit par la commande EXTR-CLASSE ; elle affiche de plus :

- le poids moyen des solutions
- le poids moyens des arcs dans tous les ensembles de possibilités construits.

Les commandes

S-SOL code-classe DE type-acteur ?

et

S-SOL ENSEMBLE DE type-acteur ?

suppriment de la base les solutions construites par les commandes D-SOL.

b) Les commandes impliquant le jumelage :

Le jumelage des informations est une opération irréversible impliquant des modifications de structure définitives (cf. 3.3.9.) ; ces commandes sont donc soumises à certaines règles de sécurité pour prévenir les erreurs de manipulation qui ne pourraient être redressées. Nous avons pour cela choisi des codes de commandes prêtant peu à confusion, et soumis l'exécution de ces fonctions à la fourniture d'un mot de passe

PW mot-passe ?

Les commandes sont les suivantes :

JUMELAGE num-classe DE type-acteur ?

Cette commande provoque la sélection des solutions dans la classe définie par num-classe ; si le nombre de solutions de poids maximal obtenues est supérieur à 1 (ambiguité), il y a édition de ces solutions et abandon du traitement. S'il n'existe qu'une solution de poids maximal, le jumelage des acteurs composants est effectué ; la classe de cohérence est détruite.

JUMELAGE ENSEMBLE DE type-acteur ?

Le même traitement est effectué sur un ensemble de classes de cohérence.

JUMELAGE-COMPLET type-acteur ?

Le processus est effectué sur toutes les classes de cohérence du type d'acteur indiqué.

Enfin, nous avons prévu, pour résoudre les cas d'exception, une commande autorisant le jumelage forcé d'une classe de cohérence ; cette commande n'utilise que l'algorithme de jumelage, l'utilisateur indiquant lui-même les composantes de la solution qu'il a choisi manuellement :

JUMELAGE-FORCE code classe DE type-acteur ?

Cette commande engage un dialogue avec l'utilisateur, et lui demande la définition des composantes de la solution, sous la forme de listes de codes-acteur. En fin d'opération, la classe de cohérence est supprimée.

3.3.7. Présentation de l'algorithme de couplage

Les principes relatifs à la définition de cet algorithme ont été développ au chapitre 2.5. ; il est utilisé par toutes les commandes de couplage de l'environnement de travail "couplage" (Cf. 3.3.5.c).

3.3.7.a) Présentation des structures logiques d'informations

Nous définissons ici les principales structures d'informations utilisées par l'algorithme de couplage.

a) Représentation des classes de cohérence

Nous avons ici à représenter des graphes m-partis ; les conventions ci-dessous seront largement reprises pour définir d'autres objets de même nature comme les possibilités de jumelage ou les solutions de jumelage (Cf. 3.3.8.).

Les classes de cohérence sont des graphes non orientés ; on peut donc réduire leur représentation à une matrice triangulaire. Le mode de défini choisi est le suivant :

(m) type classe = [{tarc-1}]^m
avec
type tarc-1 = (OR : chaîne ; SET-EXTR:{(EXTR:chaîne;PDS:réel)})

Considérons le couplage de N fichiers relatifs à un type d'acteur déterm toute classe de cohérence est composée d'un maximum de m = C_2^N sous-graph bi-partis de couvertures X^i et X^j, i, j $\in$ [1, ..., N], i $\neq$ j. La suite de m éléments de la définition de 'classe' représente ces m sous-graphes Si on choisit la matrice triangulaire supérieure pour représenter la classe, on a la correspondance suivante entre i, j et k :

$$k = (j-2)(j-1)/2 + i, \text{ avec } 1 \leq i < j \leq N$$

Soit une variable G de type 'classe' ; G[k] représente le sous-graphe bi-parti de la classe correspondante à ses couvertures X^i et X^j.

Tout élément A de G[k] est un ensemble d'arcs ayant même origine A.OR (code de l'acteur origine) de X^i, leurs extrémités dans X^j étant enregis dans A.SET-EXTR. Tout élément B de A.SET-EXTR est un acteur de X^j de cod B.EXTR, extrémité d'un arc (x_1, x_2) tel que :

x_1 = A.OR

x_2 = B.EXTR

de poids de cet arc est donné par B.PDS.

b) Définition complète des types t-act et t-even

Nous avions donné en 3.3.3.a des définitions partielles de ces types ; ils sont complétés de la façon suivante pour prendre en compte les informations nécessaires à l'algorithme de couplage :

(n) type t-act = (TYPE:chaîne;V-TEVEN:[t-even]^N;Q:réel;MATCOMP:t-mat;
MAX:f-acteur;NONMAX:f-acteur;SET-CL:{classe})

avec :

type f-acteur = ou ({act-personne},{act-couple})

et pour le type t-even :

type t-even = (TYPE:chaîne;PERM:f-acteur;TRAV:f-acteur telque TRAV⊆PERM)

Dans 't-act', SET-CL représente l'ensemble des classes de cohérence construites pour le type d'acteur donné ; MAX et NONMAX sont des ensembles de travail utilisés durant l'algorithme de couplage.

Dans 't-even', PERM représente l'ensemble des données permanentes relatives au type d'acteur et au type d'évènement considéré, permettant une restauration des données en cas de suppression de toute classe de cohérence. TRAV est un sous-ensemble du fichier permanent définissant les enregistrements restant à coupler ; au chargement de la base, TRAV = PERM.

3.3.7.b) Restrictions apportées à l'algorithme général de couplage

L'algorithme présenté en 2.7.5.2 a été implémenté avec les restrictions suivantes :

Les tests de productivité sont limités aux relations entre caractéristiques logiquement invariantes ; cette restriction n'est pas gênante dans la mesure où ces informations sont suffisamment nombreuses pour conserver, malgré les cas d'omissions, un bon pouvoir discriminant. Dans le cas contraire, on risque de ne pas déceler tous les éléments maximaux, et les performances peuvent se dégrader.

- Le test de productivité est limité à la valeur indéterminée, élément universel de toute relation (Cf. 2.4.3.a).

- La troisième limitation concerne seulement l'algorithme présenté ci-après on considère que toutes les relations de couplage, pour un type d'acteur donné, ont en commun la définition des relations R'^{ij} (relations entre caractéristiques invariantes). Bien que paraissant contraignante, cette restriction n'est en fait pas gênante dans la mesure où les couplages s'effectuent entre acteurs du même type (donc possèdant naturellement la même description, donc étant liés par les mêmes relations) ; elle intervient surtout lorsqu'on envisage des couplages entre acteurs de types différents, ce qui est relativement rare.

Selon les remarques faites en 2.5.2.c, ces options permettent les simplifications suivantes :

- Unicité de la fonction MAX (Cf. 2.5.1.c) pour le type d'acteur considéré
- Possibilité de représenter les couvertures comme un sous-ensemble unique de $E = E^1 + \ldots + E^N$. Dans la définition du type 't-act' (Cf. (n) ci-dessus), les ensembles MAX et NONMAX sont respectivement utilisés pour effectuer une partition de l'ensemble des sommets de toute classe de cohérence en éléments maximaux et non maximaux ; ils sont utilisés pa la procédure ARC (Cf. 2.5.1.f) qui complète la construction de la classe

3.3.7.c) L'algorithme de couplage

Les assertions suivantes sont vérifiées avant et après l'exécution de la procédure COUPLAGE :

<TYPACT est défini ; ∃ I ∈ [1,...,N] | E ∈ TYPACT.V-TEVEN [I].TRAV ;
CL = nult ; TYPACT.MAX = TYPACT.NONMAX = ϕ>

COUPLAGE (E, TYPACT, CL)

<CL est la classe de cohérence relative à l'enregistrement de départ E de TYPACT ; TYPACT.SET-CL contient l'ensemble des classes de cohérence construites pour ce type d'acteur ; TYPACT.MAX = TYPACT.NONMAX = ϕ>

Remarque : Les procédures notées (*) ne sont pas détaillées ici.

procédure COUPLAGE (E:acteur ; TYPACT:t-act ; CL:classe)

var N:entier

début co construction de la classe de cohérence de l'enregistrement de départ E co

(1) <$\exists I > 0 \mid E \in$ *TYPACT.V-TEVEN[I].TRAV ; CL=nult ; TYPACT.MAX=TYPACT.NONMAX=*ϕ>

COUVERTURE (E, TYPACT , CL)

(2) <*TYPACT.MAX contient les éléments maximaux de la couverture classe ; TYPACT.NONMAX contient les éléments non maximaux de la couverture ; CL contient tous les arcs de la classe comportant au moins un élément maximal avec leur poids*>

ARC (TYPACT, CL) (*)

(3) <*CL est éventuellement complétée par des arcs joignant des éléments non maximaux de sa couverture, contenus dans TYPACT.NONMAX ; TYPACT.NONMAX = TYPACT.MAX =* ϕ *; CL* $\neq$ *nult*>

aj (CL $\in$ TYPACT.SET-CL)

(4) <*TYPACT.SET-CL est l'ensemble des classes de cohérence construites lors du couplage des acteurs correspondant à TYPACT*>

fin

```
procédure COUVERTURE (E:acteur ; TYPACT:t-act ; CL:classe)
var NOUVMAX, COUV:f-acteur ; TERMINE:booléen ; F:acteur
début co construction de la couverture de la classe;recherche des éléments maximau
    (1) <CL=nult ; NOUVMAX=COUV=ϕ ; TYPACT.NONMAX=TYPACT.MAX=ϕ ; (1) de COUPLAGE>
        aj (E ∈ COUV) TERMINE := faux
    (2) <COUV≠ϕ, contient l'enregistrement de départ>
    tantque non TERMINE
      faire (3) <COUV≠ϕ contient les éléments de la couverture du pas précédent ;
                 NOUVMAX=ϕ ; TYPACT.NONMAX contient les éléments non maximaux ;
                 TYPACT.MAX contient les éléments maximaux>
             MAX (COUV, TYPACT, NOUVMAX) (*) co cf. 2.5.1.e co
        (4) <NOUVMAX contient les éléments maximaux de COUV (nouveaux) ; COUV≠ϕ
             TYPACT.NONMAX contient les éléments non maximaux>
             TERMINE := NOUVMAX = ϕ
        (5) <traitement terminé si plus d'éléments maximaux trouvés>
        pourtout F ∈ NOUVMAX
          faire (6) <NOUVMAX≠ϕ ; COUV=ϕ>
                 COUPES (F, TYPACT, COUV, CL)
            (7) <COUV contient les coupes de F ; F ne peut plus être obten
                 dans les coupes ultérieures; les arcs trouvés pour F ont
                 enregistrés dans CL avec leur poids>
                 tr (F ∈ NOUVMAX dans TYPACT.MAX)
            (8) <F est supprimé de NOUVMAX et placé dans TYPACT.MAX>
          finfaire
        (9) <COUV contient l'ensemble des coupes des éléments de NOUVMAX ; tous
             les arcs trouvés pour tous les éléments de NOUVMAX sont enregistré
             dans CL avec leur poids ; NOUVMAX=ϕ ; TYPACT.MAX contient les élém
             maximaux de la couverture ; TYPACT.NONMAX contient les éléments no
             maximaux>
    finfaire
fin
```

```
procédure COUPES (F:acteur ; TYPACT:t-act ; COUV:f-acteur ; CL:classe)
var EVENP, EVENS:t-even ; I:entier
début  co construction des coupes de F co
    (1) <cf (6) de COUVERTURE>
        I := FICH (F) (*)
    (2) <F ∈ TYPACT.V-TEVEN [I].TRAV>
        EVENP := TYPACT.V-TEVEN [I].TRAV
    pour tout EVENS ∈ TYPACT.V-TEVEN | EVENS ≠ EVENP
      faire
        (3) <EVENS ≠ EVENP ; F ∈ EVENP.TRAV>
            COUPE (F, EVENP, EVENS, TYPACT, COUV, CL)
        (4) <COUV a été augmenté de la coupe de F dans EVENS.TRAV ; les arcs
            correspondants ont été enregistrés dans CL avec leur poids>
      finfaire
    (5) <COUV a été augmenté de la coupe de F dans les fichiers différents de EVENP ;
        tous les arcs correspondants ont été ajoutés à CL avec leur poids>
        sup (F ∈ EVENP.TRAV)
    (6) <F ne peut plus être obtenu dans aucune coupe >
fin
```

```
procédure COUPE (F:acteur ; EVENP ; EVENS:t-even ; TYPACT:t-act ; COUV:f-acteur ;
                 CL:classe)
var T2:f-acteur telque T2 ⊆ EVENS.TRAV ; Z1, Z2:chaîne
début co coupe de F dans EVENS.TRAV. Phase de sélection rapide
     co Le code de cette procédure est engendré par l'instruction SELECTION cf
  (1) <cf (3) de COUPES ; T2 = ϕ>
      Z1 := F.caract1  Z2 := F.caract2
      co caract1 est le critère d'accès rapide primaire ; caract2 est le critère
         secondaire co
      Si Z1 ≠ u
        alors co valeur du critère primaire définie pour F co
                 T2 := {E ∈ EVENS.TRAV|E.caract1 = Z1 ou E.caract1 = u}
              (2) <T2 contient les acteurs de EVENS.TRAV vérifiant le critère
                   primaire, ou ayant cette valeur indéfinie>
        sinon co valeur du critère primaire indéfinie co
                 Si Z2 ≠ u
                   alors co valeur du critère secondaire définie pour F co
                           T2 := {E ∈ EVENS.TRAV|E.caract2 = Z2 ou E.caract2 = u
                        (3) <T2 contient les acteurs de EVENS.TRAV vérifiant le
                            critère secondaire, ou ayant cette valeur indéfinie>
                   sinon co valeur des deux critères indéfinie : parcours
                          séquentiel co
                         T2 := EVENS.TRAV
                   finsi
        finsi
  (4) <T2 contient le référentiel de recherche sélectionné par les critères ;
       T2 ⊆ EVENS.TRAV>
      SUITE-COUPE (F, EVENP, EVENS, T2, TYPACT, COUV, CL)
  (5) <COUV a été augmenté de la coupe de F dans EVENS.TRAV après sélection rapid
       dans T2 ; les arcs correspondants ont été enregistrés dans CL avec leur po
fin
```

```
procédure SUITE-COUPE (F:acteur ; EVENP, EVENS:t-even ; T2:f-acteur ; TYPACT:t-act ;
                       COUV:f-acteur ; CL:classe)
var E:acteur ; PDS:réel
début co suite de la construction de la coupe à partir du référentiel T2 co
    (1) <cf (4) de COUPE>
    pour tout E ∈ T2
        faire
            (2) <T2 ≠ ϕ ; E ∈ EVENS.TRAV ; P ∈ EVENP.TRAV>
                Si RELACT-PROD (F, E, TYPACT)          (*)
                    alors
                        (3) <les caractéristiques invariantes de E et F sont cohérentes
                            si RELEV (F, E, EVENP, EVENS, TYPACT) (*)
                                alors
                                    (4) <Les caractéristiques évènementielles sont cohérent
                                        ainsi que les caractéristiques invariantes ; E
                                        et F sont cohérents et E appartient à la couvertur
                                        de F>
                                        PDS := MESURE (F, E) (*)
                                    (5) <PDS a pour valeur le poids de l'arc (F, E)>
                                        ENRCLASSE (F, E, EVENP, EVENS, PDS, CL) (*)
                                    (6) <l'arc (F, E) et son poids sont enregistré dans CL>
                                        aj (E ∈ COUV)
                                    (7) <E appartient à la couverture de la classe>
                                finsi
                    finsi
        finfaire
fin
```

Les fonctions RELACT-PROD, RELEV, MESURE sont particulières à chaque application ; comme la procédure COUPE, elles sont produites à partir d'un langage de définition particulier (Cf. 3.3.5.a).

3.3.8. Présentation de l'algorithme de sélection des solutions de jumelage

Les principes relatifs à la définition de cet algorithme ont été présenté dans le chapitre 2.7.5.c ; il est utilisé par toutes les commandes de l'environnement "jumelage" qui impliquent la construction de solutions de jumelage à partir d'une classe de cohérence (Cf. 3.3.6.b).

3.3.8.a) Définition des structures logiques d'information

Nous donnons ici les modalités de représentation des diverses information évoquées au chapitre 2.7.5 utilisées dans l'algorithme de sélection des solutions de jumelage.

a) Représentation des possibilités de jumelage

Rappelons tout d'abord que la matrice de compatibilité ainsi que le facte de qualité sont définis relativement à chaque type d'acteurs à traiter (Cf. 3.3.3.a) (e) et (f)); les possibilités de jumelage extraites d'une classe de cohérence sont rangées dans une suite, triée par ordre décroi sant des poids.

(q) var V.POSS : [possibilité]*
avec
type possibilité = (PDS:réel ; TRACE : {entier}; SET-SOM:{chaîne};
GRAPHE : tgraphe)

-TRACE contient la définition de la trace de la possibilité ; l'ensemble d'entiers correspond aux numéros d'ordre des fich composant la trace, dans V-TEVEN (suite des fichiers relatif au type d'acteur traité (Cf. 3?3.3.a (b)).

-SET-SOM contient l'ensemble des sommets constituant la possibilité, représentés uniquement par leur code.

-GRAPHE contient la représentation des arcs constituant la possibilité (Cf. (t) ci-dessous).

b) Représentation des solutions de jumelage

Les solutions de jumelage sont représentées de la façon suivante :

(r) type solution = (PDS:réel ; SET-COMP:{composante})
et
var SET-SOL:{solution}

où - PDS est le poids de la solution
- SET-COMP contient l'ensemble des composantes connexes de la solution.

Le résultat du programme de sélection des solutions est placé dans SET-SOL.

Une composante connexe est représentée comme suit :

(s) type composante = (TRACE:{entier}; SET-SOM:{chaîne}; GRAPHE:tgraphe ; COMPLET:booléen)

où : - TRACE, SETSOM, GRAPHE sont définies comme dans le type 'possibilité' (Cf. (q) ci-dessus)
- COMPLET est un booléen ayant la valeur vrai si et seulement si la composante est un graphe m-parti complet (Cf. 2.7.3.a)

c) Représentation des graphes

Les possibilités de jumelage, les composantes connexes des solutions sont représentées de manière analogue aux classes de cohérence (Cf. 3.3.7.a (m)) :

(t) type tgraphe = [(CARD-SEC:entier ; ARC:{tarc})]^{m}
avec
type tarc = (OR:sommet-1 ; EXTR:{sommet-2})
et

(u) type sommet-1 = (CODE:chaîne ; DEGRE:entier)
sommet-2 = (CODE:chaîne ; DEGRE:entier ; PDS:réel)

Considérons une variable G de type tgraphe :

- rappel : comme pour les classes de cohérence, on a :

- $m = C_2^N$

- G[k] contient le sous-graphe bi-parti de G correspondant aux couvertures X^i et X^j avec :

$$1 \leq i < j \leq N \quad , \text{ et } \quad k = (j-1)(j-2)/2+i$$

- G[k].ARC contient l'ensemble des arcs correspondant au sous-graphe G[k]
- tout élément A de G[k].ARC représente des arcs de G[k] ayant pour origine commune A.OR(élément de X^i), leurs extrémités (dans X^j) étant enregistrées dans A.EXTR. On a donc :

$$A.OR.DEGRE = \underline{card}(A.EXTR) = degré_{ij}(A.OR)$$

et pour tout élément S de A.EXTR :

$$degré_{ij}(S) = S.DEGRE \leq \underline{card}(G[k].ARC)$$

- La variable G[k].CARD-SEC contient le cardinal de X^j ; elle est utilisé pour vérifier si le graphe G est complet :

  - G est complet si et seulement si G[k] est complet pour tout $k \in [1,m]$
  - G[k] est complet si et seulement si :

$$\forall A \in G[k].ARC,\ A.OR.DEGRE = G[k].CARD.SEC$$

3.3.8.b) Présentation de l'algorithme

L'algorithme est représenté par la procédure SELECTION définie ci-après ; les assertions avant et après l'exécution de cette procédure sont les suivantes :

(1) < CL est une classe de cohérence non unitaire, construite à partir du couplage de N fichiers relatifs à un type d'acteur déterminé par TYPACT ; TYPACT.Q et TYPACT.MATCOMP ont une valeur définie ; SET-SOL est vide >

SELECTION (CL,TYPACT,SET-SOL)

(2) < SET-SOL est l'ensemble des solutions de poids maximal extraites de CL à partir des paramètres TYPACT.Q et TYPACT. MATCOMP (Cf. 2 SET-SOL peut être vide >

Dans la présentation qui suit, toutes les prdcédures élémentaires ne sont pas détaillées ; elles sont indiquées par (*).

```
procédure SELECTION (CL:classe ; TYPACT:t-act ; SET-SOL = {solution})
var V-POSS : (PDSTOTAL:réel ; SP:[possibilité]*)
    co V-POSS représente l'espace des possibilités ; PDSTOTAL est le poids de ces
       possibilités co
début
    co Construction des possibilités à partir de CL(détail non donné ici) co
    (1) < V-POSS.PDSTOTAL = u ; V-POSS.SP = nult >
        CONST-POSS (CL, TYPACT, V-POSS) (*)
    (2) < V-POSS.PDSTOTAL ≥ 0 ; V-POSS.SP est trié par ordre croissant des possi-
        bilités composantes ; V-POSS.PDSTOTAL = 0 <=> toutes les possibilités
        sont de poids nul (Cf remarque 3 en 2.7.5.c), ou il n'y a pas de possi-
        bilités de jumelage >
    co Construction-selection des solutions de jumelage co
    Si V-POSS.PDSTOTAL > 0
      alors
        (3) < V-POSS.SP ≠ nult ; V-POSS.PDSTOTAL > 0 ; SET-SOL = ϕ >
            SELECT-SOL (V-POSS.PDSTOTAL, V-POSS.SP, TYPACT, SET-SOL)
        (4) < SET-SOL ≠ ϕ contient les solutions de poids maximal non nul >
      sinon
        (5) < V-POSS.PDSTOTAL = 0 ; V-POSS.SP peut être égal à nult >
            GENSOL (V-POSS.PDSTOTAL, V-POSS.SP, SET-SOL) (*)
        (6) < SET-SOL ne contient qu'une solution dont les composantes sont
            réduites à un seul sommet, ou SET-SOL est vide car
            V-POSS.SP = nult >
      finsi
fin
```

```
procédure SELECT-SOL (PDSTOTAL:réel; SP:[possibilité]* ; TYPACT:tact ;
                      SET-SOL:{solution})
var P:possibilité; PDSMAX, RESTE:réel; S1, S2:{solution} telque S1,S2 ⊂SET-SOL;
    S: solution
co SET-SOL = S1∪S2; S1 contient les solutions complètes, S2 contient les
   solutions incomplètes. Voir remarque 4 en 2.7.5.c co
debut
    (1) < Cf. (3) de SELECTION; S1 = S2 = ϕ ; PDSMAX = RESTE = u >
          PDSMAX := 0     RESTE := PDSTOTAL
    (2) < SP ≠ nult ; RESTE >0 >
  pour tout P ∈SP
    faire co on considère les possibilités par ordre de poids décroissant c
      (3) < au pas n, SET-SOL = S1 ∪S2 représente S_{n-1}, P représente
            P_n cf. 2.7.5;c >.
          CONSTRUCT-SOL (P, TYPACT, RESTE, PDSMAX, SET-SOL)
      (4) < au pas n, SET-SOL représente S_n ; PDSMAX est le poids maximal
            des éléments de S1; S1 et S2 sont disjoints. >
      si RESTE >PDSMAX
        alors
          (5)  < il peut exister des solutions maximales dérivées de P >
               INIT-SOL (P, S1, SET-SOL)   (*)
          (6)  < une solution complète, constituée par P, est incorporée
                 à SET-SOL >
        finsi
      RESTE := RESTE - P.PDS
    finfaire
    (7) < S1 ne contient que des solutions complètes; il peut subsister dans
          S1 des solutions de poids inférieur à PDSMAX; les éléments de S2,
          non complets, ne sont pas des solutions de jumelage >
    pour tout S∈SET-SOL | S.PDS < PDSMAX ou S ∈S2
      faire
          supp ( S ∈SET-SOL)
      finfaire
      (8) < SET-SOL ne contient que des solutions de jumelage, de poids
            maximal >
fin
```

```
procédure CONSTRUCT-SOL(P:possibilité ; TYPACT:t-act ; RESTE,PDSMAX:réel ;
                        SET-SOL:{solution})

var   NEWSOL, S:solution ; COMPATIBLE, COMPLET:booléen

début

    (1) < cf. (3) dans SELECT-SOL ; NEWSOL, S = nult >

  Pour tout  S ∈  SET-SOL

    faire  co comparaison de P avec toutes les solutions existantes co

      Si S.PDS + RESTE ≥ PDSMAX

        alors   NEWSOL := nult

            (2) < il peut exister des solutions maximales dérivées de S et
                  de P ; NEWSOL = nult >

                COMPAT-SOL (P, S, NEWSOL, TYPACT, COMPATIBLE, COMPLET)

            (3) < COMPATIBLE <=> S et P sont compatibles ; COMPLET <=> S ∪ P
                  est une solution de jumelage ; NEWSOL contient S ∪ P si
                  S et P sont compatibles, elle est égale à nult sinon >

            Si COMPATIBLE

              alors

                  (4) < P et S sont compatibles, NEWSOL contient P ∪ S >

                      DERIV-SOL(NEWSOL,COMPLET,PDSMAX,SET-SOL)

                  (5) < NEWSOL est ajoutée à SET-SOL ; elle ne peut être
                        comparée de nouveau à P ; NEWSOL est ajoutée à S1 ou
                        S2 selon que COMPLET est vrai ou faux ; PDSMAX contient
                        le poids maximal des éléments de S1 >

              finsi

        sinon

            (6) < On ne peut deriver une solution maximale de S et P >
                supp (S ∈ SET-SOL)

            (7) < S est éliminée de SET-SOL, et donc de S1 ou S2 >

        finsi

    finfaire

    (8) < P a été comparée à tous les éléments de SET-SOL ; PDSMAX est le poids maxi-
          mal des éléments de S1 qui sont des solutions de jumelage ; S2 contient des
          graphes non complets >

fin
```

```
procédure COMPAT-SOL(P:possibilité ; S,NEWSOL:solution ; TYPACT:t-act ; COMPATIBLE,
                     COMPLET:booléen)
var COMP,C:composante ; DISJOINT:booléen
début co Comparaison de P et S ; construction de NEWSOL = P ∪ S co
    (1) < cf.(2) dans CONSTRUCT-SOL ; COMP = nult >
        INIT-COMP(COMP,P) (*) COMPLET := COMPATIBLE := vrai
    (2) < COMP représente le même graphe que P >
    Pour tout C ∈ S.SET-COMP co comparaison de COMP avec les composantes de S co
    faire
        Si COMP.TRACE = C.TRACE
        alors   (3) < COMP et C ont même trace ; ils vérifient la 2° règle
                      de compatibilité (cf.2.7.3.b)>
                COMPAT-C(COMP,C,COMPATIBLE,DISJOINT,TYPACT)
                (4) < COMPATIBLE <=> COMP et C sont compatibles, et COMP
                      contient COMP ∪ C ; DISJOINT <=> COMP et C sont dis-
                      jointes, donc compatibles>
        sinon
                (5) < COMP et C ont des traces différentes >
                      COMPATIBLE := DISJOINT := COMP.SET-SOM ∩ C.SET-SOM = ϕ
                (6) < COMPATIBLE <=> DISJOINT <=> COMP et C sont disjointes >
        finsi
        Si non COMPATIBLE
        alors   (7) < il existe une composante de S incompatible avec
                      COMP:abandon >
                sortie
        sinon
                Si DISJOINT
                alors (8) < C et COMP sont disjointes ; C doit être
                            incorporée à NEWSOL >
                      aj (C ∈ NEWSOL.SET-COMP)
                      (9) < C devient une composante de NEWSOL >
                finsi
        finsi
    finfaire
    (10) < COMP contient l'union de toutes les composantes de S compatibles et
           connexes à P ; NEWSOL contient les composantes de S disjointes de COMP ;
           COMPATIBLE <=> toutes les composantes de S sont compatibles avec P >
           Si COMPATIBLE
           alors   NEWSOL.PDS := S.PDS + P.PDS
                   aj (COMP ∈ NEWSOL.SET-COMP)
                   (11) < NEWSOL est entièrement constituée >
                   COMPLET := TEST-COMPL (NEWSOL) (*)
                   (12) < COMPLET <=> toutes les composantes de NEWSOL sont complètes >
           finsi
fin
```

```
procédure COMPAT-C (COMP, C:composante ; COMPATIBLE,DISJOINT:booléen ; TYPACT:t-act)

début co évaluation de la compatibilité de deux composantes co
|
|     (1) < cf (3) de COMPAT-SOL : COMP et C ont même trace >
|           DISJOINT := COMP.SET-SOM ∩ C.SET-SOM = ϕ
|
|   Si non DISJOINT
|
|     alors
|     |
|     |     (2) < COMP et C ont même trace et sont connexes >
|     |           COMPAT-DEGRE (COMP,C,TYPACT.MATCOMP,COMPATIBLE) (*)
|     |
|     |     (3) < COMPATIBLE <=> COMP ∪ C constitue un graphe vérifiant les cont-
|     |           raintes de degré maximales et minimales fixées par MATCOMP ; si
|     |           ces contraintes sont satisfaites, COMP contient à présent l'union
|     |           de son ancienne valeur avec C >
|     |
|     |   Si COMPATIBLE
|     |     alors
|     |     |
|     |     |     (4) < COMP a été modifié >
|     |     |           COMP.COMPLET := COMPL (COMP.GRAPHE) (*)
|     |     |
|     |     |     (5) < COMP.COMPLET <=> COMP.GRAPHE est un graphe m-parti complet >
|     |     |
|     |     finsi
|     |
|     sinon
|     |
|     |     (6) < COMP et C ont même trace et sont disjointes >
|     |           COMPATIBLE := vrai
|     |
|     finsi
|
fin
```

```
procédure  DERIV-SOL (NEWSOL:solution ; COMPLET:booléen ; PDSMAX:réel ; SET-SOL
                      SET-SOL:{solution})

début
|
|       (1) < cf (4) de CONSTRUCT-SOL >
|
|             aj (NEWSOL ∈ SET-SOL)
|
|       (2) < NEWSOL est ajouté à SET-SOL >
|
|     Si COMPLET
|
|       alors
|       |
|       |       (3) < NEWSOL est une solution de jumelage >
|       |
|       |             aj (NEWSOL ∈ S1)
|       |
|       |       (4) < S1 ⊆ SET-SOL ne contient que des solutions de jumelage >
|       |
|       |     Si NEWSOL.PDS > PDSMAX
|       |       alors
|       |       |
|       |       |       PDSMAX := NEWSOL.PDS
|       |       |
|       |       finsi
|       |
|       sinon
|       |
|       |       (6) < NEWSOL n'est pas un graphe dont toutes les composantes connexes
|       |             sont complètes ; elle n'est pas une solution de jumelage ;
|       |             NEWSOL vérifie toutes les contraintes de compatibilité >
|       |
|       |             aj (NEWSOL ∈ S2)
|       |
|       |       (7) < S2 ⊆ SETSOL contient les solutions non complètes >
|       |
|       finsi
|
fin
```

3.3.9. Présentation de l'algorithme de jumelage

Le jumelage d'un ensemble d'acteurs de type donné est avant tout une restructuration des informations concernées ; par définition cette opération concerne une classe d'homologie, et peut donc être effectuée à partir de n'importe quel représentant de cette classe. Les entités concernées sont des acteurs ; selon le type considéré, elles sont rattachées à d'autres informations par des liens ayant une signification particulière (cf. les types act-personne et act-couple définis par (f) et (g) au chapitre 3.3.4.a). La restructuration de l'information concerne principalement le transfert de ces liens vers le représentant de la classe d'homologie ; une fois ce transfert effectué, tous les acteurs autres que le représentant sont supprimés. Nous montrons au chapitre 3.4 comment ces transformations sont utilisées pour reconstituer une population.

3.3.9.a) Les données jumelables dans MERCURE - restriction

La procédure de jumelage de MERCURE est actuellement figée aux types d'acteurs définis par (f) et (g) que nous rappelons ci-dessous :

(f) type act-personne = (CODE:chaîne ; D-PERSONNE:personne ;
UNION:{(CONJOINT:act-personne ; COUPLE-D:act-couple ; EVEN:mariage)};
BIOGRAPHIE:{évènement})

(g) type act-couple = (CODE:chaîne ; EPOUX, EPOUSE:act-personne ;
ENFANTS:{act-personne} ; BIOGRAPHIE:{évènement})

La restriction porte sur la seule prise en compte des liens directs de parenté, d'union, ou de nature biographique (rappelons que seuls les types 'personne', 'évènement' et 'mariage' ne sont pas standard).

3.3.9.b) L'algorithme de jumelage

Considérons la procédure élémentaire de jumelage de deux acteurs de même type :

< TYPACT est l'un des deux éléments de APPLICATION.V-TACT ; E et F sont deux acteurs de ce type :

$\exists J, \exists K \in [1,..N] \mid E \in$ TYPACT.V-TEVEN [J].PERM
$F \in$ TYPACT.V-TEVEN [K].PERM >

JUM.ELEM (E,F,TYPACT)

< F est supprimé de TYPACT.V-TEVEN [J].PERM si les conditions de jumelage correspondant à son type sont vérifiées (cf. conditions détaillées dans les procédures JUM-PERS et JUM-COUPLE) ; si ces conditions sont vérifiées, E aura repris tout ou partie des liens de parenté, union, biographie, de F >

Les conditions d'applicabilité du jumelage sont en principe vérifiées si le couplage a été bien défini ; on est cependant contraint de protéger la de données, notamment dans le cas des jumelages forcés où l'utilisateur pourrait fort bien commettre des erreurs irréparables. Les conditions définies plus loin dans les deux procédures de base JUM-PERS et JUM-COUPL sont assez générales ; on conçoit cependant la difficulté de définir un langage général permettant à la fois d'exprimer et d'engendrer ce type de contrôle quel que soit le type d'acteur envisagé.

L'utilisation de la procédure JUMEL-ELEM définie ci-après est évidente lors du traitement des composantes d'une solution ; nous détaillerons pas cet algorithme.

```
procédure  JUM-ELEM (E, F:acteur ; TYPACT:t-act)

début
|
|    si  type  (E:act-personne)
|
|      alors
|      |
|      |      < E et F sont de type 'act-personne' >
|      |
|      |        JUM-PERS (E,F,TYPACT)
|      |
|      |      < le jumelage n'a été effectué que si les parents de E et F sont iden-
|      |        tiques, ou si l'un ou l'autre de ces couples parents est indéfini ;
|      |        dans ce cas, F a été supprimé et E a repris les liens UNION et
|      |        BIOGRAPHIE de F ; E.PARENTS n'a été modifié que si ce couple était
|      |        indéfini à l'appel de la procédure >
|      |
|      sinon
|      |
|      |      < E et F sont de type act-couple >
|      |
|      |        JUM-COUPLE (E,F,TYPACT)
|      |
|      |      < l'action est ineffective s'il existe au moins deux conjoints différents
|      |        (époux ou épouse) dans les deux couples considérés ; dans le cas
|      |        contraire, F a été supprimé, et E a repris les liens ENFANTS et
|      |        BIOGRAPHIE de F ; E.EPOUX ou E.EPOUSE n'ont été modifiées que si elles
|      |        étaient indéfinies à l'appel de la procédure >
|      |
|      finsi
|
fin
```

Considérons la procédure de jumelage des acteurs de type 'act-personne' le jumelage n'est possible sans problèmes de mise à jour généalogique, si les deux couples parents sont identiques, ou si l'un des deux couple parents n'est pas défini (ou les deux).

```
procédure JUM-PERS(E,F:act-personne ; TYPACT:t-act)
var I:entier
début
    Si E.PARENTS = nult ou F.PARENTS = nult ou E.PARENTS = F.PARENTS
      alors
        Si E.PARENTS = nult
          alors < (1) >
            E.PARENTS := F.PARENTS
          finsi
        E.UNION := E.UNION ∪ F.UNION
        E.BIOGRAPHIE := E.BIOGRAPHIE ∪ F.BIOGRAPHIE
        I := FICH (F)  (*)
      < F ∈ TYPACT.V-TEVEN[I].PERM >
        supp (F ∈ TYPACT.V-TEVEN[I].PERM)
      finsi
fin
```

(1) : on ne modifie E.PARENTS que si cette information est indéfinie (risque de perte d'information si F.PARENTS est indéfini).

Pour la procédure de jumelage des couples, la condition est plus complex le jumelage n'est impossible que s'il existe au moins un conjoint différ dans les deux couples.

```
procédure  JUM-COUPLE (E,F:act-couple ; TYPACT:t-act)

var I:entier ; EPX-OK, EPS-OK:booléen

début
|     EPX-OK := E.EPOUX = nult ou F.EPOUX = nult ou F.EPOUX = E.EPOUX
|     EPS-OK := E.EPOUSE = nult ou F.EPOUSE = nult ou F.EPOUSE = E.EPOUSE
|     Si EPX-OK et EPS-OK
|        alors   < le jumelage est possible >
|        |     Si E.EPOUX = nult
|        |        alors
|        |        |       < (1) >
|        |        |         E.EPOUX := F.EPOUX
|        |        finsi
|        |     Si E.EPOUSE = nult
|        |        alors
|        |        |       < (1) >
|        |        |         E.EPOUSE := F.EPOUSE
|        |        finsi
|        |     E.ENFANTS := E.ENFANTS ∪ F.ENFANTS
|        |     E.BIOGRAPHIE := E.BIOGRAPHIE ∪ F.BIOGRAPHIE
|        |     I := FICH (F)
|        |   < F ∈ TYPACT.V-TEVEN[I].PERM >
|        |     supp (F ∈ TYPACT.V-TEVEN[I].PERM)
|        finsi
fin
```

(1) : on ne modifie E.EPOUX et E.EPOUSE que si ces informations sont indéfinies.

3.3.10. Caractéristiques générales de l'implémentation-restricti

3.3.10.a) Le système hôte SOCRATE

Nous ne rappelons ici que les possibilités standard de ce système, dont l'utilisation a largement conditionné la réalisation des objectifs généraux définis en 3.1.

Selon ses caractéristiques les plus générales, SOCRATE peut être défini comme un SGBD permettant la définition et l'exploitation conversationnell ou batch de bases de données de type hiérarchique ou réseau [B-DAT]. Les fonctions de définition et d'exploitation de la base sont accessibles via des langages spécialisés, de haut niveau, qui sont essentiellement :

- Le langage de définition de structure : qui est utilisé pour la définition des caractéristiques, des fichiers de base et des relations entre fichiers. Sans entrer ici dans le détail de ce langage, nous soulignerc la possibilité d'ajout dynamique de structures offerte à ce niveau, qui permet l'extension de la base par la définition de nouveaux fichier
- Le langage de requêtes : qui permet l'interrogation, la modification conversationnelle des données enregistrées dans la base.

SOCRATE offre aussi des moyens standard de développement de logiciels d'exploitation de la base, dont le plus intéressant est sans doute le macro-générateur. Comme tous les outils de ce type, il permet l'expansio d'un code particulier à partir du corps d'une macro-instruction et des valeurs des paramètres effectifs. Le macro-générateur de SOCRATE permet l'expansion de code en langage de requêtes ; son utilisation est donc possible pour :

- la production de requêtes complexes, exécutées en temps-réel ; ce mode d'utilisation du macro-générateur permet en particulier la définition de langages de commande de haut niveau, accessibles à des non informaticiens.
- la production de code compilable dans des procédures cataloguées ; ce mode d'utilisation permet la production automatique du code de procédu complexes, à partir de langages spécialisés de haut niveau.

Les possibilités d'ajout dynamique de structure sont utilisées pour compléter la structure noyau de MERCURE lors de la définition des fichiers acteur et évènement (cf. 3.3.4.a).

Le macro-générateur est utilisé pour la gestion de toutes les commandes d'exploitation ; le code expansé consiste essentiellement en deux catégories d'appels de procédures :

- le contrôle des paramètres de la macro-instruction ; ces procédures standard délivrent les messages d'erreur appropriés.
- l'exécution des algorithmes standard correspondant à une fonction.

Cette implémentation des commandes s'avère très fiable et performante ; elle est fondée sur une grande modularité des fonctions de base de MERCURE le code engendré par le macro-générateur n'étant alors qu'une structure de contrôle qui est très rapidement générée et compilée en temps réel.

3.3.10.b) Restrictions d'implémentation

MERCURE a été implémenté sous la version 1.4 de SOCRATE disponible sur le 360-45 du CICG ; la superposition de ce système à CP/CMS n'était pas de nature à donner d'excellentes performances d'exécution, mais nous a permis de bénéficier du bon contexte de travail propre à ces deux systèmes. La portabilité du logiciel sur une autre machine paraît bonne dans la mesure où les nouvelles versions de SOCRATE implémentées (ou en cours d'implémentation) sur les sites envisagés, sont essentiellement des extensions (les quelques restrictions observées ne concernent pas les outils de base utilisés dans la version actuelle de MERCURE).

Les principales restrictions concernent des commandes d'exploitation manquantes, comme le chargement de la base à partir de fichiers, et la gestion des modèles de variation qui est encore peu developpée (cf. 3.3.5.a) Enfin, une limitation importante réside dans le type des informations que l'on peut introduire dans la base ; elles ne peuvent consister qu'en actes individuels (personne + évènement), ou qu'en couples (conjoints + évènement). Cette limitation peut être facilement levée, mais interdit pour l'instant l'exploitation des recensements, par exemple, où la structure des données est celle d'une famille complète.

3.3.11. Conclusion

Le développement de ce prototype nous a permis de définir une architectu générale pour une base de données démographiques, ainsi qu'une première implémentation des algorithmes généraux de couplage et de jumelage des informations. L'expérimentation conduite à partir de jeux d'essais restreints ne permet pas d'évaluation quantitative quant à l'efficacité du logiciel ; elle nous a toutefois permi une évaluation qualitative des fonctions proposées et des options générales qui ont été prises, et qui paraissent assez satisfaisantes au dire des utilisateurs potentiels.

MERCURE est néanmoins un logiciel assez volumineux, de par la multiplici de ses fonctions ainsi que de l'option de généralité qui a été choisie ; il est vraisemblable que ses performances d'exploitation auront à souffrir de ce dernier aspect par rapport à celles d'un produit "ad-hoc".

La plupart des limitations évoquées plus haut peuvent être facilement levées,mais notre objectif essentiel demeure ailleurs, par exemple dans la recherche et l'implémentation d'algorithmes plus efficaces pour résoudre la phase de sélection des solutions de jumelage (cf. 2.7.5.d) ; dans l'état actuel du prototype, on peut envisager le traitement de tout application de reconstitution de population à partir de données fragmentaires (actes paroissiaux, état civil, etc.), ce qui couvre déjà une grande majorité des travaux effectués dans ce domaine.

CHAPITRE 3.4 Application du logiciel MERCURE à la constitution d'une base de données démographiqhes

3.4. Application du logiciel MERCURE à la constitution d'une base de données démographiques :

- Rappelons que le point de départ de cette étude a été une application en démographie historique ; cette expérience nous a permis de dégager quelques unes des spécifications essentielles de la stratégie de reconstitution de la population mise en oeuvre dans MERCURE. Nous ne ferons ici qu'une présentation succinte de cette utilisation que l'on peut retrouver dans [B.CH2] ,[B.CH4] et [A.BOC]. Nous nous attachons ici surtout à montrer les différentes phases de la construction de la base.

3.4.1. La structure de représentation de la population

- Les opinions sont assez diverses sur ce point ; la notion de population varie du simple fichier de personnes à la base de données généalogique, notre objectif a donc tout d'abord été de réaliser une synthèse de ces différentes acceptations, et d'offrir une structure de données permettant le maximum de variété dans les investigations ultérieures (la reconstitution n'est évidemment qu'une étape).

- La structure choisie permet les niveaux d'accès suivants :

  - les événements, pour les opérations classiques d'analyse statistique liées à la mortalité, la natalité, la nuptialité etc...

  - les biographies, pour les études statistiques relatives au comportement des individus (espérance de vie, âge au mariage etc...)

  - les familles (couples + enfants) pour toutes les études relatives à cette structure essentielle : fécondité, durée de vie des familles etc...

- les généalogies ; MERCURE a été l'un des tout premiers systèmes de reconstitution de population à offrir cette possibilité (cf. aussi [B.SK2]).

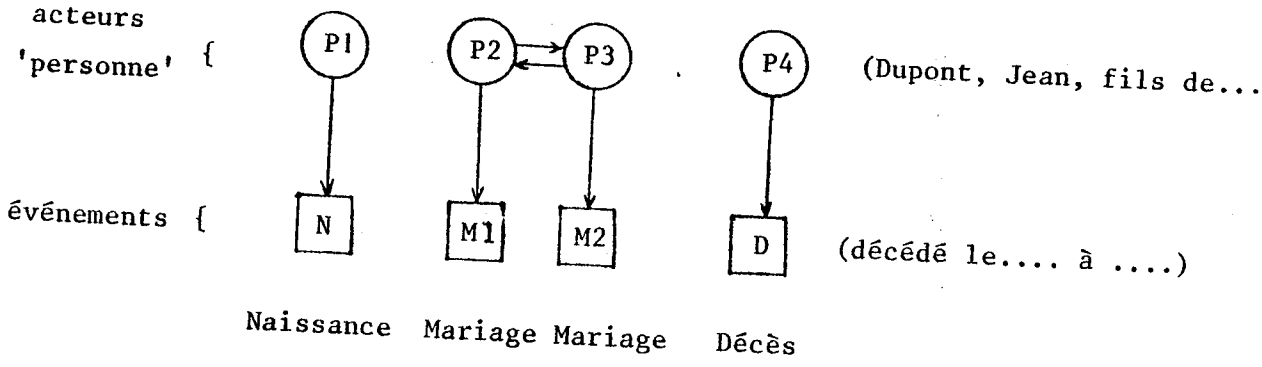
On peut obtenir des généalogies ascendantes ou descendantes.

Ce niveau d'accès ouvre des perspectives nouvelles d'investigation comme l'étude de l'évolution sociale des familles, par exemple.

3.4.2. Structuration des données au chargement de la base :

- La structure de base acteur-événement est fondamentale dans toutes les opérations qui suivent ; dès le chargement de la base (cf. 3.3.4), les données originelles (actes) sont organisés en structures élémentaires du type suivant :

fig. 1



- Les liens verticaux sont des liens biographiques (cf. BIOGRAPHIE dans le type "act-personne"), les liens horizontaux correspondent à des unions ; (P2, P3, M1, M2) représente la structure obtenue lors du chargement d'un acte de mariage.

3.4.3. La phase de reconstitition des biographies

- Cette phase réunit les opérations de couplage et de jumelage propres à la définition d'un modèle biographique : toutes les relations de couplage définies à ce niveau expriment des règles de compatibilité biographique entre événements concernant une même personne : date de naissance ≤ date de décès etc...

- La constitution des classes de cohérence s'effectue donc selon ces règles, et toutes les possibilités d'interaction sont disponibles à ce niveau pour valider un modèle jugé satisfaisant (cf. 3.3.5. et 3.3.6.). Rappelons que les classes sont constituées indépendamment des données, ce qui autorise une grande souplesse de travail. On obtient par exemple, à partir de la fig. 1 :

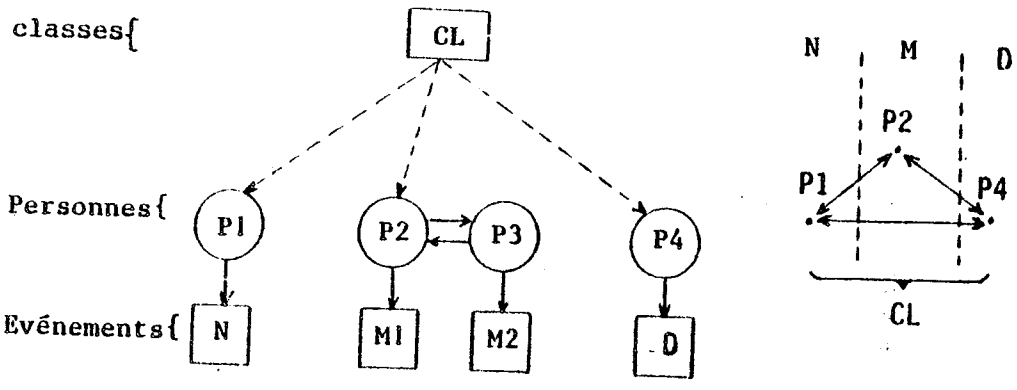


fig. 2

Si le processus de sélection confirme cette solution, le jumelage (autour de P2 par exemple) va produire la structure suivante :

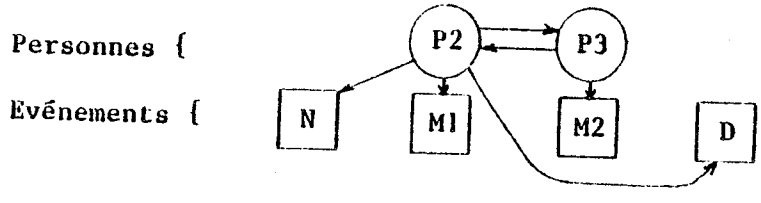


fig. 3

Le transfert de liens correspondant à cette opération a enrichi la biogr phie de P2, alors que les enregistrements 'acteur' relatifs à P1, P4 ont disparu, ainsi que la classe de cohérence d'origine : le jumelage est irréversible.

3.4.4. La génération des couples

Une reconstitution des familles, et à fortiori des généalogies, n'est possible que si les actes mentionnent les couples parents des individus concernés ; on va donc engendrer une nouvelle catégorie d'acteurs : les couples.

On en distingue deux types:

- Les couples mentionnés dans un acte : ce sont le plus souvent les parents de la personne concernée ; l'enregistrement couple est créé avec un lien 'Père' vers l'acteur d'origine ; l'événement créé est une mentio portant la date de l'acte, et toute information récupérable sur les pare (profession, lieu de résidence, mention de vie ou de décès du père, de l mère...).

- Les couples déclarés : ils sont construits à partir des actes de mari des liens sont créés avec les acteurs composants (cf. EPOUX EPOUSE dans type 'act-couple'). Les événements créés sont des déclarations où l'on pro cède de la même façon que pour les couples mentionnés.

Remarquons qu'un acte de mariage donne lieu à la production de trois couples :

- un couple déclaré
- deux couples mentionnés : les parents des conjoints

Après génération des couples, la figure 3 devient :

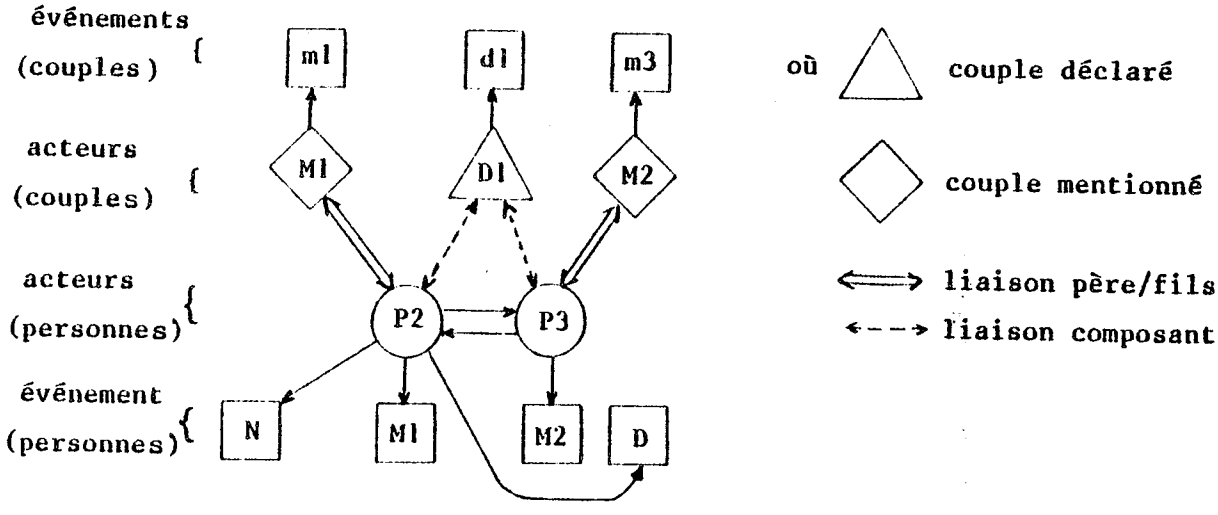


fig. 4

3.4.5. La phase de reconstitution généalogique :

Dans la même étape, on va reconstituer à la fois les familles et les généalogies. On va ici tout d'abord coupler les deux types de couples, selon un nouveau modèle qui correspond à la recherche de relations de filiation; les dates jouent encore ici un grand rôle :

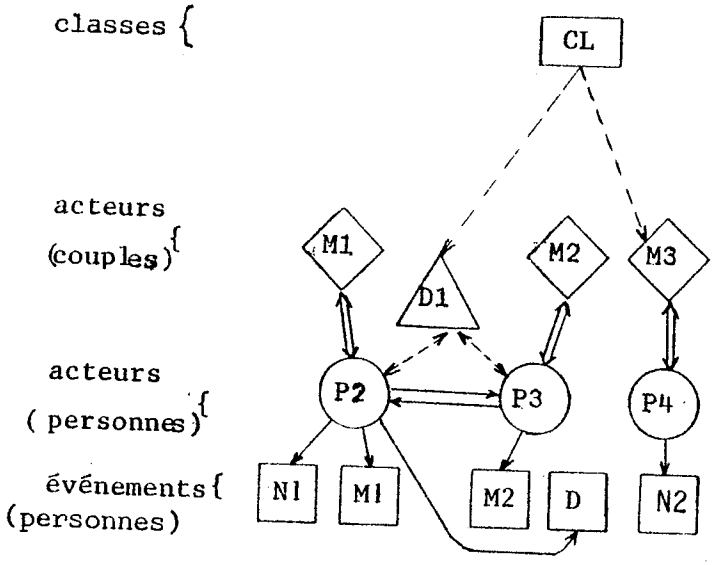
- vérifier qu'une date de mention à la naissance n'est pas antérieure à la date de formation du couple (sauf avis de naissance illégitime).

- vérifier qu'une date de mention à la naissance n'est pas postérieure à la date de décès de la mère (si elle est disponible) etc...

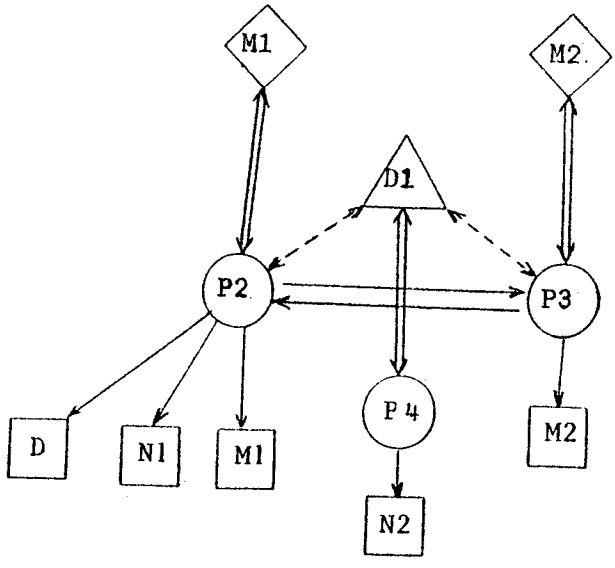
On voit donc qu'à la création des couples, on a avantage à profiter au maximum des résultats de la phase biographique ; aux couples déclarés on associe un intervalle [date fondation, Min(date décès père, date décès mère)] et on considère comme valide toute mention entrant dans l'intervalle, et dont l'identité des composants concorde.

Ces règles étant définies, on procède comme au niveau biographique par mise au point successive du modèle. Dans ce cas, le jumelage introduit des liaisons intergénérationnelles de parenté ; tout jumelage d'un couple déclaré revient à attribuer un enfant à ce dernier. A l'issue de cette opération, on a donc le graphe généalogique de la population (aussi complet que l'état des données et la qualité du modèle le permettent)

Exemple : (les évènements relatifs aux couples ne sont pas représentés)



Si la décision de jumeler D1 et M3 est prise, on obtient :



Où M1 et M2 représentent à présent les couples grand parents de P4; D1 = (P2, P représente les parents de

fig. 5

+++

3.4.6. Conclusion

On voit que le processus de travail est identique dans les deux phases biographique et généalogique ; seuls les modèles changent et la description des types d'acteurs ou d'évenement. Les algorithmes généraux et le système de définition/modification interactives des modèles sont donc utilisables durant ces deux phases essentielles du processus de reconstitution.

La séparation entre les phases de reconstitution biographique et généalogique peut-être critiquée en ce sens que les choix effectués au stade biographique déterminent fortement les possibilités obtenues au stade suivant ; un exemple typique est celui de l'attribution de la date de décès d'une mère, qui conditionne évidemment la progéniture qui pourra lui être attribuée. Il parait donc souhaitable de s'orienter vers une stratégie plus souple dans laquelle les décisions de jumelage d'informations biographiques pourraient être en partie au moins (lorsque les solutions obtenues sont estimées comme assez incertaines), reportées au stade de la généalogie.

Une telle stratégie, qui conduirait évidemment à augmenter le nombre de possibilités analysées, implique la recherche d'algorithmes de sélection très efficaces ; ceci rejoint les remarques déjà effectuées dans le chapitre 2.7.

CHAPITRE 3.5. - Présentation du logiciel de contrôle des informations source

3.5. Présentation du logiciel de contrôle des informations sourc

3.5.1. Nécessité d'un tel logiciel

Dans les chapitres II.2.3. et II.2.4., nous avons insisté sur la néce sité de ne pas dégrader davantage les informations source lors de leur tra fert sur support informatique ; nous avons également rappelé les principa types de contrôles ainsi que leurs limites. Les contrôles directs corres- pondent principalement à des vérifications de type, alors que les contrôle indirects consistent à vérifier des relations entre les valeurs de plusieu caractéristiques. La réalisation des contrôles directs est d'autant plus simple que le langage choisi pour implémenter ces opérations offre de mei leures possibilités standard de définition et de validation de types ; si des langages de haut niveau tels que ALGOL 68 [B.ALG] ou ADA [B.CII] présentent d'excellentes caractéristiques à cet égard, ils ne sont (ou n'étaient) pas disponibles au moment où nous avons réalisé le logiciel pr senté dans ce chapitre. Nous avons donc choisi de réaliser ce logiciel en langage PL/1 , qui nous paraissait, parmi les langages disponibles, le mieux adapté à cette fonction [B.CAV].

3.5.2. Le générateur de programmes de contrôle :

Si on considère des fichiers relatifs à des applications réelles, dont la description comporte plusieurs dizaines de caractéristiques, l'écriture complète des différentes procédures de contrôle est une tâche extrêmement longue et fastidieuse. Il nous a donc paru plus intéressant de définir un logiciel qui, à partir d'une spécification externe des contrôles à effectuer pour un fichier particulier, génère automatiquement l'ensemble des procédures de validation correspondantes ; le code produit est un programme PL/1 compilable.

Les paramètres de spécification des contrôles fournis au générateur sont de deux types :

- La description D du fichier à traiter, qui définit l'ensemble des caractéristiques à contrôler, ainsi que leur type.

- Les relations internes existant éventuellement entre les différentes caractéristiques, qui permettent la définition des contrôles indirects.

On peut, de plus, profiter des opérations de contrôle pour effectuer un certain nombre d'investigation supplémentaire sur les données (cf. ci-après).

3.5.3. Les langages de spécification des différents paramètres :

3.5.3. a) Définition de la description du fichier source :

Nous nous sommes inspiré du langage de définition de structure de SOCRATE ; un certain nombre de types ont cependant été rajoutés pour nos besoins particuliers. La syntaxe de ce langage est donnée ci-dessous ; les symboles terminaux sont soulignés.

Syntaxe :

description : { *bloc* }$^+$ |
bloc : *identificateur* début { *caractéristique* }$^+$ fin
caractéristique : *caract-simple* | *bloc*
caract-simple : *identificateur type*
type : mot [ℓ] | texte [ℓ] | num (v_1 A v_2] | dans [*liste-val*]
constante "*val*" | etiquette [ℓ] | séquence " v_3 " |
idem *identificateur* | code [ℓ_1 ℓ_2]
liste-val : *val* { , *val* }*

Lexicographie :

identificateur : chaîne de caractères $\in$ {A..Z, 0..9, -} , de longueur ≤ 16, commençant par une lettre.

val : chaîne alphanumérique (sans espace) de longueur quelconque

l, l_1, l_2, v_3 : entier positif.

v_1, v_2 : valeurs numériques quelconques (réelles ou entières), signé ou non ; $v_1 < v_2$

A chaque caractéristique ainsi définie correspond une procédure de contrôle de type.

La structure de bloc n'a été introduite que dans le but de permettre un regroupement logique d'informations ; tout identificateur doit-être unique dans la description.

Nous présentons ci-dessous les différents types de variables prévus :

- mot (l) : ce type correspond à des valeurs qui sont des chaînes de caractères appartenant à l'ensemble { A,..., Z, - , ' }, cadrées à gauche, de longueur maximale l. Les positions éventuellement libres à droite contiennent des caractères "espace".

Exemple :

NOM-PERE mot (16)

+++

- num (v_1 a v_2) : ce type correspond à des valeurs numériques (réelles ou entières), signées ou non, comprises entre v_1 et v_2 (bornes comprises)

On a obligatoirement $v_1 < v_2$.

Exemple :

AGE num (18 A 60)

+++

- texte (l) : ce type correspond à des valeurs qui sont des chaînes de caractères alphanumériques quelconques, de longueur ≤ l. Aucune propriété particulière n'est vérifiée pour les caractéristiques de ce type.

Exemple :

COMMENTAIRE texte (80)

+++

- dans (liste-val) : ce type correspond à des valeurs qui sont des chaînes alphanumériques appartenant obligatoirement à l'ensemble cité dans liste-val.

Exemple :

SITUATION dans (MARIE, CELIBATAIRE, VEUF).

+++

- code (l_1 l_2) : ce type est comparable au type dans mais dispense l'utilisateur de fournir la liste des valeurs autorisées lors de la définition de la description du fichier.

- Etant donné l'ensemble V des valeurs d'une caractéristique, un codage V' des éléments de cet ensemble, on range sur support externe l'ensemble des couples (v', v) représentant la fonction de codification de V. Sur le fichier à contrôler apparaissent seulement les valeurs codifiées appartenant à V' ; ces valeurs sont des chaînes alphanumériques de longueur $\leq l_1$, les valeurs originales étant des chaînes alphanumériques de longueur $\leq l_2$.

La procédure de validation du type consiste à vérifier que la valeur contrôlée appartient à V'.

Exemple :

PROFESSION code (3 13)

Les valeurs de PROFESSION sont des chaînes alphanumériques de longueur ≤ 3 appartenant à un ensemble déterminé. Cet ensemble est représenté en mémoi secondaire avec en correspondance, pour tout élément, la valeur non codifi de la caractéristique, qui est ici une chaîne alphanumérique de longueur maximale 13.

- Soit par exemple l'élément suivant du code :

$l_1 = 3$ $l_2 = 13$

PLB , PLOMBIER

+++

Ce type est surtout utile pour contrôler l'appartenance de valeurs à des ensembles de cardinal élevé (plusieurs centaines d'éléments). Dans ces conditions, le type dans est évidemment très incommode. De plus, le type code permet la modification de la liste de valeurs sans affecter le programme de contrôle, ce qui n'est pas le cas du type dans.

Constante "val" : Ce type correspond à un ensemble de valeurs limité à un seul élément. Bien que logiquement redondant avec le type dans (on aurait alors liste-val = val), il a été introduit pour des raisons d'optimisation du code engendré (pas de table de valeurs créée).

Remarquons qu'en donnant à 'val' des valeurs du type "chaîne de caractères espace", on peut contrôler des zones de cadrage définies à la saisie entre les différentes informations.

Exemple :

BLANC constante " "

- Etiquette (1) et séquence "v_3" : Ces deux types de caractéristiques permettent la définition d'un contrôle de séquence lorsqu'un enregistrement logique du fichier est constitué de plusieurs enregistrements physiques consécutifs. Si cela est le cas, la description du fichier doit faire apparaître cette décomposition ; chaque enregistrement physique doit comporter une caractéristique de type étiquette et une caractéristique de type séquence. La procédure engendrée vérifie que, pour un même enregistrement logique, toutes les valeurs de variables de type étiquette sont identiques (vérification que tous les enregistrements physiques lus sont relatifs à un même enregistrement logique), et que les variables de type séquence ont les valeurs prévues (vérification du nombre et de l'ordre des enregistrements physiques constituants). L'exemple complet de définition de description donné plus loin illustre la mise en oeuvre de ce type particulier de contrôle .

- Idem identificateur :

- Cette clause permet d'indiquer que la caractéristique déclarée est du même type que celle référencée par 'identificateur'. Dans la description, identificateur doit correspondre à une caractéristique définie avant celle qui lui fait référence par la clause idem. Il s'agit donc essentiellement d'une commodité syntaxique permettant d'éviter la redéfinition de types complexes.

Exemple :

```
SITUATION dans (célibataire, veuf, marié, divorcé)
.
.
.
SITUATION-MERE idem SITUATION
```

+++

Exemple complet de définition d'une description :

- Soit un fichier dont les enregistrements logiques correspondent à une description de personnes. Chaque enregistrement logique est saisi sur deu cartes perforées ; pour éviter tout mélange accidentel, un code fichier 'FP' est reporté en tête de chaque carte. Chaque enregistrement logique est distingué des autres par un code enregistrement sur trois caractères qui est le même sur chaque carte constituante (étiquettes E1, E2) ; l'ordre de succession de ces cartes est défini par les codes séquence S1 et S2 respectivement égaux à 1 et 2.

La figure 1 illustre la définition de la description de ce fichier.

```
PERSONNE
   Début
      CARTE1
         début
            CODE-FICH 1 constante "FP"
            E1 étiquette (3)
            S1 séquence "1"
            NOM mot (16)
            NOM-JFILLE mot (16)
            PRENOM code (3 13)
            SITUATION dans (célibataire, veuf, marié, divorcé)
            B1 constante "  "
            SEXE dans (M,F)
            B2 idem B1
            PROFESSION code (3 13)
            B3 idem B1
            DATE-NAISSANCE
               début
                  JOUR-N num (1 a 31)
                  MOIS-N num (1 a 12)
                  ANNEE-N num (1800 à 1980)
               fin
            DATE MARIAGE
               début
                  JOUR-M idem JOUR-N
                  MOIS-M idem MOIS-N
                  ANNEE-M idem ANNEE-N
               fin
            B4 idem B1
            AGE num (0 a 100)
            B5 constante "                "
         fin

      CARTE2
         début
            CODE-FICH 2 constante "FP"
            E2 étiquette (3)
            S2 séquence "2"
            B6 idem B1
            ADRESSE texte (68)
            B7 idem B1
            PROFESSION code (4 16)
         fin
   Fin |
```

Fig. 1

3.5.3. b) Définition des contrôles directs de présence et des contrôles indirects

Le langage permet de définir ces différents contrôles ainsi que les messages d'erreurs correspondants.

Syntaxe :

contrôles : { *alternative* }* |
alternative : *si* condition *et-non* condition *alors* message *fsi* | (1)
si condition *alors* message *fsi* (2)
condition : *condition-simple* { *op-logique condition*}*
condition-simple : *identificateur présent* | *identificateur absent* |
expression-simple
expression-simple : *identificateur op-comp identificateur*
op-logique : *et* | *ou* | *non*
op-comp : = | =| ≥ | >=| < | < =
message : ' *chaine-alphanumérique* '

Lexicographie :

identificateur : chaîne de caractères appartenant à la liste d'identificateurs donnée par la description du fichier

chaîne-alphanumérique : sa longueur est quelconque.

Une alternative de la forme (1) est interprétée de la façon suivante :

condition-1 ==>condition-2

Soit : si condition-1

```
alors
    si non condition-2
        alors
            erreur : = vrai

            produire (message)
        finsi
finsi
```

On peut naturellement définir des équivalences logiques en définissant les implications inverses :

Si condition-1 et-non condition-2 alors message-1 fsi
Si condition-2 et-non condition-1 alors message-2 fsi

Une alternative de type (2) est interprêtée de la façon suivante :

```
si condition
alors
    erreur : = vrai

    produire (message)
finsi
```

Les mots réservés présent et absent permettent, à l'aide des expressions de type (1) ou (2), de définir des contrôles de présence.

Exemple :

si NOM absent alors 'nom absent' fsi

définit le contrôle de présence obligatoire des valeurs de la caractéristique NOM et le message d'erreur correspondant.

si SITUATION = 'marié' et SEXE = 'F' et-non
NOM-JFILLE présent
alors 'nom de jeune fille absent' fsi

définit un contrôle indirect de présence des valeurs de la caractéristique NOM-JFILLE et le message d'erreur correspondant ; ce contrôle vérifie l'implication :

SITUATION = 'marié' et SEXE ='F'⇒NOM-JFILLE présent

+++

Enfin, les alternatives de type (1) permettent la définition de contrôles indirects de valeurs.

Exemple :

si ANNEE-M présent et ANNEE-N présent et-non ANNEE-M> ANNEE-N
alors 'années de naissance et de mariage incohérentes' fsi

+++

3.5.3. c) Statistiques de présence simultanée des valeurs de plusieurs caractéristiques :

Cette opération peut-être effectuée à l'occasion des opérations de contrôl si l'utilisateur le désire. Elle est destinée à fournir le taux de présenc simultanée des valeurs d'un ou plusieurs groupes de caractéristiques ; ces résultats sont souvent d'un grand intérêt dans notre type d'application, car ils contribuent à donner une bonne idée d'ensemble de la qualité des informations à traiter.

- La syntaxe de définition de ces tests est la suivante :

test-groupes : {*groupe* }* |
groupe : {*identificateur*}+*

lexicographie :

identificateur : appartient à la liste des identificateurs définie par la description du fichier. Deux identificateurs identiques ne peuvent appartenir au même groupe.

Exemple :

NOM PRENOM *
NOM PRENOM PROFESSION * |

En fin de traitement, les taux de présence simultanée des groupes NOM PRENOM et NOM PRENOM PROFESSION, sont édités.

+++

3.5.4. Modalités de mise en oeuvre du générateur :

La définition des paramètres de contrôle s'effectue de manière conversationnelle, dans un ordre imposé par le programme :

a) Définition de la description du fichier (cf. 3.5.3.a)
b) Définition des contrôles directs de présence (cf. 3.5.3.b)
c) Définition des contrôles indirects de présence (cf. 3.5.3.b)
d) Définition des contrôles indirects de valeur (cf. 3.5.3.b)
e) Définition des tests de présence simultanée (cf. 3.5.3.c)

Seul le paramètre a) est obligatoire ; après contrôle syntaxique, ces informations sont fournies en entrée au générateur de programmes. Lorsque la description comporte des caractéristiques de type code, la saisie des fonctions de codification correspondantes s'effectue séparément, (le générateur donne, après analyse de la description, la liste des fichiers code à produire). La génération de programme s'effectue par l'interclassement de segments fixes (généraux à tous les programmes de contrôle, comme la vérification des types de base), et de segments variables produits par l'interprétation des paramètres. L'annexe II.a illustre le fonctionnement du générateur de programmes de contrôle.

3.5.5. Caractéristiques des programmes de contrôle engendrés :

Tout programme de contrôle ainsi engendré gère les fichiers suivants :

- le fichier des données à contrôler (entrée)
- le fichier des enregistrements corrects (sortie)
- le fichier des enregistrements erronés (sortie)
- le fichier des diagnostics et statistiques diverses (sortie)
- le fichier des points de reprise sur incident (entrée-sortie)

En raison du volume des informations traitées, une procédure de reprise sur incident est incorporée systématiquement aux programmes de contrôle ; en début de session, elle permet la reprise du traitement à partir de positions sur les différents fichiers, enregistrées dans le fichier des points de reprise.

Si des caractéristiques de type code sont à contrôler, le programme vérifie que les fichiers correspondants ont bien été créés.

Pour chaque enregistrement erroné, il y a production de l'impression suivante sur le fichier des diagnostics d'erreurs :

- le contenu intégral de l'enregistrement logique
- pour chaque erreur décelée :
  a) le nom de la variable
  b) sa valeur
  c) le diagnostic d'erreur

Les informations a) et b) ne sont fournies que pour les contrôles directs, la localisation des erreurs n'étant en général possible que pour ce type de contrôle.

En fin de traitement, un état récapitulatif est édité :

- nombre de cartes lues : n1
- nombre de cartes en désordre : n2
- nombre d'enregistrements logiques analysés (cartes en ordre) : n3
- nombre d'enregistrements logiques corrects : n4
- nombre d'enregistrements logiques erronés : n5
- taux d'enregistrements logiques erronés : $\frac{n5}{n3} \times 100$
- taux d'absence brut de chaque caractéristique élémentaire (ramené à n3)
- si cela a été demandé, édition du taux de présence simultané de groupes de caractéristiques.

Nous donnons en annexe II.b un exemple de fonctionnement d'un programme de contrôle. De par la nature des traitements effectués, l'aspect conversationnel y est très limité ; il se résume à choisir, en début d'exécution, le positionnement sur les fichiers :

- départ en début de fichiers
- départ sur point de reprise

3.5.6. Restrictions d'implémentation :

Le générateur ainsi que les programmes engendrés ont été conçus pour fonctionner sous CP/CMS ; cela introduit naturellement des limitations quant à leur portabilité. De ce point de vue, le problème le plus important est relatif aux organisations de fichiers et aux fonctions d'entrée-sortie :

- dans la version utilisée, les fichiers CMS sont standard : support disque, enregistrements de longueur fixe égale à 80 caractères, facteur de blocage de 10.

- les fonctions d'entrée-sortie standard autorisent l'accès direct (relatif) aux enregistrements, mais ne sont pas conformes à la syntaxe PL/1 (interface d'entrée sortie PL/1-CMS).

Dans la syntaxe présentée dans les chapitres 3.5.3. a et 3.5.3. b, la plupart des symboles terminaux utilisés ont été modifiés (car jugés plus explicites) par rapport à ceux effectivement disponibles dans la version implémentée ; leur prise en compte nécessiterait qu'une modification mineu du générateur, et n'hypothèque en rien les possibilités affectives présentées plus haut.

3.5.7. Conclusion

Ce logiciel permet de procéder très rapidement à la définition de programmes de contrôle ; des extensions de possibilités sont néanmoins envisageables, la plus importante étant certainement la production de programmes interactifs de saisie-contrôle des fichiers.

Nous pensons qu'une telle réalisation est possible à partir des mêmes principes que ceux déjà utilisés pour cette première version, et est souhaitable compte-tenu de l'évolution des techniques de saisie (utilisation des terminaux écran-clavier) visant à réduire le délai de correction.

CHAPITRE 3.6. - Présentation du transducteur phonétique PIAFPHO

3.6. Présentation du transducteur phonétique PIAFPHO :

3.6.1. Nécessité d'un tel logiciel

Beaucoup d'applications présentent des données linguistiques comme les noms patronymiques, de lieux, les prénoms etc... S'il est parfois possible de les codifier en fonction de listes préétablies (professions, prénoms...), il n'en va généralement pas ainsi pour les noms patronymiques dont la grande diversité échappe à toute règle. Lorsque le couplage porte sur des fichiers de personnes (démographie etc...), ces informations revêtent évidemment une importance cruciale ; on constate malheureusement aussi qu'elles figurent souvent parmi les moins fiables (cf. 2.3.1., 2.3.2.). Une solution classique consiste à substituer à la comparaison directe des noms, celle de leur interprétation phonétique (modèle de variation, cf. 2.4.2.a). L'objectif du logiciel PIAFPHO est la production automatique de transductions phonétiques hierarchisées (cf. chapitre suivant). Les données traitées étant généralement fortement liées à un contexte linguistique particulier (patois, accent du terroir, époque considérée), il est souvent difficile d'appréhender d'emblée le modèle de transduction ; de ce point de vue, il est extrêmement précieux de disposer d'un logiciel interactif de mise au point des paramètres linguistiques de transduction. PIAFPHO bénéficie des mêmes possibilités d'interaction que le logiciel PIAF (Programme Interactif d'Analyse du Français) [B.COU] dont il constitue un cas d'application particulier. Nous n'en présentons ici que les principes de base, on pourra trouver plus de détails dans [B.CH5] et [B.COU].

3.6.2. Les différents niveaux de transduction :

- PIAFPHO a été conçu pour produire trois niveaux de transduction ; des modifications mineures du programme pourraient, selon les besoins particuliers, ramener ce nombre à deux ou un. Chaque niveau de transduction correspond à un niveau d'interprétation particulier de la chaîne d'entrée permettant des regroupements de plus en plus importants des noms en classes disjointes (ensembles de noms délivrant la même interprétation phonétique au niveau considéré). Un objectif important est d'obtenir une transduction unique pour chaque nom, à un niveau donné ; ceci conduit naturellement à des partitions hierarchisées de l'ensemble des noms.

3.6.2. a) Premier niveau de transduction

Elle est opérée selon des critères de prononciation proches de ceux du français actuel ; l'invariant obtenu ne permet de déceler que des variations mineures :

Exemple :

COUTTE et COUTES donnent K.OU.T
BRENIER et BREGNER donnent B.R.E.GY.E1
où 'GY' est la codification du phonème "gn", et "E1" celle de "é".

+++

A ce niveau déjà, il y a donc confusion volontaire entre un certain nombre de phonèmes ; par exemple toutes les variantes du "O" sont traduites par 'O', le "é" et le "è" sont traduits par 'E1' etc...

3.6.2. b) Second niveau de transduction :

Cette transduction prend en compte certaines graphies dont la prononciation est naturellement ambiguë, ou sujette à variation du fait de l'accent du terroir.

Exemple :

Dans notre application, on prend en compte certaines variations liées à l'accent savoyard :

- Les phonèmes "IN" et "AN" sont confondus et traduits par "2N" ; ainsi

  OLLINS et OLLANS donnent O.L.2N au deuxième niveau.

- Les phonèmes "OU" et "O" sont confondus et traduits par 'O' ; ainsi

  COUTTE et COTTE donnent K.O.T. au deuxième niveau.

3.6.2. c) Troisième niveau de transduction :

On ne conserve, à ce niveau, que les phonèmes jugés les plus représentatifs. Les exemples précédents illustrent le fait que les problèmes d'ambiguité les plus importants se situent surtout au niveau des voyelles et des diphtongues ; cette troisième transduction vise donc essentiellement à isoler les consonnes jugées les plus stables dans le nom analysé. Certaines assimilations de consonnes sont également effectuées.

Les voyelles et les diphtongues sont systématiquement ignorées, sau si elles apparaissent en début de nom ; ce procédé a déjà été utili dans des systèmes de codification phonétique des noms, notamment le Russel Soundex Code pour l'anglais, et le Code Henry pour le franç [A.CLL] [A.BOP].

Exemple :

COUTAZ, COUTE, COTE donnent K.T au troisième niveau
AIMERIN, AYMARD, EYMARD donnent E!.M.R. au troisième ni
VOULAT, VILLOT, VALIAT donnent V.L au troisième niveau

+++

3.6.3. Les paramètres linguistiques :

Comme dans PIAF, les paramètres linguistiques sont constitués par trois ensembles d'informations :

- Un dictionnaire contenant les bases (chaînes de caractères) caractéristiques des phonèmes ; par exemple 'AU', 'EAU', 'O', etc... pour "O" [B.GDJ], rattachées à des modèles de transductio

- Des modèles de transduction : ils traduisent l'appartenance d'une base à une catégorie phonétique particulière ; ils permettent également de sélectionner les décompositions valides du nom en indiquant les enchaînements autorisés en cours d'analyse. Cet enchaînement est représenté par une grammaire d'états finis définissant les conditions de validation d'un modèle et, pour chacun d'eux, les modèles subséquents possibles.

- Des règles de transduction : la définition des modèles fait intervenir des validations de règles qui représentent les divers états de l'automate ; à toute règle validée correspond une opération de transduction particulière (concaténation de symboles phonétiques).

Pour définir un automate transducteur, l'utilisateur a donc à définir ces trois types de paramètres ; cela se fait de manière interactive, à l'aide de commandes pour le dictionnaire, de langages pour les modèles et les règles. Dans notre cas particulier, les trois niveaux de transduction ont été obtenus de la façon suivante :

- A chaque niveau de transduction correspond un sous-ensemble du dictionnaire, des modèles, des règles.

- Chaque niveau de transduction utilise en entrée le résultat du précédent (sauf le premier qui analyse les noms sous leur forme d'origine).

- Chaque niveau de transduction produit, pour une chaîne d'entrée, une transduction unique.

3.6.4. Modalités d'utilisation de PIAFPHO

Divers paramètres fixant la configuration d'exécution d'une session de travail sont fixés lors d'un prologue conversationnel :

- Mode d'entrée des données : elles peuvent être soit délivrées au terminal, soit être extraites d'un fichier.

- Mode de sortie des résultats : ils peuvent être affichés au terminal ou enregistrés sur un fichier.

- Fonctionnement du transducteur : la transduction peut s'effectuer en mode 'pas à pas' ou en mode 'continu'. En mode 'pas à pas', il affiche systématiquement la chaîne d'entrée ainsi que le résultat des trois transductions, et se place en attente d'une autre entrée. Ce mode permet de contrôler le fonctionnement des transductions lors de la phase de mise au point des paramètres linguistiques. En mode 'continu', les sorties sont les mêmes mais le programme ne peut rester bloqué en cas d'erreur (impossibilité de terminer l'une des trois transduction) ; il émet alors un diagnostic à la console et passe au nom suivant. Ce mode d'exploitation est surtout utilisé lorsque l'on a des fichiers en entrée et en sortie, et que le modèle linguistique est au point.

Les fonctionnements anormaux ont deux causes principales :

- existence de plusieurs transductions possibles à un niveau donné
- impossibilité de terminer l'analyse d'un nom.

Dans les deux cas, ces erreurs proviennent d'une insuffisance ou d'une incohérence dans la définition des paramètres linguistiques ; le programme émet alors les résultats éventuels, suivis de '*' signalant l'erreur. L'utilisateur a alors le choix entre plusieurs possibilités d'action en travaillant en mode 'pas à pas' :

- accès au dictionnaire : effectué par la commande $DIC ; il peut alors l'interroger et le modifier de manière conversationnelle. Les commandes correspondantes sont celles de PIAF.

- accès aux modèles et aux règles : par la commande GRAM ; il peut alors modifier ces paramètres.

- visualisation des étapes intermédiaires de la transduction : après la commande '$', toute transduction est entièrement tracée, étape par étape, ce qui permet de localiser la sous-chaîne non analysable de la donnée, et d'effectuer les corrections nécessaires dans les paramètres.

- sortir de l'environnement 'erreur' : en tapant 'retour chariot'

- accepter la solution produite (après correction des paramètres), par la commande $OUI ; le résultat est alors éventuellement enregistré sur le fichier de sortie.

- terminer la session de travail, par la commande $FIN

- L'ensemble de ces commandes, leur enchaînement possible, est donné par le diagramme de l'annexe III.a ; des exemples de sessions sont donnés en annexe III.b.

3.6.5. Restrictions d'utilisation :

- PIAFPHO a été développé à partir d'une version opérant sous CP/CMS de PIAF ; ce logiciel est lui-même écrit en PL360. Les possibilités de portabilité sont donc relativement limitées ; des développements de PIAF étant en cours en PASCAL, nous pensons pouvoir disposer à moyen terme d'une version de PIAFPHO beaucoup plus portable.

3.6.6. **Conclusion** :

- L'utilisation de PIAFPHO s'est avérée très efficace, bien que la définition des modèles linguistiques soit relativement complexe. Les possibilités d'interaction permettent une définition et une correction rapide de ces paramètres ; le seul fait de pouvoir définir simultanément trois types de transduction est en soi une preuve convaincante de la généralité de cet outil. Enfin, l'exploitation s'avère également performante puisque la transduction à trois niveaux de 5000 noms patronymiques a necessité environ 2 minutes de temps CPU sur l'IBM 360-45 alors utilisé au Centre Inter-Universitaire de Calcul de Grenoble. Ceci représente un temps moyen de 8 ms par transduction, la longueur moyenne des noms étant de 8 caractères.

4.- C O N C L U S I O N

-=-=-=-=-=-=-

L'analyse, aussi détaillée que possible, que nous avons présentée au chapitre II s'est avérée très utile pour la recherche et le développement des méthodes mises en oeuvre dans les différents logiciels proposés ; elle nous a permis de mieux cerner les multiples aspects d'un problème encore peu étudié sur le plan formel malgré les applications importantes en cours dans divers centres de recherche (cf. [A.BEA], [B.SK2], [A.BOU] et [A.BCH]). Cette étude nous a conduit à aborder des domaines très divers comme les bases de données, l'intelligence artificielle, la théorie de l'information etc... La spécificité des données traitées et des structures à produire fait que l'on est amené à définir une méthodologie particulière ; comme nous l'avons mentionné à plusieurs reprises, un objectif essentiel a été d'apporter tout d'abord des améliorations qualitatives au processus de reconstitution de la population. Notre apport au domaine se situe donc tout d'abord à la définition de notions telles que les classes de cohérence, les modèles de variation, les possibilités et les solutions de jumelage ; nous nous sommes ensuite attachés à définir des algorithmes aussi généraux que possible permettant de créer et utiliser les structures correspondantes.

Les méthodes d'évaluation proposées sont encore embryonnaires ; la théorie de l'information nous semble un point de départ très prometteur [B.DCS] pour la définition de critères de ressemblance sans doute mieux adaptés à notre problème que ceux développés en classification automatique.

Un autre aspect concerne la recherche de stratégies plus efficaces dans la sélection des solutions de jumelage (intelligence artificielle) avec la recherche de techniques plus performantes de traitement des graphes (possibilités, solutions). A plus long terme, une recherche intéressante consiste à étudier dans quelles conditions les phases de couplage et de décision pourraient être conduites en parallèle au lieu d'être exécutées en séquence comme actuellement ; comme dans d'autres types d'applications

[B.CH6], on pourrait escompter de meilleures performances globales, dues à une sélection plus rapide des solutions.

Dans son état actuel de prototype, MERCURE offre plusieurs originalités par rapport aux logiciels existants ; en plus des aspects méthodologiques évoqués plus haut, il se distingue essentiellement par son adaptabilité et son intéractivité. Sur le plan de la reconstitution automatique de population, son utilisation est actuellement limité aux données biographiques ; il reste à développer le traitement d'informations plus complexes comme le couplage de recensements ou de "feux" (household) qui constituent également une source d'informations démographique importante. Un aspect important de MERCURE est néanmoins de proposer une véritable organisation de base de données, permettant la construction et l'exploitation des structures complexes et volumineuses propres à ce type d'application. Nous avons montré dans le chapitre 3.4. les divers niveaux de représentation de la population ; si nous n'avons actuellement que peu développé l'aspect exploitation de la population reconstituée (cf. annexe 4), nous sommes convaincus que la structure choisie permet tous les types d'investigations classiques dans les applications de démographie ou de génétique (cf. [B.HAS] [B.MDS].

Les logiciels annexes de contrôle et de transduction phonétique sont également susceptibles d'améliorations sensibles ; la saisie de l'information doit tout d'abord être considérée dans un contexte conversationnel, de façon à permettre une correction immédiate des informations. On pourrait aussi aller beaucoup loin, dans le sens d'une interprétation plus automatisée des textes originaux (actes) ; bien qu'encore peu envisageables pratiquement, des travaux très intéréssants ont été développés dans ce sens par G.P. ZARRI [B.ZAR] dans le cadre du projet RESEDA. Enfin, la transduction phonétique paraît toujours une des méthodes les plus efficaces pour résoudre le problème central des variations des noms ; si les résultats fournis par PIAFPHO se sont avérés satisfaisant il reste à simplifier le processus de définition des automates transducteurs de façon à le rendre plus accessible à des utilisateurs non informaticiens.

- ANNEXE 1 -

RESUME DES NOTATIONS EMPLOYEES

Les notations sont données de manière complètement indicées, dans l'ordre de leur définition.

- $A^i = \{a^i_j\}$ — Ensemble de départ
- K^i_j — Caractéristique définie sur A^i
- $D^i = \{K^i_j\}$ — Description définie sur A^i
- $S_D i$ — Saisie de A^i relativement à D^i
- $E^i = \{e^i_j\}$ — Fichier, image de A^i par $S_D i$
- C^i_j — Caractéristique fichier définie sur E^i, déduite de $S_D i$ et K^i_j
- $V^i_j = \{v^j_k\}$ — Ensemble des valeurs de K^i_j et C^i_j
- Π^i — Ensemble produit des ensembles de valeurs des caractéristiques de D^i
- $\bar{K}^i_j$, $\bar{C}^i_j$ — Caractéristiques inverses
- A^i_{jk}, E^i_{jk} — Désignent respectivement les ensembles : $\bar{K}^i_j (v^j_k) \subseteq A^i$ et $\bar{C}^i_j (v^j_k) \subseteq E^i$
- $A = \bigcup_{i=1}^{n} A^i = \{a_j\}$
- $E = \bigcup_{i=1}^{n} E^i = \{e_j\}$
- $H_j \subseteq E$ — Classe d'enregistrements homologues

- R_{jk} relation entre C_j et C_k

- $R_{jk,r}$ coupe de R_{jk} dans V_k, pour $v_r \in V_j$

- $\mathscr{R}^{il}_{jk}$ relation induite par R_{jk} dans $E^i \times E^l$

- $\mathscr{R}^{il}_{jk,m}$ coupe de $\mathscr{R}_{jk}$ dans E^l, pour $e_m \in E^i$

- R^{il} relation de cohérence dans $\Pi^i \times \Pi^l$

- R^{il}_{α} coupe de R^{il} dans Π^l, pour $\alpha \in \Pi^i$

- $\mathscr{R}^{il}$ relation induite par R^{il} dans $E^i \times E^l$

- $\mathscr{R}^{il}_{m}$ coupe de $\mathscr{R}^{il}$ dans E^l, pour $e_m \in E^i$

- M modèle de variations

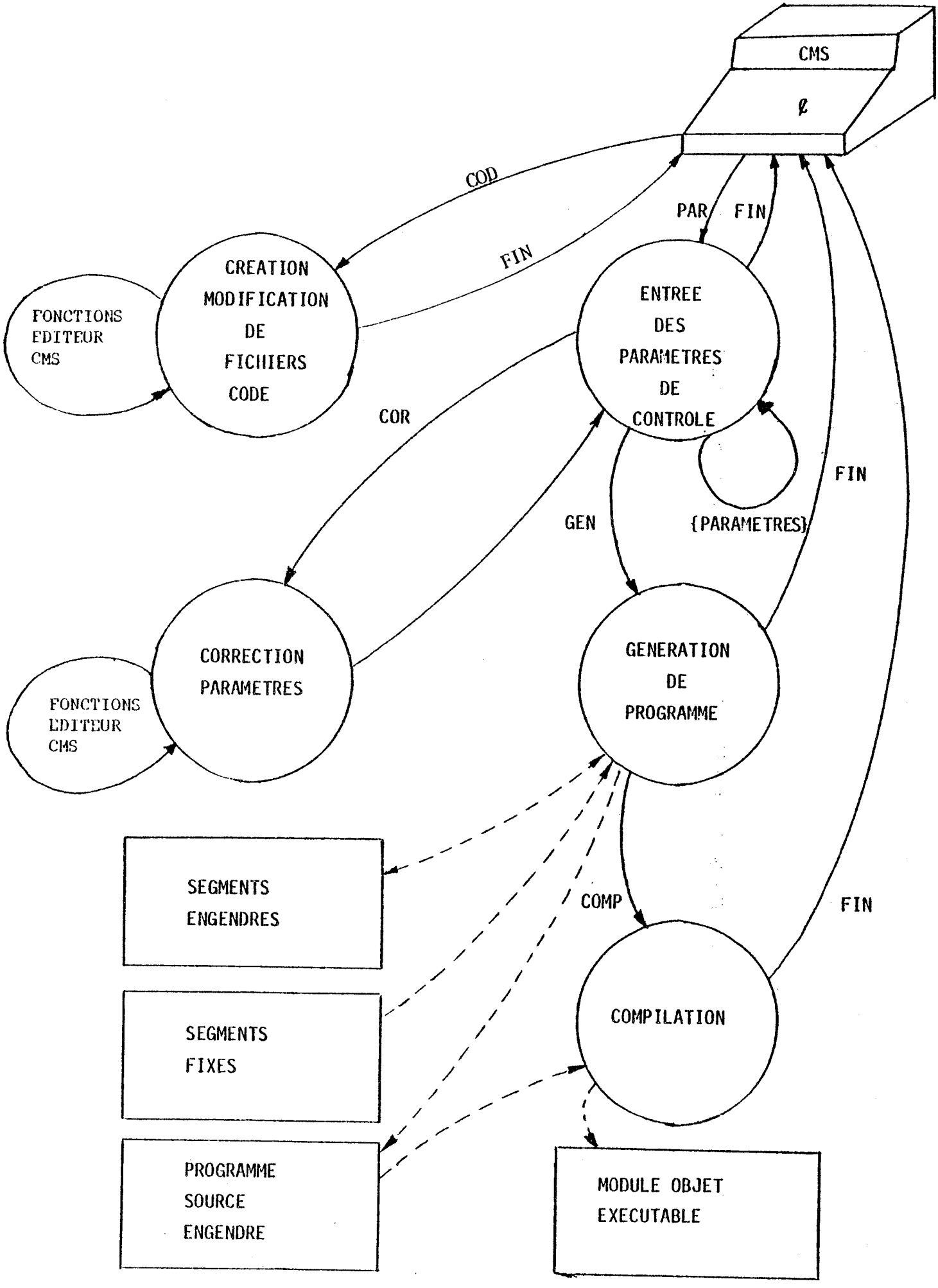
- $\mathscr{R}_{jk}$ relation de cohérence étendue

- $\mathscr{r}^{il}_{jk}$ relation induite par $\mathscr{R}_{jk}$ dans $E^i \times E^l$

- ANNEXE 2 -

EXEMPLE D'UTILISATION DU LOGICIEL DE CONTROLE

DIAGRAMME DE FONCTIONNEMENT DU GENERATEUR



Annexe 2-a.

- Entrée des paramètres de contrôle

```
gencont

DEFINITI PARAMETR OU CODES ? PAR COD

par

EXECUTION BEGINS...
DONNEES SUR FICHIER? REPONDEZ OUI OU NON
_non
STRUCTURE DU FICHIER  REPONDEZ OUI OU NON    /structure du fichier/
_oui

-personne debut
_
 cartel debut

_code_fichl constante "fp"

_el etiquette (3)

_sl sequence "1"
E
_nom mot (16)

_nom_jfille mot (16)

 prenom_ind code (3 13)

_situation_ind liste (celibataire, marie, veuf, divorce)

_bl constante " "

_sexe_ind liste (m, f)

_b2 constante " "

_profession_ind code (3 13)


 b3 constante " "
-
 date_naissance debut

_jour_n valeur (1 a 31)

_mois_n valeur (1 a12)

_annee_n valeur (199@00 a 1970)

_fin

_date_mariage debut

_jour_m idem jour_n

_mois_m idem mois_n
-
_annee_n idem annee_n
```

```
  fin
 _b4 constante "      "
 _fin
 _carte2 debut
 _code_fich2 constante "fp"
 -e2 etiquette (3)
 _s2 sequence "2"
 _adresse debut
 _nom_immeuble texte (24)
 _rue_num texte (25)
 _ville_codep texte (25)
 _fin
 _fin
 _fin
 _|
CONTROLE PRESENCE      REPONDEZ OUI OU NON          /contrôles directs de présence
_oui
 _si sexe_ind absent alors 'sexe absent a tort' fsi
 _|
CONTROLE INDIRECT      REPONDEZ OUI OU NON          /contrôles indirects/
_oui
 _si sexe_ind = 'f' et situation_ind = 'marie' alors
 _nom_jfille present sinon 'nom jeune fille absent' fsi
 _|
GROUPES                REPONDEZ OUI OU NON          /test présence groupes/
_oui
 _nom prenom_ind *
 _|
ENTREE DES DONNEES TERMINEE
```

```
*STRUCTURE CORRECTE*
GENERATI CORRECTO ? GEN COR
gen                                          /appel générateur de programmes/

EXECUTION BEGINS...
INTERCLASSEMENT TERMINE
COMPILAT ? COMP                              /interclassement/
comp
                                             /compilation/
gencont
DEFINITI PARAMETR OU CODES ? PAR COD
cod
NOM FICHIER CODE ?                           /création des fichiers code PRENOM/
sprenom
NEW FILE.
INPUT:
mar marie                                    /entrée des codifications/
jsn josianne
ple peirre
jac jacques
pal paul
and andre
mrc marcel
jpl jean paul
jlc jean luc
jyv jean yves

EDIT:
file
*ENTER SORT FIELD DEFINITIONS
1 3                                          /tri/
    10 RECORDS READ ON PASS 1.
EXECUTION BEGINS...
ENTREZ LE NOM DU FICHIER CODE
_prenom                                      /création des fichiers/
NOM DU FICHIER CODE CREE:  PRENOMI$
GENERATI CODE ? COD
cod
NOM FICHIER CODE ?
smetier                                      /création du fichier code METIER/
NEW FILE.
INPUT:
c@sec secretaire                             /entrée des codifications/
mac macon
vds vendeuse
rnt rentier
men menuisier
tol tolier
chm cheminit
for forgeron
mar marin
mdc medecin
pch pecheur

EDIT:
file
*ENTER SORT FIELD DEFINITIONS
1 3                                          /tri/
    11 RECORDS READ ON PASS 1.
```

```
EXECUTION BEGINS...
ENTREZ LE NOM DU FICHIER CODE                    /creation du fichier code métier/
 metier
NOM DU FICHIER CODE CREE:  METIERI$
GENERATI CODE? COD
fin
TERMINE
```

Annexe 2-b.

- Exemple d'entrées à contrôler

```
FPAA1DUPONT          DURAND          MARMARIE        F SEC 0102194503081965
FPAA2TOUR BELLEDONNE                 25 BLD GENERAL LECLERC   38000 GRENOBLE

FPAA 1DURAND                         PIECELIBATAIRE M MAC 2608194523121967

FPAAB1DUBOIS                         ISBMARIE        F VDS 1212193832051960
FPAAB2LE LESSEPS                     35 BLD DES ARENES        06000 NICE

FPAAC1DIEUDONNE                      MRSDIVORCES     : RNT 0112189802041930
FPAAC2LES CHARMILLES                 AV DE TOULON             13013 MARSEILLE

FPAAD1LESAGE         DUPOND          JYVCELIBATAIRE M CHM 0308194508071965
FPAAD2                               35 RUE DES PETITS PERES  75011 PARIS

FPAAE1LESAGE         DUPOND          MARMARIE        F VDS 1312194512121968
FPAAE2VILLA MON REVE                 17 AVENUE DES RESISTANTS 38000 GRENOBLE
```

Exécution du programme engendré

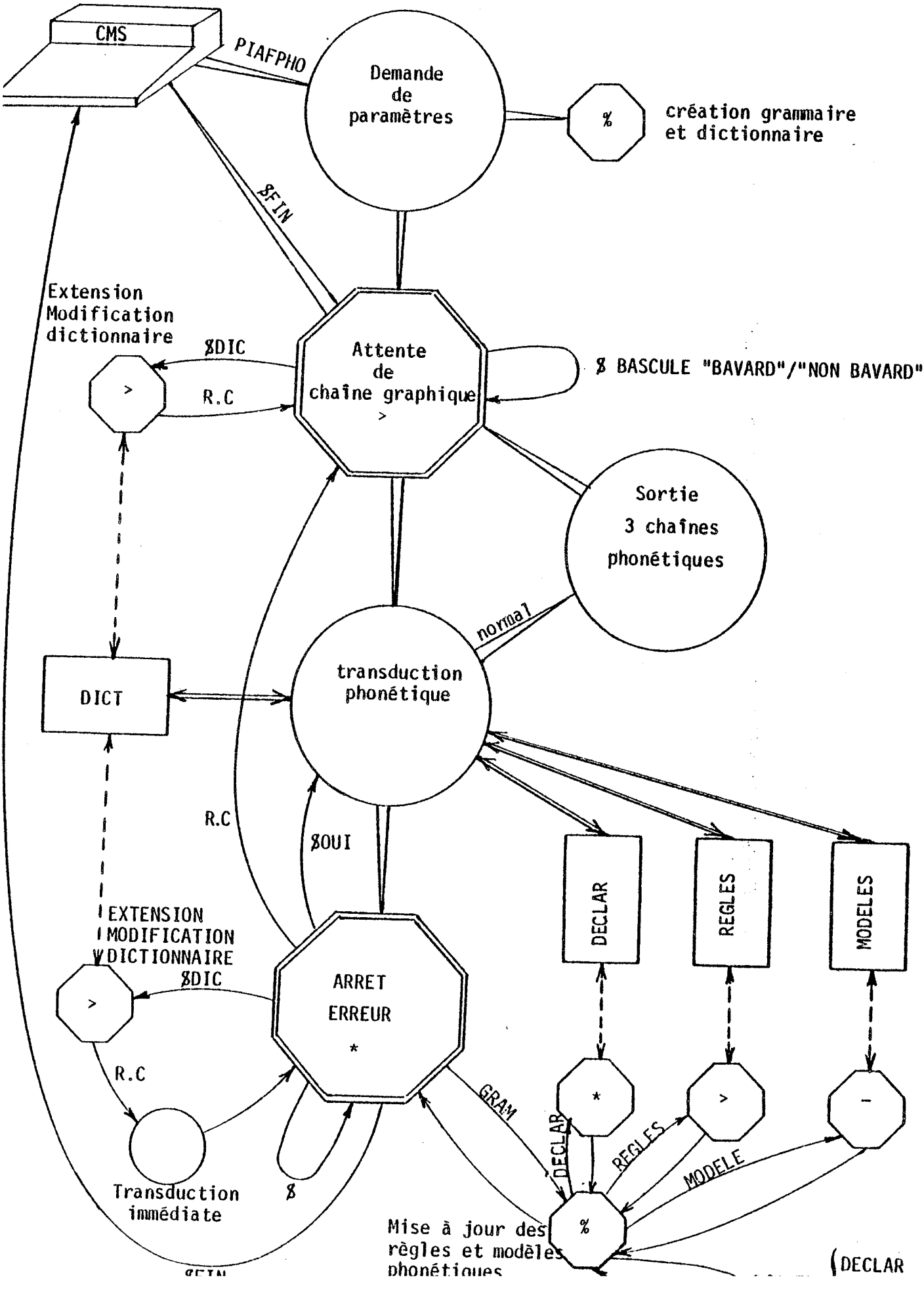
```
controle

EXECUTION BEGINS...
EDITION DES TAUX DE PRESENCE SUR LE FICHIER SAIN ? REPONDEZ OUI OU NON
_non
REPRISE APRES INCIDENT ? REPONDEZ OUI OU NON      /départ en début de fichier/
_non
CORRECT
FPAA 1DURAND                                  PIECELIBATAIRE M MAC 2608194523121967
CARTE NON EN SEQUENCE: REJETEE
**********
FPAAB1DUBOIS                                  ISBMARIE        F VDS 1212193832051960
FPAAB2LE LESSEPS                 35 BLD DES ARENES            06000 NICE
DIAGNOSTICS
PRENOM_IND          ISB                       CODE INCONNU DANS  PRENOM
JOUR_M              32                        VALEUR NON DANS LES BORNES
                                              NNOM JEUNE FILLE ABSENT
ERREUR
**********
FPAAC1DIEUDONNE                               MRSDIVORCES     RNT 0112189802041930
FPAAC2LES CHARMILLES             AV DE TOULON                 13013 MARSEILLE
DIAGNOSTICS

PRENOM_IND          MRS                       SEXE ABSENT A TORT
SITUATION_IND       DIVORCES                  CODE INCONNU DANS  PRENOM
ANNEE_N             1898                      ELEMENT INCONNU DANS LA LISTE
ERREUR                                        VALEUR NON DANS LES BORNES
**********
CORRECT
CORRECT

NOMBRE DE CARTES LUES                 /résultats globaux du contrôle/
00011
NOMBRE DE CARTES REJETEES CAR EN DESORDRE
00001
NOMBRE D'ENREGISTREMENTS LOGIQUES ANALYSES
00005
NOMBRE D ENREGISTREMENTS LOGIQUES CORRECTS
00003
NOMBRE D ENREGISTREMENTS LOGIQUES ERRONNES
00002
TAUX D ENREGISTREMENTS LOGIQUES ERRONNES
00.4000
```

- ANNEXE 3 -

EXEMPLES D'UTILISATION DU LOGICIEL

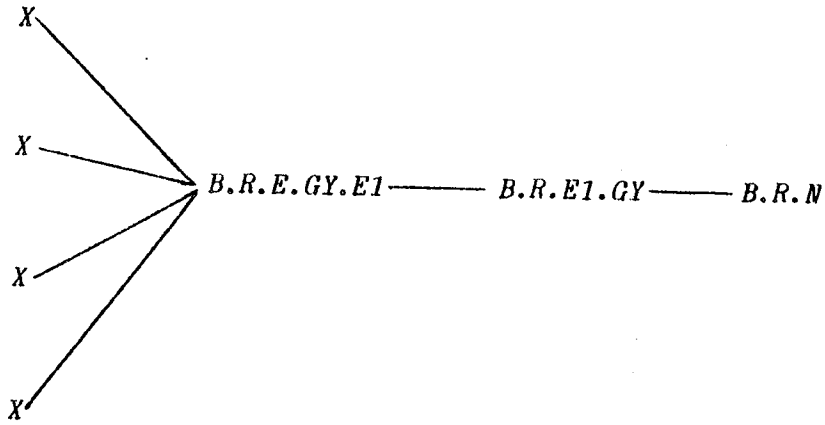
"PIAFPHO"

DIAGRAMME DE FONCTIONNEMENT DE PIAFPHO



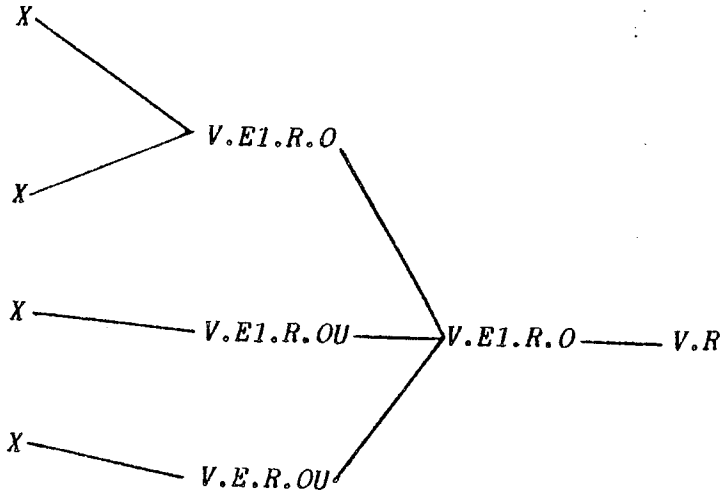
```
piafpho


** PIAFPHO A VOTRE SERVICE. **
* EQUIPE INTEL. ARTIF. *
- LABO INFORMAT U.S.M.G. - B.P.53 - 38041 GRENOBLE CEDEX -
EXECUTION BEGINS...

DONNEES ? ( T OU NOM DE FICHIER )
> t                                          /choix du support d'entree/

RESULTATS ? ( T OU IMP OU NOM DE FICHIER )
> t                                          /choix du support de sortie/

MODE ? ( ARRET ERREUR OU CONTINU )
> arret                                      /choix du mode de fonctionneme

> beauchamp                                  /entrée d'un nom/

/niveau1/  1   B    0    CH   AN   P
/niveau2/  2   B    0    CH   2N   P
/niveau3/  3   B    CH   P
>
  $dic                                  /accés au dictionnaire/

> ?/p/                                  /interrogation de la base 'p'/

/PH(*)/F/      /PP(*)/P/
/P(*)/P2/
/P/PMU/
/P/P1/

> /p (*)/gtmu/
>                                       /introduction de la base 'p ' liée au modèle GTMU
                                        (terminales muettes)/

> beauchamp                             /réintroduction de  la donnée/

  1   B    0    CH   AN
  2   B    0    CH   2N
  3   B    CH
>
  beauchant                             /essais de variantes orthographiques/
  1   B    0    CH   AN
  2   B    0    CH   2N
  3   B    CH
> bauchamp
  1   B    0    CH   AN
  2   B    0    CH   2N
  3   B    CH
>
```

brenier
1 B R E GY E1
2 B R E1 GY
3 B R N
bregnier
1 B R E GY E1
2 B R E1 GY
3 B R N
bregner
1 B R E GY E1
2 B R E1 GY
3 B R N
brennier
1 B R E GY E1
2 B R E1 GY
3 B R N

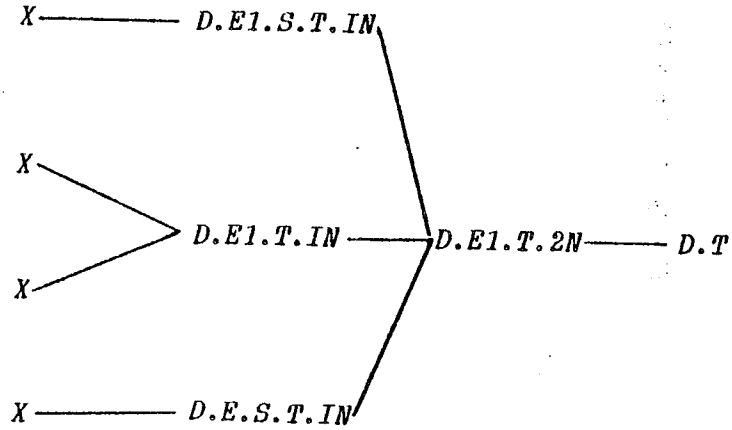


verrot
1 V E1 R O
2 V E1 R O
3 V R

valrot
1 V E1 R O
2 V E1 R O
3 V R
verroud
1 V E1 R OU
2 V E1 R O
3 V R
veroud
1 V E R OU
2 V E1 R O
3 V R



d estaing
1 D E1 S T IN
2 DE1 T 2N
3 D T
destaing
1 DE1 T IN
2 DE1 T 2N
3 D T
des taig@ng
1 DE1 T IN
2 DE1 T 2N
3 D T
de staing
1 DE S T IN
2 DE1 T 2N
3 D T



- ANNEXE 4 -

EXEMPLES D'UTILISATION DU LOGICIEL

"MERCURE"

Interrogation des paramètres logiques
d'une application

```
mercure essai ?              / accès à l'application de nom "essai" /
LE SYSTEME MERCURE-3 EST A VOTRE DISPOSITION
COMMANDE?

QUESTION:
par ?                        / accès à l'environnement "paramètres" /

ENVIRONNEMENT DE TRAVAIL: PARAMETRES
QUESTION?
i-apl ?          / commande de listage des paramètres de l'application /
PARAMETRES APPLICATION: ESSAI

TYPE ACTEUR: PERSONNE              / Un seul type d'acteurs: "personne" /
   PERMANENTS:  33
   CLASSES:  0
   RESTANTS:  33
   TAUX DE COUPLAGE:  0
     TYPE EVENEMENT: NAI                    DE PERSONNE   /type évènement "naissance"/
         PERMANENTS:  12
         RESTANTS:  12
         TAUX DE COUPLAGE:  0
     TYPE EVENEMENT: MAR                    DE PERSONNE   /type évènement "mariage"/
         PERMANENTS:  9
         RESTANTS:  9
         TAUX DE COUPLAGE:  0
     TYPE EVENEMENT: DEC                    DE PERSONNE   /type évènement "décès"/
         PERMANENTS:  12
         RESTANTS:  12
         TAUX DE COUPLAGE: 0
```

Affectation - Interrogation des paramètres "données" de relations élémentaires

```
par ?

ENVIRONNEMENT DE TRAVAIL: PARAMETRES

a-rel r1 ?          / activation de la relation de nom r1 /

QUESTION:

mes-g r1 = 10 ?     / affectation d'un poids global égal à 10, à r1 /

QUESTION:

i-rel r1 ?          / interrogation des paramètres "données" de r1 /

  RELATION: R1
   ETAT: ACTIVE
   MESURE GLOBALE POIDS:  10

QUESTION:

intervalle r1 entre 18 et 60 ?   / affectation d'un intervalle d'évaluation
QUESTION:                          pour r1 /

i-rel r1 ?
  RELATION: R1
   ETAT: ACTIVE
   INTERVALLE DE COMPARAISON: 0      A  90
   MESURE GLOBALE POIDS:  10

QUESTION:

n-rel-r1 ?     / neutralisation de la relation r1/

QUESTION:

i-rel r1 ?

  RELATION: R1
   ETAT: NEUTRALISEE
   INTERVALLE DE COMPARAISON: 0      A  90
   MESURE GLOBALE POIDS:  10
```

Exemple de construction de classe de cohérence

```
cou ?        / accès à l'environnement de couplage /
ENVIRONNEMENT DE TRAVAIL: COUPLAGE
QUESTION:

d-classe nbb de personne ?   / construction de la classe de l'acteur "nbb" /
CLASSE: 1 NOMBRE D ELEMENTS: 7   / la classe comporte 7 personnes /

QUESTION:

i-classe 1 de personne   mode g seuil 0 ?  / listage de la classe mode "graphe" /
**  CLASSE : 1   **

                  * DEC                          * MAR                  *
                  * DCC    | DDD                 * MCC    | MDD         *
NAI       *
          *  NBB  *  111   |                     *  111   |  101        *
          *  NEE  *  111   |  111                *         |  101        *
          *  NDD  *         |                     *  110   |             *
MAR       *
          *  MCC  *  111   |                     *
          *  MDD  *         |  101                *

QUESTION:

i-classe 1 de personne mode l seuil 0 ?  /listage de la classe mode "liste" /

**  CLASSE : 1   **

NAI                          MAR                          DEC
  ACTEUR: NBB                  ACTEUR: MCC                  ACTEUR: DCC
    MCC   111                    NBB   111                    NBB   111
    MDD   101                    NDD   110                    NEE   111
    DCC   111                    DCC   111                    MCC   111
 AACTEUR: NEE                  ACTEUR: MDD                  ACTEUR: DDD
    MDD   101                    NBB   101                    NEE   111
    DDD   111                    NEE   101                    MDD   101
    DCC   111                    DDD   101
  ACTEUR: NDD
    MCC   110

QUESTION:
s-classe 1 de personne ?  / suppression de la classe de cohérence /
```

Exemple de familles reconstituées

```
famille ?

NOM EPOUX ?
durand          / identification du couple par les caractéristiques
                  nominatives; l'identification par le code du couple
PRENOM EPOUX ?    est possible /
marcel
NOM EPOUSE ?
dupin
PRENOM EPOUSE ?
marie
DATE ?
1602

****FAMILLE****
    * DATE * 1602
    * PERE * DURAND          MARCEL      1580 1602 1660
    * MERE * DUPIN           MARIE       1585 1602 1670
  ** ENFANTS **
          DURAND             URSULE      1607 1625 1660
          DURAND MARCEL      MARCEL      1603 1627 1650
          DURAND             MARCELLE    1605      1610
```

Exemple de biographie reconstituée

```
biographie ?

NOM ?
*       / identification de la personne par son code /
CODE ?
nch

*** DURAND        MARCEL
  * MARIAGE *     1602  GRENOBLE
        DUPIN           MARIE
  * NAISSANCE     1580  GRENOBLE
  * DECES         1660  GRENOBLE
```

Exemple de généalogie descendante

```
genealogie ?

GENEALOGIE ASCENDANTE OU DESCENDANTE ?.....
descendante
NOM ?                    / identification de l'origine de la généalogie par les
dupont                     caractéristiques nominatives de la personne; il est
PRENOM ?                   possible de l'indiquer à l'aide de son code acteur, s'il
jean                       est connu de l'utilisateur /
PRENOM PERE ?
jean
NOM MERE ?
durand
PRENOM MERE ?
ursule

** ANCETRE **
DUPONT          JEAN      (1575 1600 1650)
        * ENFANTS *
        ** ANCETRE **
        DUPONT          PAUL      (1604 1625 1660)
                * ENFANTS *
                ** ANCETRE **
                DUPONT          PAUL      (1627 1650 1685)
                        **ENFANTS**
                        DUPONT          LEA       (1651    1720)
                        DUPONT          PAUL      (1653    1730)
                DUPONT          MARIE     (1631    1635)
                DUPONT          PIERRE    (1629    1670)
        DUPONT          JEANNE    (1601    1605)
        DUPONT          PIERRE    (1606    1650)
```

NOTA: les éléments d'une génération ne sont pas triés par ordre chronologique de naissance.

Exemple de généalogie ascendante

```
genealogie ?

GENEALOGIE ASCENDANTE OU DESCENDANTE ?.....
ascendante
NOM ?
*            / identification de l'origine de la généalogie par le code
CODE ?         de l'acteur correspondant /
ncc

DUPONT          PIERRE    (1629      1670)
        * PERE *
        DUPONT            PAUL      (1604 1625 1660)
                * PERE *
                DUPONT              JEAN      (1575 1600 1650)
                        * PERE *
                        DUPONT      JEAN
                        * MERE *
                        DUFOUR      ANNE
                * MERE *
                DUBOIS              LEONTINE (1580 1600 1630)
                        * PERE *
                        DUBOIS      ANDRE
                        * MERE *
                        DURAND      PAULE
        * MERE *
        DURAND          URSULE    (1607 1625 1660)
                * PERE *
                DURAND              MARCEL    (1580 1602 1660)
                        * PERE *
                        DURAND      EMILE
                        * MERE *
                        DUBOIS      HELENE
                * MERE *
                DUPIN               MARIE     (1585 1602 1670)
                        * PERE *
                        DUPIN       PAUL
                        * MERE *
                        DURAND      MADELEINE
```

BIBLIOGRAPHIE

La liste des références a été scindée en deux parties :

Bibliographie A

Elle réunit l'ensemble des références relatives aux expériences d'application dans le domaine de la démographie historique, mentionnées au chapitre I.

Bibliographie B

Elle réunit l'ensemble des références aux travaux mentionnés dans les chapitres II, III et IV.

BIBLIOGRAPHIE A

Références aux applications en démographie historique

[A.BEA] Lee L. BEAN, M. SKOLNICK et al., "The Mormon historical demography project, Structure and data evaluation"
Conference on "Method of Automatic Family Reconstitution", Florence 7

[A.BCH] BEAUCHAMP, P., CHARBONNEAU H; et al., "La reconstitution automatique des familles un fait acquis".
Revue Population, Numéro spécial Mars 1977.
voir aussi

BEAUCHAMP P., RAY R., LEGARE I., "Reconstitution automatique des familles par le programme "Hochelaga II".
Population et Familles 33(3) p. 1-40, 1977.

[A.BOC] M. BORNAREL, Y. CHIARAMELLA, "Un programme interactif de reconstitution de populations : MERCURE. Application à la paroisse St Laurent de Grenoble au 18è siècle".
Conférence on "Methods of Automatic Family, Reconstitution"
Florence 1977.

[A.BOP] G. BOUCHARD, S. POUYEZ, "Name variations and computerized record linkage".
Historical Methods, mai 1980

[A.BOU] G. BOUCHARD, P.BRARD, "Le programme de reconstitution des familles saguenayennes : données de base et résultas provisoires".
Histoire sociale/Social History, Vol. XII, N°23, pp. 170-185, 1979
voir aussi :

G. BOUCHARD, C. POUYEZ, R.ROY, "L'avenir des fichiers de populations dans les sciences humaines : le projet de fichier-réseau de la population saguenayenne".
Etudes Canadiennes 1980.

[A.CHA] A. CHAMOUX, "La reconstitution des familles. Espoirs et réalités" Annales économies-sociétés-civilisations, Armand Collin, 1972.

[A.CLL] H. CHARBONNEAU, I. LEGARE, Y. LAVOIE, "Etat civil et recensements canadiens",
Annales de démographie historique, Annexes III et IV, 1972.
voir aussi :

L. HENRY, "Projet de transcription phonétique des noms de famille", Annales de démographie historique, 1976.

[A.DAP] Y. DAUBEZE, J.C. PERROT, "Un programme d'études démographiques sur ordinateur",
Annales économies-sociétés-civilisations, Armand Collin, 1972.

[A.DUD] S. DUPAQUIER, M. DEMONET, "Ce qui fait les familles nombreuses", Annales économies-sociétés-civilisations, Armand Collin, 1972.

[A.FLH] M. FLEURY, L.HENRY, "Nouveau manuel de dépouillement et d'exploitation de l'état civil ancien",
Editions de l'Institut National d'Etudes démographiques, 1965.

[A.SCH] R.S. SCHOFIELD, "La reconstitution des familles par ordinateur", Annales économies-sociétés-civilisations, Armand Collin, 1972.

[A.SKO] M. SKOLNICK et al., "Automatic reconstitution of families from parish registers : Parma Valley",
Conference on"Methods of Automatic Family Reconsitution", Florence, 1977.
voir aussi :

M. SKOLNICK, M. CAVALLI-SFORZA et al., "A preliminary analysis of the genealogy of Parma Valley, Italy",
Journal of Human Evolution, 5, pp. 95-115, 1976.

BIBLIOGRAPHIE B

Références aux travaux en Informatique et Mathématiques

[B.ABL] M. ASTRAHAN, M.W. BLASGEN et al., "System R: Relational Approach to database Management",
ACM Transactions on Database System, Vol.1, N°2, pp.97-137, 1976
voir aussi :

D.D. CHAMBERLIN, M.M. ASTRAHAN et al., "SEQUEL 2: A unified approach to Data definition, manipulation and control",
IBM Journal of Research and Development, pp.560-575, nov. 1976.

[B.ABR] J.R.ABRIAL, J.P COHEN et al., "Projet Socrate, Specifications général
Université Scientifique et Médicale de Grenoble, 1972.

[B.ACD] J. ACZEL, R. DAROCZY, "On measures of information and their characterizations",
Mathematics in Science and Engineering, Academic Press, 1975.

[B.ALG] Groupe ALGOL de l'AFCET, "Definition du langage algorithmique ALGOL 68",
Editions Hermann, 1972.

[B.CAV] J.M. CAGNAT, F. VEILLON, "Cours de programmation en langage PL/1",
Librairie Armand Collin, Collection V, 1972.

[B.CH1] Y. CHIARAMELLA, "Evaluation du pouvoir discriminant d'un ensemble d'informations. Applications aux bases de données",
Congrès AFCET Informatique, nov. 1980.

[B.CH2] Y. CHIARAMELLA, "A general demagraphic Data Base System MERCURE"
Second International Conference on Data Bases in Humanities and Social Sciences, Madrid, 1980.

[B.CH3] Y. CHIARAMELLA, "Towards an interactive use of computers in Human Sciences. Two examples of adaptable and interactive programs", 15th International Congress on Medieval Studies, Kalamazoo, 1980.

[B.CH4] Y. CHIARAMELLA, "Un programme interactif de reconstitution de population à partir d'informations floues : MERCURE", 3rd International Congress on Computing at the Humanities , Waterloo, 1977.

[B.CH5] Y. CHIARAMELLA, "Détection automatique des variations orthographiques sur des noms propres. Definition d'un transducteur morpho-phonétique interactif", 3ème Congrès International de linguistique computationnelle, Ottawa, 1976.

[B.CH6] Y. CHIARAMELLA, J. COURTIN, E. GRANDJEAN, G. VEILLON, "Utilisation des techniques de recherche en parallèle, au contrôle de données ambigües, Applications aux sciences humaines", Congrès AFCET, Panorama de la nouveauté informatique en France, 1976.

[B.CII] CII Honeywell Bull, "Reference manual for the ADA programming language", Rapport de définition du langage, 1980.

[B.COD] E.F. CODD, "Further normalization of the data base relational model", IBM Research RJ909, 1971.

[B.COU] J. COURTIN, "Algorithmes pour le traitement ineractif des langues naturelles", Thèse de Doctorat es Sciences Mathématiques de l'Université Scientifique et Médicale de Grenoble, 1976.

[B.COV] J. COURTIN, J. VOIRON, "Introduction à l'algorithmique et aux structures de données", Département informatique de l'IUT B de Grenoble, Cours, 1975.

[B.DAT] C.J. DATE, "An introduction to Database Systems",
Addison Wesley, 1977.

[B.DCS] A. DUCHAUSSOY, "Résultats récents en théorie de l'information, application à l'analyse structurale",
Revue BUSEFAL, Langages et Systèmes Informatiques, N°4, 1980.

[B.DUC] P. DUCIMETIERE, "Les méthodes de la classification numérique",
Revue de statistique appliquée, Vol.18, N°4, 1970.

[B.ENG] R. ENGMANN, "Set-theoretic concepts in data structures and data base",
Technishe Hogeschool twente, Memorandum N°111, 1976.

[B.GAR] J. CARY AUGUSTSON, J. MINKER, "An analysis of some graph theoretical cluster techniques",
JACM, Vol.17, N°4, pp.571-588,1970.

[B.GDJ] E.GRANDJEAN, "Conception et réalisation d'un dictionnaire pour un analyseur interactif de langues naturelles",
Mémoire CNAM, Université de Grenoble, 1975.

[B.HAS] S. HASSTEDT, P.CARTWRIGHT, "PAP Pedigree Analysis Package",
Technical Report N°13, University of UTAH, 1979.

[B.HOA] C.A.R. HOARE, "Recursive data structures",
Advanced Research projects Agency, National Science Foundation, 1973.

[B.KES] K. KENNEDY, J. SCHWARTZ, "An introduction to the set theoretical language SETL",
Computing and Mathematics with applications, Vol.1, pp.97-119, Pergamson Press, 1975.

[B.KOS] C.H.A. KOSTER, "on infinite modes"
ALGOL Bulletin 30 in SIGPLAN Notices, Vol.4, N°3, 1969.

[B.LER] J.C. LERMAN, "Les bases de la documentation automatique"
Gauthier-Villars, Collection "Programmation"

[B.MDS] T. MANESS , S. DINTELMAN, M. SKOLNICK, "Automatic program generation for processing a high level relational like language",
Proc. ACM, pp.62-68, 1979.

[B.NIL] NILS J. NILSSON, "Probelm solving methods in Artificial Intelligence",
Mac Graw-Hill Book Company, 1971.

[B.PAS] K. JENSEN, N. WIRTH, "PASCAL - User manual and report"
Second Edition, Ed. Springler-Verlag, 1978.

[B.PIC] C. PICARD, "Theorie des questionnaires",
Gauthiers-Villars, 1965.

[B.ROZ] B.K. ROZEN, C.H. LEWIS, "Recursively defined data types",
IBM Research , 1974

[B.TER] M. TERRENOIRE, "Un modèle mathématique de processus d'interrogation : les pseudo-questionnaires",
Thèse de Doctorat es Sciences Mathématiques de l'Université Scientifique et Médicale de Grenoble, 1970.

[B.SK1] M. SKOLNICK, "The constitution and analysis of genealogies from parish registers with a case study of Parma Valley, Italy",
PhD Dissertation, Stanford University, 1974

[B.SK2] M. SKOLNICK, V. ARBON, "Genealogical data base for the automatic reconstitution of family",
Conference on "Methods of Automatic Family reconstitution", Florence, 1977.
voir aussi :

M.SKOLNICK, S. DINTELMAN et al., "A computerized family history data base system",
Sociology an Social Research, 63, pp.506-623, 1979.

[B.THE] A. THESEN, "Computer Methods in Operational Research", Academic press, 1978.

[B.ZAR] G.P. ZARRI, "Rapport sur les recherches effectuées du 1er octobre 78 au 1er avril 1979. Projet RESEDA/1", Rapport d'activité CNRS-LISH, 1979.