Université Joseph Fourier – Grenoble 1

THÈSE

par

Petko Valtchev

pour l'obtention du

Doctorat de l'Université Joseph Fourier – Grenoble 1

(Spécialité informatique)

Construction automatique de taxonomies pour l'aide à la représentation de connaissances par objets

présentée et soutenue publiquement le 15 Décembre 1999

Composition du jury

Rapporteurs :	Yves KODRATOFF Amedeo NAPOLI
Examinateurs :	Yves CHIARAMELLA Bernard MONJARDET Laurent TRILLING
Directeur :	Jérôme EUZENAT

Thèse préparée au sein
de l'INRIA Rhône-Alpes

Mis en page avec la classe TheseCRIN.

Je tiens à remercier Monsieur Yves Kodratoff, Directeur de Recherche au CNRS, d'avoir accepté de juger ce travail et de m'avoir aidé à améliorer ce mémoire par ses remarques et conseils. Je remercie vivement Monsieur Amedeo Napoli, Chargé de Recherche au CNRS, d'avoir bien voulu rapporter sur mon travail et d'avoir porté un regard si approfondi sur le manuscrit. Je remercie tous les deux d'avoir accepté des délais aussi serrés.

Je tiens à remercier particulièrement Monsieur Bernard Montjardet, Professeur à l'Université Paris 1, d'avoir bien voulu s'intéresser à mon travail.

Je remercie Monsieur Yves Chiaramella, Professeur à l'Université Joseph Fourier, et Monsieur Laurent Trilling, Professeur à l'Université Joseph Fourier, d'avoir accepté de participer à ce jury.

Je remercie Monsieur Jérôme Euzenat, Chargé de Recherche à l'INRIA Rhône-Alpes, de m'avoir toujours soutenu dans mon travail, d'avoir montré autant de patience et de m'avoir accordé, dès le premier jour de notre collaboration, sa confiance. Je le remercie pour sa disponibilité pendant la longue période de rédaction de ce manuscrit.

Je voudrais remercier Monsieur François Rechenmann, Directeur de Recherche à l'INRIA Rhône-Alpes, pour l'accueil au sein du projet Sherpa, mais aussi pour son soutien moral durant toutes ces années.

Je remercie les membres, anciens et actuels, de l'équipe Sherpa qui ont apporté leur contribution à l'aboutissement de ce travail :

Gilles Bisson pour nos nombreuses discussions et pour ses conseils qui m'ont aidé face à bien de problèmes, souvent fortement «circulaires».

Cécile Capponi pour nos échanges scientifiques si précieux et pour son encouragement dans les moments les plus difficiles.

Jérôme Gensel pour sa disponibilité et pour ses conseils d'ancien thésard qui m'ont accompagné au long de ce chemin.

Jean-Marc Gabriel pour sa joie et ses plaisanteries qui ont tant égayé le quotidien au labo.

Patrice Uvietta pour sa patience dans nos conversations, pour son humour et pour son ouverture qui m'ont redonné la confiance quand je n'en avais plus.

Hidde de Jong pour le regard critique, mais toujours amical, qu'il a porté sur mes travaux.

Danielle Ziébelin pour m'avoir présenté aux membres de l'équipe et pour son encouragement durant mes premières années de jeune chercheur.

Les autres membres de l'équipe Sherpa pour leur bonne humeur et leur patience dans notre cohabitation : Philippe, Michel, Daniel, Pierre, Olivier, Nina, Nathalie, Florence, Jean-Yves, Ysabelle, Sueli, Françoise. Un grand merci à Thibault et à Vincent pour la bonne ambiance du B216.

Des remerciements particuliers aux lecteurs de ma thèse – Jérôme, Gilles, Patrice, Moës.

Un grand merci à tous les amis qui m'ont soutenu, de près ou à distance, pendant toute la période de la thèse : Momo, Jean-Yves et Claudine, Thierry, Juan-Carlos, Irena, Zdravko, Christo, Gennady.

Merci à ma famille d'être restée si près de moi, malgré les distances. Enfin et surtout, je remercie Mirela, pour tout.

Table des matières

Introduction

1 Cadre de l'étude

Notre étude se situe dans le cadre de la structuration de bases de connaissances à objets et de ce fait elle évolue au carrefour de la représentation de connaissances à objets et de l'extraction de connaissances à partir de données.

Les systèmes de représentation de connaissances (SRC) permettent de décrire et stocker les connaissances concernant un domaine particulier, tel que le calcul scientifique ou le diagnostic médical, au sein d'une base de connaissances (BC). Une BC est souvent utilisée à des fins de résolution de problèmes, mais peut également servir de structure d'accueil pour la totalité des connaissances sur le domaine concerné ou un sous-domaine particulier. Les SRC offrent deux types de fonctionnalité :

- des moyens d'expression des connaissances à inclure dans une BC, dont le composant principal est un langage de description des connaissances ayant pour rôle de favoriser la représentation structurée et fidèle de ces connaissances,
- un ensemble de mécanismes de manipulation des connaissances qui assurent leur complétion et leur modification ; un type particulier de manipulation, les *inférences*, permet de produire de nouvelles connaissances à partir des celles déjà existantes dans la base.

Les SRC à objets (SRCO) organisent les connaissances exprimées au sein d'entités structurées, les *objets* et les *classes*. Les *objets structurés* représentent les éléments identifiables du domaine modélisé, ils sont regroupés dans des classes selon un partage de caractéristiques. Les objets et les classes possèdent une identité et sont décrits par des listes d'attributs typés et valués. Ils peuvent donc être vus comme des tuples dont les attributs sont des éléments. Les types des attributs sont soit des types de données auquel cas un large éventail, potentiellement extensible, de types est utilisable, soit des familles d'objets. Cette dernière possibilité permet de former des descriptions de complexité structurelle potentiellement illimitée. Une dimension orthogonale de la structuration des connaissances est ajoutée par l'emploi des attributs multivalués. Ainsi, la valeur d'un attribut n'est plus une seule entité, mais une collection d'entités.

Les classes d'une base de connaissances sont organisées dans une ou plusieurs *taxonomies*, par la relation de *spécialisation*, qui définit un ordre partiel. La spécialisation respecte l'inclusion entre ensembles d'objets et rend possible une factorisation des propriétés des classes qui est exploitée par le mécanisme d'*héritage*.

Les SRCO offrent un éventail de mécanismes de manipulation dont l'inférence de valeurs au sein des objets et le positionnement d'un nouvel objet dans la taxonomie (*classification*).

Dans notre étude, nous avons repris comme système de référence le SRCO Troeps qui offre une vision plus organisée du modèle d'objets. Ainsi, les objets d'une base sont divisés en *concepts*, qui représentent des familles disjointes d'individus. Les concepts, à l'instar des tables des bases de données relationnelles, définissent la structure des objets, c'est-à-dire l'ensemble d'attributs, et leur identité. Les concepts peuvent être abordés à travers un ou plusieurs *points de vue*, décrivant des perspectives particulières sur les objets du concept et comportant une taxonomie de classes chacun. En outre, le système impose un typage strict pour les attributs.

2 La problématique

Les SRCO offrent des moyens puissants de description et d'exploitation de connaissances qui reposent, pour la plupart, sur l'organisation des objets au sein d'une taxonomie de classes (spécialisation, héritage, classification). Le rôle de la taxonomie est donc primordial au sein d'une base de connaissances à objets : de sa cohérence dépend le bon fonctionnement de toute l'application. L'importance de la taxonomie de classes motive notre intérêt pour le processus de sa construction, qui constitue une étape clé de la conception d'une base de connaissances.

2.1 Conception d'une taxonomie de classes

Le processus de conception d'une taxonomie se déroule sur toute la durée de la constitution de la base de connaissances : au départ, une taxonomie satisfaisante est construite, puis celle-ci est modifiée en fonction des résultats de son utilisation [Masini *et al.*, 1989]. La construction de la taxonomie initiale suppose une identification préalable des individus du domaine ainsi que leur répartition en catégories.

Il n'existe cependant aucune méthodologie universelle que l'on pourrait suivre pour mener à bien cette tâche : la construction de la taxonomie est reconnue comme étant parmi les difficultés majeures de la conception par objets, aussi bien en RC qu'en programmation [Haton *et al.*, 1991]. En effet, l'ensemble des classes est censé refléter des facteurs subjectifs [Napoli, 1992] tels que la vision particulière du concepteur sur le domaine (degré de maîtrise du domaine, expérience), le but poursuivi par la conception, etc.

Cependant, il arrive que le concepteur ne soit pas en mesure de spécifier une organisation des objets en classes. Cette situation se produit lorsque la base de connaissances porte sur un domaine en cours d'étude et que le but de sa construction n'est pas d'épauler un système de résolution de problèmes, mais de mieux organiser la connaissance actuelle sur le domaine.

2.2 Besoin d'aide à la conception

Suivre les méthodes actuelles de conception à objets (voir par exemple [Booch, 1994]) n'apporte que peu dans la situation : les quelques conseils généraux qu'elles donnent conduisent à l'identification des classes très larges, qui seront, avec une grande probabilité, déjà présentes dans la base. Quant aux classes plus spécifiques, leur identification est laissée au concepteur (la conception de classes est un processus manuel d'après [Haton *et al.*, 1991]).

En l'absence d'une idée précise sur la structure de ces classes, le concepteur a donc le choix entre :

- la définition de classes arbitraires qu'il fera ensuite évoluer au cours de la construction et de l'utilisation de la taxonomie,
- l'inspection de l'ensemble des objets disponibles afin de détecter les emplacements possibles des classes.

La seconde option consiste à examiner plusieurs regroupements possibles des objets en classes. Dans cette éventualité, nous considérons que le SRCO peut offrir une assistance au concepteur en se chargeant de la recherche des regroupements plausibles des objets.

Cette observation est à la base de notre étude dont le but est de concevoir *une méthode générale d'aide à la construction de taxonomies dans les SRCO*. Dans la suite nous dégageons quelques caractéristiques qu'une telle méthode doit posséder.

3 Objectifs

La conception d'une taxonomie est un processus d'extraction de connaissances structurelles à partir d'un ensemble de connaissances factuelles décrivant des éléments individuels du domaine. Le processus se déroule dans le contexte de la construction d'une base de connaissances.

L'objectif de notre travail est de fournir une aide à la conception des taxonomies dans le plus grand nombre de situations. Parmi les facteurs dont il est nécessaire de tenir compte citons donc :

Prépondérance des objectifs du concepteur il ne s'agit en aucun cas de *détecter* une taxonomie unique et prédéterminée, mais au contraire, d'*aider* le concepteur de la base de retrouver une **organisation de ses connaissances qui le satisfait** ; pour cela le processus de conception doit être conçu de telle manière que le concepteur puisse exprimer son «point de vue» ;

Intégration au sein de la base l'ensemble des objets à examiner constitue une partie de la base et peut entretenir des liens variés avec le reste de son contenu – essentiellement d'autres objets et classes sur ces objets – un contenu que le processus cherche à enrichir ; ce processus ne peut donc se concevoir que comme complètement intégré au sein de la base et donc travaillant **directement** sur les descriptions des objets afin de tirer au maximum parti de la connaissance déjà formalisée ;

Flexibilité vis-à-vis de l'état de la base le contenu de la base de connaissances reflète l'état de la formalisation des connaissances sur le domaine ; celui-ci s'enrichit progressivement et le processus doit pouvoir **s'adapter** à cette richesse variable dans la recherche de classes pertinentes ;

Forte structuration des connaissances manipulées les objets constituent un formalisme très **riche** en moyens d'expressions (types de données variés) et **hautement structuré** (collections, relations entre objets) ; le processus de construction et plus précisément les mécanismes d'inspection et comparaison d'objets doivent porter sur **l'intégralité des descriptions** des objets, indépendamment de leur degré de structuration.

Compte tenu des remarques précédentes, nous pouvons formuler les facteurs auxquels les divers éléments doivent répondre. Tout d'abord, en ce qui concerne les outils de construction de classes, ils doivent être *complets* du point de vue des descriptions des objets : la totalité de leurs attributs doit pouvoir contribuer dans la construction. En particulier, les propriétés et les relations des objets ne doivent pas être séparées mais traitées ensemble et cela en préservant une adéquation avec leur sémantique. Le mélange entre ces différentes sortes de caractéristiques doit être fait selon un principe explicitement établi afin de permettre l'interprétation des résultats de la construction.

Ensuite, il n'y a pas une manière unique d'organiser un ensemble d'objets en classes. Une méthode de construction doit donc permettre de construire des taxonomies différentes, en exprimant des «visions» différentes au moyen des entrées de la méthode. L'existence de plusieurs taxonomies sur le même ensemble d'objets dans TROEPS fait de ce système un terrain d'application naturel pour les techniques de construction automatique de classes.

Finalement, la stratégie d'application des méthodes de construction doit être suffisamment *flexible* afin de permettre la construction des classes à des stades variés de la constitution d'une base de connaissances : en présence de taxonomies sur peu de familles d'objets ainsi qu'en ayant une ou plusieurs taxonomies pour chaque famille. Cela doit permettre à ces méthodes de tirer profit des connaissances existantes de la base afin de construire des classes qui seraient plausibles au maximum. En particulier, elle doit assurer l'application simultanée de la méthode sur plusieurs familles à la fois afin de résoudre les problèmes liés à la co-définition (liens conjugaux, autres relations circulaires, etc.).

En somme, la construction de classes doit tenir compte :

– d'un point de vue statique :

  – des divers types de données, éventuellement extensibles, dans les descriptions des données,
  – de l'existence des collections dans les attributs des objets,
  – des liens entre objets établis par les relations,
  – des classes existantes sur les autres ensembles d'objets ;

– d'un point de vue dynamique :

  – de l'absence d'une taxonomie unique et «objective» que l'on cherche à découvrir ; la méthode doit offrir des moyens souples pour explorer l'espace des diverses solution possibles, pour cela des paramètres doivent être fournis dont l'effet sur le résultat est le plus intuitif possible,
  – selon le degré de complétion de la base, plusieurs ensembles d'objets peuvent nécessiter une organisation en classes ; il serait judicieux que le SRCO puisse guider l'utilisateur dans le processus global de raffinement des classes afin d'optimiser la réutilisation des classes préalablement constituées dans l'identification des nouvelles classes.

4 L'existant

Le processus de construction d'une base de connaissances qui a fait l'objet de multiples études en acquisition de connaissances, est communément présenté comme une activité comportant des éléments d'acquisition, d'implantation et de validation des connaissances impliquant un spécialiste humain du domaine considéré et d'un cognititien qui assure la médiation entre l'humain et le système.

Dans notre contexte, nous considérons que le spécialiste construit directement, sans intermédiaire, sa base de connaissances en formalisant progressivement ses propres connaissances. Il s'agit donc de lui offrir une aide dans ce processus de formalisation en lui imposant le moins possible de contraintes. En particulier, on ne fait aucune hypothèse sur les informations qu'il est capable de fournir au SRCO à propos des classes recherchées.

Les problèmes liés à l'aide à l'acquisition par des outils automatiques d'analyse des connaissances ont été largement abordés, notamment au sein de la communauté d'apprentissage automatique. Les travaux ont abouti à un ensemble de systèmes intégrés permettant de combiner l'acquisition et l'apprentissage dont l'efficacité a été prouvée par de multiples applications [Morik *et al.*, 1993; Webb *et al.*, 1999; Causse et Nédellec, 1992; Nédellec *et al.*, 1995].

Toutefois, ces approches diffèrent sensiblement de nos préoccupations. En effet, d'une part, l'utilisation de ces systèmes suppose la présence d'un cognititien et d'autre part, elle requiert une connaissance relativement précise de la part du spécialiste sur les catégories du domaine : il est censé pouvoir donner des exemples et des contre-exemples de celles-ci.

Ne pouvant pas exiger du concepteur l'identification des membres des classes recherchées, nous nous tournons vers d'autres méthodes de construction de classes qui ne demandent que très peu d'indications dans leur travail. Celles-ci sont qualifiées de *non-supervisées*, par opposition aux méthodes évoquées précédemment qui sont quant à elles *supervisées* car elles bénéficient d'une connaissance très précise fournie par le concepteur.

Des méthodes de regroupement d'individus sans indication de classe ont été proposées dans le cadre de l'apprentissage automatique mais aussi en analyse de données. Le principe sous-jacent de toutes les méthodes est la recherche de groupes homogènes et contrastés au sein d'un ensemble d'individus.

L'analyse de données offre un cadre plus formel pour l'identification des classes, avec des algorithmes efficaces et robustes qui ont souvent un fondement mathématique. Les approches sous-jacentes sont pour la majorité basées sur la notion de distance entre deux individus. Toutefois, le problème du regroupement est défini sur un ensemble d'individus unique, sorti de son contexte, c'est-à-dire sans connaissance sur le domaine dont il est issu. Les caractéristiques des individus sont souvent le fruit d'un codage sur leurs propriétés initiales, lequel codage est exigé par l'application de l'appareil mathématique.

Le regroupement conceptuel en apprentissage cherche à remédier aux insuffisances de l'approche statistique en s'attachant à produire des descriptions pour les classes, à traiter les propriétés des individus de manière directe et à utiliser le plus possible les connaissances du domaine. Un principe de base de l'approche est de construire les classes qui optimisent la qualité des descriptions qui peuvent en être faites. Les critères de qualité formulés sont fondés sur des résultats des domaines voisins de l'IA, telle la psychologie cognitive, la statistique, mais aussi les neurosciences.

La majorité des méthodes d'apprentissage souffrent d'insuffisances quant à notre problématique. En effet, conçues dans une optique d'autonomie pour l'agent apprenant, elles ne laissent que très peu de place pour les interactions avec leur utilisateur en cherchant à établir *la* taxonomie optimisant un critère unique et préétabli. De plus, même si plusieurs types de données sont admis dans les descriptions des individus, leur ensemble est naturellement limité par la nature du critère de qualité utilisé. Les attributs à valeurs multiples ne sont quasiment pas considérés. Finalement, les relations entre individus sont présentes, quand c'est le cas, sous une forme particulière qui les rend externes à la description de ces individus. Cela conduit à la considération, comme dans le cas des méthodes d'analyse des données, d'un seul ensemble d'individus. Les rares exceptions à cette règle imposent des contraintes sur la nature des relations entre les différents ensembles d'individus distingués qui limitent leur applicabilité.

5 Notre approche

La thèse que nous avançons ici est qu'afin de répondre conjointement à tous les besoins évoqués plus haut, la méthode recherchée doit montrer une flexibilité et une puissance suffisantes. Ceci n'est pas le cas des méthodes effectives existantes. Nous nous proposons donc d'étendre une des approches, celle par proximité, qui nous semble être la plus proche de nos occupations.

L'approche que nous décrivons dans ce document repose donc sur l'utilisation d'une mesure de dissimilarité entre objets. La mesure s'applique uniformément sur les attributs constituant les descriptions des objets en respectant un principe unique qui permet un mélange entre propriétés et relations : un ensemble d'entités (objets, valeurs, collections) sont d'autant plus semblables entre elles que la description commune, ou la généralisation, qu'elles admettent est spécifique. Le principe se traduit naturellement sur les types de données mais aussi sur les objets. Dans ce second cas, l'écart peut être mesuré de deux manières permettant ainsi de s'adapter à des situations variées. Si les objets considérés possèdent une classe commune, celle-ci ou son degré de spécificité est utilisé pour estimer la similitude entre les objets. Dans le cas contraire, ce sont les descriptions propres des objets qui sont comparées. Les collections sont comparées en établissant un appariement préalable entre leurs membres. Les cas de co-définition entre couples d'objets sont traités en posant un système d'équations.

La valeur de la dissimilarité telle que nous l'avons définie est toujours possible à calculer. Pour la contrôler, le concepteur pourra utiliser des poids sur les attributs exprimant ainsi sa vision sur leur importance relative vis à vis du but du regroupement.

L'application de la dissimilarité transforme un ensemble d'objets en une matrice carrée de dissimilarités permettant ainsi de diviser la construction des classes en deux étapes consécutives. Tout d'abord, un algorithme de regroupement automatique divise les objets en un ensemble de groupes compacts et contrastés. Ensuite, chaque groupe est transformé en une classe et reçoit une description qui résume les descriptions des objets membres.

Ce procédé permet de déjouer les pièges de la co-définition : les classes sont d'abord identifiées et ensuite décrites, ce qui permet en particulier les références croisées dans leurs descriptions. Par exemple, les classes de clients et les classes des produits pourront être construites en même temps puis décrites ; chaque classe faisant référence à une classe de la famille opposée.

Une stratégie globale permet de séparer les ensembles de familles co-définies d'une base afin de traiter les concepts d'un ensemble simultanément. Les ensembles sont quant à eux, traités selon l'ordre induit par les attributs relationnels inter-ensembles.

La construction automatique peut être appliquée à n'importe quel ensemble d'objets, en particulier elle peut être utilisée pour partager les instances d'une classe dans une taxonomie déjà existante. Cela permet une construction souple et itérative où les diverses parties d'une taxonomie sont construites selon des critères différents.

Un certain nombre d'opérateurs de modification locale sur une taxonomie (suppression, fusion, promotion de classes), complètent les moyens nécessaires pour une aide efficace à la conception de taxonomies.

6 Apport du travail

Les principaux apports de notre travail peuvent être résumés en quelques points :

- définition d'un principe de comparaison entre objets qui permet de considérer de manière uniforme la totalité des informations qui les caractérisent : propriétés, relations à d'autres objets, classes d'appartenance,

- définition d'une mesure de proximité sur les objets qui traduit ce principe le plus fidèlement possible, afin de demeurer intelligible pour l'utilisateur de la méthode de construction,

- étude de l'extension de la mesure sur des cas particuliers tels les attributs multivalués et les relations menant à une circularité,

- preuve de propriétés intéressantes de cette mesure, propriétés formelles (propriétés de mesure de dissimilarité, de métrique, etc.) ainsi qu'intuitives (croissance et décroissance en fonction des arguments par exemple),

- définition des opérateurs d'inférence de description des classes (généralisation) en particulier dans le cas des attributs multivalués,

- définition d'algorithmes efficaces de calcul de la mesure et des descriptions,

- définition d'une méthode générique de construction de classes qui sépare le processus en trois phases : calcul de dissimilarité, regroupement automatique, caractérisation des classes,

- spécification d'une stratégie de construction de classes sur un ensemble de collections d'objets qui tient compte des éventuels liens entre ces classes,
- implantation d'une version de la mesure,
- premières expérimentations avec le module de construction de taxonomies T-TREE.

7 Structure du document

Le document est organisé en trois parties. La première partie décrit le contexte de notre travail. Le chapitre 1 est un survol de la représentation de connaissances structurées et plus particulièrement de l'approche objet. Il présente la problématique du domaine, ainsi que les diverses familles de formalismes existants. L'accent est mis sur les caractéristiques liées au problème de la construction de taxonomies. À la fin, nous formulons un ensemble de critères auxquels une stratégie de construction de taxonomies au sein d'un formalisme à objets doit répondre. Le chapitre suivant est consacré aux méthodes de construction automatique de taxonomies développées en analyse de données et en apprentissage symbolique automatique. Nous discutons les divers aspects de ces méthodes avant de présenter quelques systèmes concrets répartis selon le principe effectif de construction de classes. Les mérites et les insuffisances de chaque approche par rapport à notre problématique sont évalués. Pour terminer, nous esquissons une approche pour construire des taxonomies dans les objets.

Dans la seconde partie, nous présentons notre méthode de construction de taxonomies. Dans le premier chapitre, nous décrivons la stratégie de construction. Les outils qu'elle met en œuvre (la mesure de dissimilarité entre objets, les algorithmes de regroupement et les opérateurs de caractérisation) sont introduits avec leur principes fondateurs. Le second chapitre est réservé à la production de descriptions pour les classes retrouvées par regroupement automatique. Le chapitre trois présente une mesure de dissimilarité entre objets qui permet de combiner les différences sur l'ensemble des attributs admissibles dans le modèle d'objets. Le chapitre suivant considère un problème particulier aux formalismes structurés : le traitement des collections. Une manière générique de comparer les collections en accord avec les principes énoncés au chapitre précédent est présentée. Le dernier chapitre est consacré au traitement des circularités dans les descriptions des objets. Il décrit la manière de détecter les couples d'objets dont la dissimilarité doit être calculée en même temps ainsi que la composition et la résolution d'un système d'équations dont les solutions sont les valeurs de la dissimilarité.

La troisième partie est consacrée à la mise en œuvre de la stratégie de construction de classes au sein du SRCO TROEPS. Le premier chapitre présente la réalisation de la stratégie avec les diverses étapes de la construction d'une taxonomie et l'architecture du module T-TREE qui constitue un environnement de construction de taxonomies intégré à TROEPS. Dans le dernier chapitre, nous donnons un exemple complet de l'utilisation de T-TREE.

Première partie

Contexte

1

La représentation de connaissances à objets

Le but du travail présenté étant de définir des techniques d'analyse (détection de groupements pertinents d'objets) au sein d'un système de gestion de bases de connaissances à objets, nous consacrons le premier chapitre au cadre général, à savoir la représentation de connaissances.

L'accomplissement d'une tâche «intelligente» par un système est lié à la mise en œuvre d'un ensemble de connaissances sur le domaine concerné. La représentation de connaissances (RC) a pour but de rendre possible le stockage et les diverses manipulations des connaissances au sein de la machine et cela de manière efficace et intelligible pour un utilisateur humain. Les approches variées vers la représentation ont donné suite à de nombreux formalismes concrets. Parmi celles-ci, l'approche objet met en exergue le besoin de représentations structurées. La structure dans les objets avec tous ses aspects et variations est au centre de nos préoccupations car c'est la source principale des difficultés pour la construction automatique de classes.

Au début du chapitre nous tâcherons de présenter une vision d'ensemble du domaine de la représentation de connaissances (section 1.1). Après avoir décrit les principes généraux régissant le domaine, nous nous arrêtons rapidement sur la logique des prédicats d'ordre un (section 1.2.1) qui constitue le formalisme de RC le mieux étudié. Ensuite, nous nous concentrerons sur les formalismes de représentation structurés : les graphes conceptuels (section 1.2.2) et les logiques de descriptions (section 1.2.3). La section suivante est dédiée aux objets structurés (section 1.3). Une réalisation concrète des concepts du paradigme objets, le système Troeps, est présentée dans section 1.4 car ce système constitue le cadre de notre étude. La comparaison entre les trois derniers formalismes nous permettra de mettre en évidence un certain nombre de traits caractéristiques des objets qui constituent leur spécificité (section 1.5). L'accent est mis sur les aspects des formalismes en lien avec le regroupement automatique (chapitre 2).

En conclusion, on aura identifié les problèmes spécifiques que la construction de taxonomies au sein d'un système à objets pose. Le chapitre suivant pourra alors être consacré aux résolutions possibles de ces problèmes.

1.1 Représentation de connaissances

Les expériences avec l'application des premiers systèmes d'IA ont rapidement mis en évidence l'importance des connaissances dans l'accomplissement d'une tâche «intelligente». Il en résulte donc le besoin de fournir à un tel système un ensemble de connaissances concernant le domaine concret d'application. Dès lors, la voie a été ouverte pour le développement de divers moyens de représentation de la connaissance. Ceux-ci ont été fédérés dans une nouvelle discipline au sein de l'IA, la représentation de connaissances (RC). Le but de la discipline est de concevoir des formalismes admettant des «descriptions du monde qu'une machine intelligente pourrait manipuler de manière formelle afin de faire des conclusions sur son environnement» [Brachman et Levesque, 1985].

1.1.1 Langages de RC

La définition ci-dessus met en évidence les deux composantes complémentaires de la RC, à savoir l'expression et la manipulation des connaissances. D'une part, les connaissances sont exprimées à l'aide d'un langage formel, dit de description des connaissances. Le langage est doté d'une *syntaxe*, précisant la manière de construire des expressions admissibles du langage et d'une *sémantique* qui permet de caractériser l'ensemble des situations qui peuvent être modélisées. D'autre part, le but est de mécaniser un certain nombre de manipulations sur les connaissances exprimées. Ainsi, il sera nécessaire de modifier, compléter, inférer de nouvelles connaissances. Les manipulations possibles sont spécifiées sous forme de mécanismes opérant sur les éléments de la représentation. Il est souhaitable que ces mécanismes respectent la sémantique.

1.1.2 Aspects d'une représentation

Un système à base de connaissances (SBC) est un système disposant d'une base qui contient l'expression explicite des connaissances manipulées par le système. La base de connaissances (BC) constitue en quelque sorte un modèle du domaine d'application du système. Dans les SBC, on distinguera entre les connaissances *descriptives* et les connaissances *inférentielles*. Les premières décrivent ce qui est et les secondes comment le manipuler.

Les expressions du langage de description sont interprétables vers des entités du domaine modélisé qui constituent leur *dénotation*. En fonction de la dénotation, nous allons distinguer deux types de connaissances descriptives, les connaissances *individuelles* et les connaissances *génériques*. Les connaissances individuelles, souvent comparées aux données, expriment des faits portant sur des entités concrètes d'un domaine. Leur dénotation est une entité identifiable. Les connaissances génériques concernent des groupes, ou catégories, d'entités. Il est donc nécessaire que ces deux types de connaissances soient identifiés dans le langage de description. Souvent, la distinction est telle que deux langages différents, un pour les individus et autre pour les catégories, sont employés.

Un autre aspect concerne le statut des expressions décrivant des catégories vis-à-vis de l'ensemble d'entités concrètes constituant leur dénotation. Certains formalismes accordent un pouvoir définitionnel aux descriptions des catégories, c'est-à-dire que celles-ci sont considérées comme fournissant des conditions nécessaires et suffisantes pour l'appartenance d'un individu à la catégorie. Ces formalismes suivent une vision classique sur la représentation des catégories qui a une longue tradition en philosophie et psychologie cognitive. Cette vision a été mise en cause à multiples reprises, en particulier dans les travaux de Wittgenstein qui a souligné l'existence de concepts dont une condition nécessaire et suffisante (CNS) est impossible à donner. Pour refléter ce phénomène, un groupe de formalismes de RC affectent une sémantique plus souple aux descriptions des catégories : celles-ci ne sont interprétées qu'en conditions nécessaires (CN). Finalement, en suivant les travaux sur la théorie des prototypes, en particulier ceux de Rosch [Rosch et Mervis, 1975], un troisième type de formalismes se contente de décrire les catégories par des traits qui sont les plus typiques, mais ne sont pas forcément partagés par tous les membres de la catégorie.

Les entités d'un domaine ne sont jamais prises isolément, elles sont impliquées dans divers types de relations et le langage de description doit permettre l'expression des connaissances sur ces relations. Selon l'importance des relations au sein d'une représentation, on distingue les langages (ou modèles) *relationnels* des langages *structurels* [Napoli, 1992]. Les premiers favorisent l'expression des relations dans lesquelles une entité participe : l'entité n'est décrite que par ses relations et la connaissance sur celle-ci est répartie entre les diverses relations. Les seconds organisent les connaissances sur les entités, individuelles ou génériques, au sein des unités de description structurées et autonomes. Il est à noter que la plupart des langages de représentation actuels mélangent les modes structurel et relationnel.

1.1.3 Exigences envers la RC

Une représentation, afin d'aboutir à la construction d'un modèle du domaine d'application, doit répondre à une multitude de critères. Nous en citons les plus saillants :

expressivité : le langage de description doit être suffisamment riche pour permettre l'expression des connaissances utiles,

intelligibilité : les descriptions produites doivent rester concises et compréhensibles afin de faciliter l'examen du contenu d'une BC,

efficacité : les mécanismes de manipulation doivent fournir un résultat dans un temps raisonnable,

correction et complétude : les mécanismes d'inférence doivent produire, à partir d'un ensemble de faits valides, de nouveaux faits qui sont valides (correction) et il doivent être capables de produire tous les faits valides (complétude).

Il est difficile de répondre à toutes les exigences, d'autant plus que certaines d'entre elles sont antinomiques. Ainsi, la richesse des expressions va à l'encontre de l'efficacité des mécanismes opérants sur ses descriptions et donc un compromis est nécessaire. Sur cette controverse largement étudiée au sein de la RC [Levesque et Brachman, 1987], les formalismes existants apportent chacun leur propre réponse de sorte qu'un continuum s'est créé entre les systèmes à langage riche et mécanismes lents et les systèmes à expressivité réduite avec exploitation efficace, mais aussi des systèmes riches et efficaces, mais incomplets.

1.2 Quelques formalismes de représentation

Les travaux en RC ont donné suite à de nombreux formalismes généraux, c'est-à-dire dédiés à la représentation des entités de nature variable (objets, actions, états, etc.). Nous donnons quelques uns des plus utilisés ci-dessous. Chacun est examiné du point de vue de ses capacités descriptives (syntaxe, sémantique) et inférentielles. L'accent est mis sur la structure des expressions individuelles et génériques.

1.2.1 Logique des prédicats

La logique classique est le premier formalisme utilisé pour modéliser les connaissances et de loin le mieux étudié. Conçue pour formaliser le raisonnement, elle offre un large éventail de moyens d'expression, dotés d'une sémantique claire, dépourvue de toute ambiguïté ainsi qu'un ensemble de mécanismes d'inférence corrects et complets. Dans la suite, nous présentons brièvement la logique des prédicats du premier ordre (LPO). Elle repose sur l'hypothèse que le monde à modéliser est constitué d'entités (objets), de leur propriétés et des relations entre entités.

Syntaxe

La syntaxe d'un langage de LPO est définie à deux niveaux : langage de termes et langage de formules.

Les *termes* décrivent les objets du domaine. Le langage des termes est composé d'un ensemble de constantes `C`, des symboles de fonction `F` et un ensemble de variables `V`. Un terme est soit une constante, soit une variable, soit une fonction appliquée à d'autres termes.

`Terme`	$\rightarrow$	`Constante`
		| `Variable`
		| `Fonction(Terme,...)`

TAB. 1.1: *La syntaxe du langage des termes en LPO*

Le langage des formules est composé de symboles de prédicats, de termes, de connecteurs logiques, de quantificateurs (universel et existentiel). Les prédicats s'appliquent aux termes et forment des expressions atomiques. Les atomes sont liés par des connecteurs dans des formules complexes. Une formule peut être quantifiée. Un ensemble de connecteurs est donné dans la table ci-dessous. Ceux-ci sont complets dans la mesure où le reste des connecteurs peuvent être obtenus par une combinaison à partir des connecteurs cités.

F	$\rightarrow$	$P(t_1, ..., t_n)$
		$\mid F \wedge F$
		$\mid \neg F$
		$\mid \forall x\ F$

TAB. 1.2: *La syntaxe du langage des formules en LPO*

Par exemple, les formules suivantes sont constituées sur un langage comportant les prédicats `appartement`, `humain`, `habitation`, ainsi que les constantes `Alex` et `Valmy-6`. Les connecteurs $\exists$ et $\Rightarrow$ sont définis à partir des connecteurs de la table 1.2.

1. $humain(Alex)$, $appartement(Valmy-6)$
2. $\forall x\ humain(x)\ \Rightarrow\ \exists y\ appartement(y)\ \wedge\ habite(x,y)$
3. $habite(Alex, Valmy-6)$

Intuitivement, celles-ci expriment le fait que l'entité représentée par `Alex` est un humain et que celui-ci habite dans l'appartement désigné par `Valmy-6`.

Sémantique

La sémantique des expressions d'un langage de premier ordre est donnée moyennant une *interprétation* $\mathcal{I} = \langle D_{\mathcal{I}}, .^{\mathcal{I}} \rangle$ où $.^{\mathcal{I}}$ est une *fonction d'interprétation* vers un ensemble D dit *domaine d'interprétation*. Les constantes et les variables s'interprètent comme des éléments de l'ensemble D. Les symboles de fonction et de prédicat sont interprétés par des fonctions et des

$\mathcal{I}(c)$	$\in$	D	$\mathcal{I}(f)$	$\in$	$D^n \rightarrow D$	$\mathcal{I}(p)$	$\subseteq$	D^n

TAB. 1.3: *Le domaine d'interprétation des symboles.*

relations du domaine d'interprétation. La sémantique des connecteurs et des quantificateurs est établie cas par cas.

Une valeur de vérité est assignée à chaque formule dans l'interprétation $\mathcal{I}$ de la manière suivante. D'abord, on associe à chaque terme un élément du domaine. Pour cela, on fixe une valuation des variables $\sigma : V \rightarrow D$. La valuation est étendue de manière inductive sur les termes.

$\mathcal{I}(c)$	$=$	$\sigma(c)$
$\sigma(f(t_1, t_2, ..., t_n))$	$=$	$\mathcal{I}(f)(\sigma(t_1), \sigma(t_2), ..., \sigma(t_n))$

TAB. 1.4: *L'interprétation des termes par rapport à une valuation σ.*

La vérité d'une formule Φ dans l'interprétation $\mathcal{I}$ et par rapport à la valuation σ, notée $\mathcal{I} \models_\sigma \Phi$, est définie par induction. Une formule atomique $p(t_1, t_2, ..., t_n)$ est vraie si le n-uplet des valuations de ses arguments $\langle\sigma(t_1), \sigma(t_2), ..., \sigma(t_n)\rangle$ appartient à l'interprétation du prédicat $\mathcal{I}(p)$. La vérité d'une formule complexe est établie en fonction des connecteurs et des quantificateurs qu'elle fait apparaître.

$\mathcal{I} \models_\sigma p(t_1, t_2, ..., t_n)$	`ssi`	$(\sigma(t_1), \sigma(t_2), ..., \sigma(t_n)) \in \mathcal{I}(p)$
$\mathcal{I} \models_\sigma$ `true`		
$\mathcal{I} \models_\sigma \neg\Phi$	`ssi`	`non` $\mathcal{I} \models_\sigma \Phi$
$\mathcal{I} \models_\sigma \Phi \wedge \Psi$	`ssi`	$\mathcal{I} \models_\sigma \Phi$ `et` $\mathcal{I} \models_\sigma \Psi$
$\mathcal{I} \models_\sigma \forall x \Phi$	`ssi`	$\mathcal{I} \models_\sigma [x/d]\Phi$ `pour tout` $d \in D$

TAB. 1.5: *Sémantique des formules*

Une formule Φ est *vraie* dans l'interprétation $\mathcal{I}$ si pour toute valuation σ, $\mathcal{I} \models_\sigma \Phi$. Dans ce cas, $\mathcal{I}$ est un *modèle* de Φ. L'interprétation $\mathcal{I}$ est *modèle* pour un ensemble de formules S si et seulement si elle est modèle pour chacune des formules. Avec deux ensembles S et S', éventuellement réduits à une seule formule, l'ensemble S' est une *conséquence* de S (noté $S \models S'$) si tous les modèles de S sont des modèles de S'.

Raisonnement

Le principal mécanisme de raisonnement est la déduction logique. A partir d'un ensemble de formules, on établit la vérité d'une assertion dans l'univers qui est décrit. Ceci équivaut à vérifier si l'assertion en est une conséquence logique. La déduction est élaborée par l'application d'un ensemble de règles d'inférence. Divers systèmes de règles existent : le *modus ponens* avec la *spécialisation universelle* en forment un, la *résolution* entre clauses en est un autre.

Bilan

Les langages logiques se caractérisent par une grande richesse et finesse de l'expression des connaissances. Les expressions logiques sont claires et non ambiguës, il est facile de leur donner un sens à partir du sens de leur composants atomiques et les connecteurs utilisés [Haton *et al.*, 1991]. Les inférences faites avec les règles sont complètes et correctes.

Malgré ses qualités indéniables, la logique des prédicats est peu adaptée à la représentation de connaissances structurées. En effet, pour représenter un objet complexe, on utilise un nom et un ensemble des propriétés et relations. Ceux-ci font partie des formules logiques qui les lient. Ainsi, les informations concernant l'objet se retrouvent dispersées au sein des diverses formules d'une base sans structure. Ce fait freine une interprétation des descriptions des entités et par là, leur modification et la vérification de leur cohérence. Finalement, la complexité algorithmique des mécanismes de raisonnement associés à la logique les rend inadaptés pour le traitement de grandes volumes de connaissances.

1.2.2 Graphes conceptuels

L'objectif initial de la création des graphes conceptuels est la représentation intelligible des assertions en langue naturelle. Ils étaient destinés à palier les inconvénients de la logique classique, à savoir, le manque de structure dans les descriptions et dans l'organisation d'une base de connaissances. En même temps, les graphes conceptuels s'inspirent fortement des réseaux sémantiques, on peut les voir comme une extension de ces derniers mais avec une syntaxe et une sémantique claires. Pour cette raison nous commençons le paragraphe dédié aux graphes conceptuels par une brève présentation des réseaux sémantiques.

Réseaux sémantiques

Les réseaux sémantiques [Quillian, 1968] ont été conçus à l'origine comme un modèle de la mémoire humaine. Ils sont fondés sur un modèle graphique permettant de combiner la représentation des concepts (sous forme de nœuds) et des relations entre concepts (sous forme d'arcs). Un mécanisme de raisonnement fondé sur le parcours de la structure graphique, la propagation de marqueurs, permettait d'établir des liens entre des concepts du même réseau. Les réseaux sémantiques ont le mérite d'avoir introduit la notion d'héritage, matérialisée par des arcs particuliers (sorte-de) entre des nœuds représentant des concepts génériques. Les arcs d'héritage introduisent une organisation hiérarchique des concepts qui favorise une factorisation des informations au sein d'un réseau et facilite le raisonnement. Celui-ci consiste en un filtrage qui permet d'apparier des structures de réseaux. D'autres types de raisonnement se sont constitués exploitant la structure graphique des réseaux et sans égard à la nature des arcs parcourus. Le manque d'une sémantique claire des éléments d'un réseau [Woods, 1975] (les arcs, les liens d'héritage) a été à l'origine des études poussées destinées de définir un cadre formel pour le modèle [Brachman, 1977]. Celles-ci ont profité essentiellement au développement d'autres formalismes de représentation, tels les graphes conceptuels, les logiques de description et la représentation par objets.

Syntaxe d'un langage de graphes conceptuels

Un graphe conceptuel est un multi-graphe connexe biparti composé de deux types de nœuds : des *concepts* et des *relations*. Les nœuds relations possèdent un ou plusieurs arcs qui les lient aux nœuds concepts.

L'exemple ci-dessous, donné dans la notation dite *linéaire*, exprime le fait que l'appartement identifié comme app-5 est situé à 5, avenue de Valmy, a pour propriétaire l'homme appelé Alex :

Exemple :
```
[appartement:app-5] -> (propriétaire) -> [humain : alex] -> (situe-à) ->
[adresse : 5, avenue de Valmy]
```

Les **concepts** désignent des entités, des propriétés, des états ou des événements. Ils consistent en une étiquette, un couple (*type de concept*, *référent*). Le **référent** peut être *générique* ou *individuel*. Un nœud conceptuel à référent individuel désigne un individu concret, par exemple `[humain : alex]` désigne l'homme portant ce prénom. Ainsi, le référent identifie l'individu sous-jacent. Le référent générique, noté par `*`, permet au concept de désigner n'importe quel individu. Ainsi, `[humain : *]` est la représentation d'un individu non précisé de type `humain`.

Le **type** est un type d'individus, tiré du **treillis des types** (cf. plus loin). Chaque type définit un ensemble de contraintes sur la structure des individus qui sont des instances de ce type. Un individu est lié à son type (en devient instance) par la relation de conformité (il satisfait les contraintes).

Les liens existant entre les individus sont représentés par les nœuds relationnels qui lient un certain nombre de nœuds conceptuels. Chaque relation est d'arité fixe.

Le vocabulaire, sur lequel sont construits les graphes conceptuels d'une représentation concrète (ensemble de types, référents et relations) est défini moyennant la notion de *support* ([Chein et Mugnier, 1992]) permettant de contraindre la construction des graphes. Ainsi, l'ensemble de types est organisé par la relation dite de sous-typage en un *treillis de types*. Le treillis est muni d'éléments minimal et maximal. La relation de conformité est explicitement donnée et elle s'étend transitivement sur le treillis : un individu est conforme à tous les surper-types de son type associé. De la même manière, l'ensemble des types de relations est partiellement ordonné par une relation de généralité. Un ordre est également spécifié entre les référents individuels qui sont deux à deux incompatibles mais plus spécifiques que le référent générique (*). Finalement, un ensemble de graphes-étoiles permet de fixer les signatures des relations utilisées. Un graphe étoile est composé d'un seul nœud relationnel lié à l'ensemble des concepts correspondant à chacun de ses arcs. La signature définit les types maximaux (pour le sous-typage) pour les nœuds conceptuels qui peuvent être connectés à chaque arc.

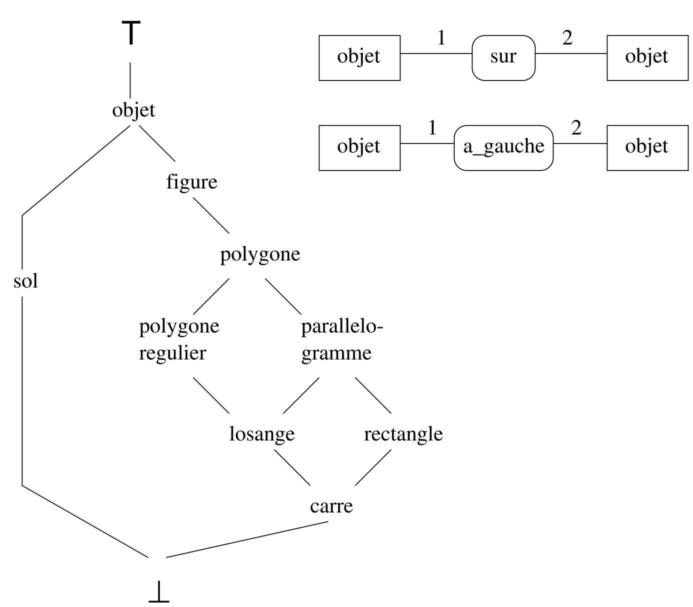


FIG. 1.1: *Exemple de support avec le treillis de types et les graphes étoiles.*

Formellement, un support $\mathcal{S}$ est un quintuplet $\langle T_C, T_R, M, ::, B\rangle$ [Mugnier et Chein, 1996] tel que :

T_C est l'ensemble des types de concepts ;

T_R est l'ensemble des types de relations ;

M est un ensemble dénombrable de marqueurs contenant :

> $\top$ le marqueur générique ;
>
> $\perp$ le marqueur absurde ;

$::$ est un prédicat de conformité sur $M \times T_C$;

B est un ensemble de graphes étoiles sur T_C, T_R ;

et :

- $\langle T_C, \leq \rangle$ est un treillis fini avec $\top_t$ et $\perp_t$ comme éléments maximal et minimal respectivement ;
- $T_C \cap T_R = \emptyset$;
- il existe une bijection $\beta : T_R \longrightarrow B$ telle que $\forall r \in T_R, \beta(r)$ est un graphe étoile pour r ;
- $\langle M, \sqsubset \rangle$ est un treillis construit de telle sorte que $\forall m \in M - \{\top_m, \perp_m\}, \perp_m \sqsubset m \sqsubset \top_m$ et tous les éléments de $M - \{\top_m, \perp_m\}$ sont incomparables ;
- pour tout $m \in M$ et tout $t, t' \in T_C$, les propriétés suivantes sont vérifiées :

$$\begin{array}{rr} m :: \top_t, & (t' \leq t) \wedge (m :: t') \Rightarrow m :: t \\ m \not:: \perp_t, & (m :: t') \wedge (m :: t) \Rightarrow m :: t' \wedge t \\ \perp_m \not:: t, & t \neq \perp_t \Leftrightarrow \top_m :: t. \end{array}$$

Un graphe conceptuel sur le support S est donc un multi-graphe $G = \langle R, C, U, l \rangle$ tel que :

- $\langle R, C, U \rangle$ est un multi-graphe biparti fini avec $C \neq \emptyset$;
- l est une fonction

$$\begin{aligned} l : \ & R \longrightarrow T_R \\ & r \longmapsto \mathit{type}(r) \\ & C \longrightarrow T_C \times (M - \perp_m) \\ & c \longmapsto \langle \mathit{type}(c), \mathit{ref}(c) \rangle \end{aligned}$$

- $\forall c \in C, \mathit{ref}(c) :: \mathit{type}(c)$;
- $\forall r \in R,$;
  1. les arcs sortants de r sont totalement ordonnés de 1 à $\mathit{degré}(r)$ (et leurs extrémités sont nommées $G_i(r)$) ;
  2. $\mathit{degré}(r) = \mathit{degré}(\beta(\mathit{type}(r)))$;
  3. $\forall i \in 1, \ldots \mathit{degré}(r), \mathit{type}(G_i(r)) \leq \beta_i(\mathit{type}(r))$

Des types complexes peuvent être définis à l'aide des graphes conceptuels, on associe un type à un graphe. Les opérations d'*expansion* et de *contraction* permettent à tout instant de passer d'un type complexe à sa définition qui est graphe, et *vice versa*. Il est ainsi possible d'opérer des descriptions à des niveaux de granularité variables.

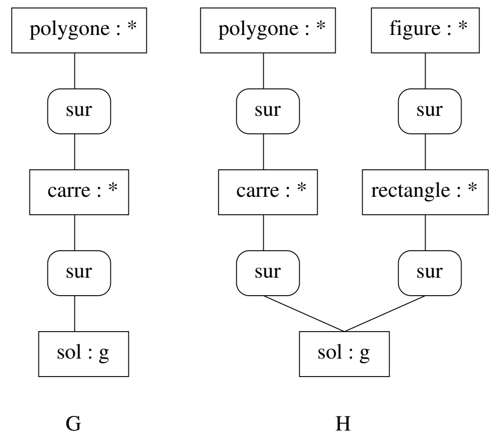


FIG. 1.2: *Exemple de graphes sur le support de la Fig. 1.1*

Sémantique

La sémantique d'un graphe conceptuel est établie moyennant un passage vers un langage logique d'ordre un ($\mathcal{CP}1$) réalisé à l'aide de l'opérateur φ défini par Sowa [Sowa, 1984].

A chacun des éléments du support est associé son équivalent dans le langage logique. Ainsi, les référents génériques sont traduits par des variables et les référents individuel par des constantes. Chaque type, conceptuel ou relationnel, est traduit par un prédicat. Les prédicats correspondant aux concepts sont monadiques, ceux des relations ont une arité égale au nombre d'arcs de la relation.

Soit un support $\mathcal{S} = \langle T_C, T_R, M, ::, B\rangle$, on considère un calcul des prédicats basé sur un ensemble de constantes $K \supseteq M - \{\top, \bot\}$, un ensemble de variables V, et un ensemble de symboles de prédicats $P \supseteq T_C \cup T_R$. Les prédicats associés à T_C sont d'arité 1, alors que les prédicats associés à T_R ont une d'arité égale au nombre d'arcs de la relation sous-jacente.

Soit G un $\mathcal{S}$-graphe $\langle R, C, U, l\rangle$, on définit [Euzenat, 1999] :

$$\begin{aligned}
\iota :\ & C \longrightarrow V \cup K \text{ injective sur } V \\
& c \longmapsto \begin{cases} x \in V \text{ si } ref(c) = \top \\ ref(c) \in K \text{ sinon} \end{cases} \\
\varphi :\ & C \cup R \cup \mathcal{S}\text{-graphe} \longrightarrow \mathcal{CP}1 \\
& c \longmapsto \mathit{type}(c)(\iota(c)) \\
& r \longmapsto \mathit{type}(r)(\iota(G_1(r)), \ldots \iota(G_{degr\acute{e}(r)}(r))) \\
& G \longmapsto \exists_{c\in C; ref(c)=\top}\iota(c) \bigwedge_{c\in C} \varphi(c) \wedge \bigwedge_{r\in R} \varphi(r)
\end{aligned}$$

Par exemple, le graphe G de la figure 1.2 se traduit par :

$$\exists x, y; sol(g) \wedge polygone(x) \wedge carr\acute{e}(y) \wedge sur(x, y) \wedge sur(y, g)$$

et le graphe H par :

$$\begin{aligned}
&\exists x, y, z, w; polygone(x) \wedge figure(y) \wedge carr\acute{e}(z) \wedge rectangle(w) \wedge sol(g) \\
&\qquad\qquad \wedge sur(x, z) \wedge sur(z, g) \wedge sur(y, w) \wedge sur(w, g)
\end{aligned}$$

La relation de généralité entre graphes suit l'implication entre les traductions de ces graphes en logique des prédicats.

Raisonnement

Une relation de généralisation/spécialisation est définie entre graphes conceptuels, notée $\leq_{\mathcal{S}}$. Un graphe conceptuel, G, est plus spécifique qu'un autre, H, si, intuitivement, il contient toutes les informations véhiculées par H et en ajoute d'autres. La *généralisation/spécialisation* s'exprime sur les graphes par une sorte particulière de morphisme de graphes, appelé la *projection*.

Ainsi, $G \leq_{\mathcal{S}} H$ si et seulement si il existe une projection de H vers G. Intuitivement, la projection est une application entre graphes qui préserve les étiquettes sur les relations et l'ordre des arcs, mais permet de restreindre les types ou les marqueurs des concepts.

Formellement, (voir [Euzenat, 1999]) une $\mathcal{S}$-projection entre deux $\mathcal{S}$-graphes $\langle R, C, U, l\rangle$ et $\langle R', C', U', l'\rangle$ est une paire d'applications

$$\begin{aligned} \langle f :\ & R \longrightarrow R', \\ g :\ & C \longrightarrow C'\rangle \end{aligned}$$

telle que :

- $\forall r \in R, \forall i \in 1, \ldots degré(r), g(G_i(r)) = G'_i(f(r))$;
- $\forall r \in R, l'(f(r)) = l(r)$;
- $\forall c \in C, type'(g(c)) \leq type(c)$ et $ref'(g(c)) \sqsubset ref(c)$.

La relation peut être établie en appliquant une suite d'opérations élémentaires de *spécialisation* (diminution du type d'un nœud conceptuel, par exemple) à H jusqu'à ce que l'on obtienne G (voir [Chein et Mugnier, 1992]).

Les opérations élémentaires de *généralisation* sont obtenues en inversant les opérations ci-dessus. Il a été démontré dans [Mugnier et Chein, 1996] que la relation $\leq$ induit une structure de treillis sur l'ensemble de graphes irredondants pour un support donné. L'irredondance est une forme normale de graphes conceptuels qui est définie comme l'absence d'équivalence (projection dans les deux sens) entre le graphe et un de ses sous-graphes stricts.

Les opérations de spécialisation sont en même temps des règles de bonne formation car elles permettent de construire des graphes conceptuels non contradictoires à partir d'un ensemble initial de graphes dits *canoniques*.

Les graphes d'une base sont organisés en une hiérarchie par la relation de généralité [Levinson et Ellis, 1992]. Le formalisme se prête à un raisonnement exact basé sur l'équivalence avec la logique du premier ordre. Il permet de faire des inférences valides à partir de la hiérarchie de généralisation.

Synthèse

Contrairement à leur prédécesseurs, les réseaux sémantiques, les graphes conceptuels introduisent une séparation entre les connaissances de nature différente. Les connaissances de nature universelle, la description des types conceptuels et de leurs relations de sous-typage, la caractérisation des relations sont organisées dans le support. La structure de treillis permet des vérifications efficaces des contraintes de types dans les graphes. Les connaissances de nature factuelle sont organisées au sein d'un ensemble de graphes. La sémantique logique garanti la correction des inférences faites dans le modèle. Finalement, le mécanisme de contraction/expansion permet la manipulation de concepts complexes.

D'autre part, le découpage des descriptions en unités graphes, introduit le même problème de dispersion des connaissances que dans le cas de la logique et des réseaux sémantiques. La description d'une entité du domaine est divisée entre les graphes qui la font apparaître.

1.2.3 Logiques de descriptions

Les logiques de descriptions (ou terminologiques) constituent l'aboutissement d'une longue période de recherche sur la formalisation dans les langages de représentation de connaissances. Elles ont profité des enseignements tirés de l'utilisation des réseaux sémantiques et des langages de frames dont une présentation succincte est donnée au début de ce paragraphe. Ainsi, les logiques de descriptions mettent l'accent sur la structure dans la description des connaissances génériques. Elles offrent des langages permettant de décrire les concepts d'un domaine (la terminologie) à l'aide de termes structurés. Ceux-ci peuvent être construits à partir d'autres termes moyennant des connecteurs. Ils sont organisés dans une hiérarchie selon la relation de subsumption. L'ancêtre de tout langage de la famille est incontestablement le système KL-ONE [Brachman et Schmolze, 1985].

Langages de frames

Le modèle des *frames* ou *schémas* a été proposée par Minsky [Minsky, 1975] afin de rendre compte des aspects du raisonnement humain qui été difficilement représentables par la logique ou les réseaux sémantiques, à savoir la réutilisation des expériences antérieures, l'accès rapide aux connaissances liées à un référent particulier, l'inférence, à partir des caractéristiques observées d'un objet ou une situation des caractéristiques non observées. Jugeant la granularité de représentation qu'offrait la logique insuffisante pour ces objectifs, Minsky a proposé une structure de données, le frame, qui renferme, en elle seule les connaissances concernant une entité du domaine étudié. Un frame, comparé par l'auteur à un formulaire avec des cases à remplir, représente une entité typique ou une situation stéréotypée.

Les idées de Minsky ont été une source d'inspiration pour des nombreux développements en RC. Malgré les divergences au niveau de la réalisation, la notion de frame a préservé son statut d'unité de représentation structurée au sein des SRC sous-jacents. Ainsi, un frame englobe une identité, c'est-à-dire un nom, et une description, l'ensemble de *slots* (*attributs*) du *frame*. Les attributs s'adressent aux diverses caractéristiques de l'entité dénotée qui peut être générique ou individuelle. Chaque attribut est à son tour structuré en un ensemble de *facettes*. Les facettes contiennent des valeurs, elles précisent le contenu de l'attribut au sein du frame (nature de l'information, moyen de l'obtenir, etc.). L'ensemble des frames est communément organisé en une hiérarchie où les frames génériques dominent les frames plus spécifiques. La notion d'héritage se retrouve intégrée au modèle : un frame peut être créé par référence à un autre frame et il en « hérite» les caractéristiques qu'il peut par la suite modifier, enrichir ou supprimer. Des inférences sur les frames sont faites à l'aide du mécanisme de *filtrage* qui consiste à apparier les descriptions, facettes et valeurs, de deux frames afin d'établir leur conformité.

Le manque de distinction entre les frames individuels et génériques, ainsi que la faiblesse des contraintes introduites par l'héritage, fait des frames un outil adapté pour la modélisation de hiérarchies partiellement définies. Toutefois, ils obscurcissent l'interprétation d'un frame et peuvent conduire à des absurdités. Dans [Brachman, 1985], R. Brachman a mis en évidence le besoin d'une distinction nette entre les aspects individuel et générique au sein du modèle.

Les logiques de descriptions constituent une des voies possibles d'évolution des idées à l'origine des frames. Une autre approche, plus pragmatique, que nous appelons les SRC à objets, est présentée dans la Section 1.3.

Syntaxe d'une logique de descriptions

Les entités manipulées par un langage de descriptions sont les individus, les concepts et les rôles. Les individus correspondent à des entités concrètes de l'univers du discours, alors que les concepts se réfèrent à des entités génériques. Les rôles décrivent des relations binaires entre individus.

Les connaissances individuelles et génériques sont introduites et manipulées à des niveaux différents au sein d'un système terminologique : le niveau terminologique pour les concepts et les rôles et le niveau assertionnel pour les individus. Chaque niveau dispose d'un langage de description propre. Une base de connaissances comporte une terminologie (la T-Box), décrite dans le langage terminologique et un ensemble d'assertions (la A-box) décrites dans le langage assertionnel.

Langage terminologique Au sein d'un système, le langage terminologique est destiné à la composition de termes structurés décrivant les concepts et les rôles.

Les termes sont construits sur un ensemble C de concepts atomiques (dont le concept universel, $\top$ et le concept absurde $\bot$) et un ensemble R de de rôles atomiques. À partir de ces deux ensembles, des termes structurés sont construits à l'aide d'une collection de connecteurs. Les connecteurs permettent de composer les descriptions de concepts ou de rôles, ainsi que de définir des restrictions sur les (co-domaines des) rôles. Les divers langages offrent des ensembles de connecteurs variables. Un ensemble minimal de connecteurs est fourni par le langage $\mathcal{FL}_0$. Celui-ci se limite à la conjonction d'expressions conceptuelles (`and`) et la quantification universelle (`all`).

La syntaxe de $\mathcal{FL}_0$ est donnée dans la table 1.6.

C	$\rightarrow$	P
		$\mid$ `Top`
		$\mid$ (`and` C_1 C_2)
		$\mid$ (`all` R C)

TAB. 1.6: *La grammaire du langage de description* $\mathcal{FL}_0$

L'ensemble des connecteurs existant inclut encore des restrictions sur les cardinalités des rôles (`atleast`, `atmost`), des disjonctions de descriptions de concepts (`or`), et la quantification existentielle (`some`), etc. Des listes plus ou moins exhaustives de connecteurs peuvent être trouvées dans [Nebel, 1990; Woods et Schmolze, 1992; Napoli, 1997].

La description d'un concept précise les caractéristiques communes des individus qui sont ses instances. Cette description peut être une définition et dans ce cas on parle d'un concept *défini*. Dans le cas contraire, le concept n'est que décrit, donc *primitif*. La différence entre les deux sortes de concept est qu'une définition fournit des conditions nécessaires et suffisantes pour l'appartenance de l'individu au concept, alors qu'une description ne fournit que des conditions nécessaires.

Une terminologie consiste en un ensemble d'introductions de concepts. Les concepts sont nommés, leur noms étant pris dans un ensemble de noms A. Le nom d'un concept est lié à sa description soit par un symbole $\doteq$ signifiant que c'est une définition et donc que le concept est défini, soit par $\dot{\leq}$ qui introduit un concept primitif. La terminologie T est donc composée par des expressions du type :

A	$\dot{\leq}$	C
A	$\doteq$	C

Ci-dessous, un exemple de terminologie est donné impliquant trois concepts : `HOMME`, `MUSICIEN` et `PERE-DE-MUSICIEN`. Les deux premiers sont primitifs, alors que le dernier est défini. Il est défini moyennant le rôle fils, sa description peut être interprétée comme « un homme dont tous les fils sont musiciens est un père de musicien ».

```
HOMME                     ≤̇   Top
MUSICIEN                  ≤̇   HOMME
PERE-DE-MUSICIEN          ≐   (and
                                HOMME
                                (all fils MUSICIEN))
```

Il est possible, dans certains systèmes (voir par exemple [Nebel, 1990]), de donner une description à un rôle, par exemple à l'aide de connecteurs comme `androle`. Cela permet de distinguer entre rôles primitifs et rôles définis.

Langage assertionnel Le langage assertionnel décrit la connaissance individuelle et factuelle. Il permet de formuler des assertions faisant apparaître les termes du langage terminologique. Ainsi, un individu est décrit par ses appartenances à un ou plusieurs concepts de la terminologie et par les rôles qui le lient à d'autres individus. Dans le langage assertionnel, les concepts sont cités en tant que prédicats monadiques portant sur les individus, alors que les rôles sont cités en tant que prédicats dyadiques.

Une base assertionnelle est constituée par des descriptions d'objets D_o et des descriptions de rôles D_r. Celles-ci sont construites à partir d'un ensemble de noms d'objets `O` et en évoquant les termes du langage terminologique sous-jacent. La syntaxe d'un langage assertionnel est donnée dans la table 1.7.

D_o	$\rightarrow$	*(C o)*
D_r	$\rightarrow$	*(R o o')*

TAB. 1.7: *La grammaire du langage assertionnel*

Un exemple de base assertionnelle minimale conforme à la terminologie du paragraphe précédent est donnée ci-dessus. Elle comporte trois assertions impliquant les deux objets `Paul` et `Jacques`.

```
(HOMME Paul)
(MUSICIEN Jacques)
(fils Paul Jacques)
```

Sémantique

La sémantique d'un langage terminologique est donnée moyennant une interprétation $\mathcal{I} = \langle D_{\mathcal{I}}, .^{\mathcal{I}} \rangle$ où $.^{\mathcal{I}}$ est une *fonction d'interprétation* vers un ensemble $D_{\mathcal{I}}$ dit *domaine d'interprétation*. La fonction $.^{\mathcal{I}}$ associe à chaque concept atomique un sous-ensemble de $D_{\mathcal{I}}$ et à chaque rôle

$\mathcal{I}$(`Top`)	$:=$	$D_{\mathcal{I}}$
$\mathcal{I}$(`and` C_1 C_2)	$:=$	$\mathcal{I}(C_1) \cap \mathcal{I}(C_2)$
$\mathcal{I}$(`all` R C)	$:=$	$\{d \in D_{\mathcal{I}} \mid \forall e \in D_{\mathcal{I}} : (d,e) \in \mathcal{I}(R) \Rightarrow e \in \mathcal{I}(C)\}$

TAB. 1.8: *L'interprétation des termes structurés*

atomique une relation binaire sur $D_{\mathcal{I}}$. La fonction d'interprétation est étendue sur les expressions du langage de manière inductive comme le précise la table 1.8.

Pour une expression de la terminologie T, sa satisfaction par rapport à $\mathcal{I}$, notée $\models_{\mathcal{I}}$ est établie comme suit :

$\models_{\mathcal{I}}$ A $\dot{\leq}$ C	$\Leftrightarrow$	$\mathcal{I}(A) \subseteq \mathcal{I}(C)$
$\models_{\mathcal{I}}$ A $\doteq$ C	$\Leftrightarrow$	$\mathcal{I}(A) = \mathcal{I}(C)$

L'interprétation $\mathcal{I}$ est un modèle de T si, pour toute expression e de la terminologie, $\models_{\mathcal{I}} e$. L'hypothèse du monde ouvert est faite, c'est-à-dire que tout fait n'étant pas explicitement nié (dans la A-Box) est potentiellement vrai. Ainsi, des déductions valides peuvent être faites à partir des connaissances exprimées par la T-Box dont les résultats complètent la A-Box.

Raisonnement

La relation de généralité est définie entre couples de termes C_1 et C_2 vis à vis d'une terminologie T. Ainsi, C_2 subsume C_1 (notée $C_1 \preceq_T C_2$) si et seulement si dans tout modèle $\mathcal{I}$ de T, $\mathcal{I}(C_1) \subseteq \mathcal{I}(C_2)$. Le test de subsumption est à la base d'un ensemble d'autres opérations nécessaires pour le raisonnement terminologique. En particulier, la subsumption permet de vérifier la consistance d'un concept ($C \not\preceq_T \perp$), l'équivalence ($C_1 \approx_T C_2$) ou la disjonction entre deux concepts ((`and` C_1 C_2) $\preceq_T \perp$).

La classification permet d'établir les relations et les dépendances entre le nouveaux concept et les concepts existants dans la hiérarchie compte tenu de la subsumption [Napoli, 1992]. Elle ne concerne que les concepts définis, les concepts primitifs ne permettant pas de faire des inférences complètes et correctes.

Synthèse

L'intérêt des logiques de descriptions réside dans la possibilité de définir des termes structurés (concepts et rôles) et de calculer, de manière efficace, la subsumption entre ces termes. Le mode d'expression ici est *relationnel*, la seule description d'un individu est composée de tous les prédicats dans la A-Box dont son identificateur est argument. La distinction syntaxique est faite, comme pour les graphes conceptuels, entre la connaissance universellement valide et la connaissance factuelle. Cette dernière englobe les connaissances sur les individus et certaines connaissances sur des groupes d'individus.

1.3 L'approche objet en représentation

À l'origine, les langages de représentation par objets se sont constitués en tant que tentative d'enrichir le modèle des frames avec de nouvelles idées, en particulier en provenance de la programmation par objets. Les premiers langages, qualifiés de *post-frames* dans [Euzenat, 1998], préservent l'utilisation de facettes, mais imposent une plus grande discipline dans cette utilisation en limitant la variété initiale. À côté, ils adoptent une distinction générique-spécifique qui se traduit par la division des unités de description, les frames, en classes et instances. Parmi les SRCO on pourra distinguer les systèmes KRL [Bobrow et Winograd, 1977], FRL [Roberts et Goldstein, 1977], SHIRKA [Rechenmann *et al.*, 1990], ROME [Carré, 1989], OBJLOG [Dugerdil, 1988], YAFOOL [Ducournau, 1988]... Certains systèmes, qualifiés de *hybrides* dans [Masini *et al.*, 1989] permettent l'envoi de messages, l'appel à des fonctions ou d'autres mécanismes propres à des paradigmes aussi variés que la programmation logique, fonctionnelle, par objet.

Actuellement, la gamme des divers SRCO est difficile à cerner par des caractéristiques universelles et communément admises. Toutefois, il est possible de donner un aperçu d'un système à objets par contraste avec autres formalismes précédemment introduits. Ainsi, les moyens d'exprimer les connaissances, désignés par langage de description, s'articulent autour de la notion d'objet structuré, classe ou instance.

Ensuite, les SRCO offrent un ensemble de mécanismes d'exploitation particuliers : l'attachement procédural, le filtrage, la classification, etc. Dans la suite nous allons nous arrêter d'abord sur la structure des objets de représentation (section 1.3.1) et les relations entre objets 1.3.2, puis sur les divers mécanismes d'exploitation 1.3.3. En dehors de la compréhension du domaine, la présentation générique de l'approche objet apporte un point de repère pour l'examen de TROEPS, le SRCO qui sert de cadre pour notre étude décrit dans la section suivante.

1.3.1 Principes de représentation

Le langage de description des connaissances dans un SRCO distingue deux types d'unités syntaxiques : les instances représentant des individus du domaine modélisé et les classes représentant des groupes d'individus ou des catégories de ce domaine. Les deux types ont une structure similaire : un ensemble d'attributs auxquels un nom est affecté. De plus, chaque objet possède un lien qui le connecte à une unité plus générale (`est-un` pour les instances et `sorte-de` pour les classes). Les classes sont organisées dans une structure hiérarchique, ici appelée la taxonomie, selon leur généralité.

Malgré les parallèles évidents entre un langage de RCO et un langage de programmation orientée-objet (LPOO), les buts poursuivis par l'un et par l'autre sont différents : le second est destiné à l'écriture de programmes, alors que le premier offre des moyens de gestion des connaissances en particulier il supporte des mécanismes de raisonnement.

Classes

Les classes représentent des catégories du domaine modélisé. Par exemple, au sein d'une base de connaissances l'ensemble des personnes sera représenté par une classe *humain*. À l'instar des classes en LPOO, une classe a pour fonction la génération d'instances. En ce sens elle peut être interprétée comme un moule : sa structure (la liste d'attributs) est une description de la structure de ses instances. Celles-ci sont créées par référence à la classe et lui sont attachées par un lien d'appartenance explicite, `est-un`. La description d'un attribut de classe délimite les valeurs que cet attribut peut prendre au sein d'une instance de la classe. De plus, l'ensemble de descriptions d'attributs fournit des conditions d'appartenance que les instances de la classe doivent satisfaire. La classe a donc deux aspects complémentaires : sa description — encore appelée *intension* — composé par la description de ses attributs et l'ensemble de ses instances ou l'*extension* de la classe.

Les classes sont organisées en hiérarchies par la relation de *spécialisation*. La spécialisation entre deux classes distingue la sous-classe, la plus spécifique et la super-classe, la plus générale. La spécialisation est une relation d'ordre partiel (réflexive, antisymétrique et transitive). Au niveau de la description d'une classe, elle est exprimé par le lien `sorte-de` qui lie une classe à sa super-classe directe. La spécialisation se traduit par une inclusion entre extensions : les instances d'une classe sont également instances de toutes ses super-classes. Concernant l'aspect intensionnel, la description d'une sous-classe affine celle de la super-classe. Cet affinement consiste en l'ajout des nouveaux attributs dans la description de la sous-classe et/ou la spécification de descriptions plus restrictives pour les attributs déjà existant dans la super-classes. Les classes au sein d'une base de connaissances à objets sont la plupart du temps organisées dans une hiérarchie unique dont la racine est une classe universelle, appelée Objet. Sur la figure 1.3, la classe `Adulte` est une sous-classe de la classe `Humain` qui à son tour est une sous-classe de `Objet`. `Adulte` reprend la structure de sa super-classe (attributs `nom` et `âge`) et l'enrichit par l'ajout d'un nouvel attribut, `conjoint`.

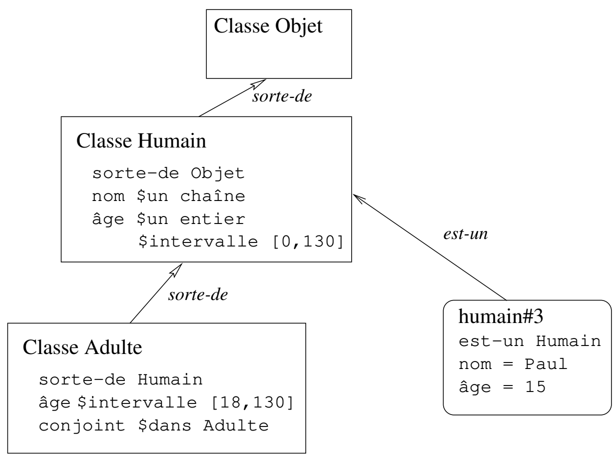


FIG. 1.3: *Exemple de deux classes et d'une instance.*

Instances

Un *objet instance*, désormais appelé simplement *objet*, représente une entité individuelle et identifiable du domaine : un objet physique, un état, une idée, etc. En tant que structure de données, un objet consiste en une description, ensemble de couples attribut valeur, à laquelle un identificateur est associé. Par exemple, une personne particulière est représentée par une instance de la classe `Humain`. Sur la figure 1.3, l'instance `humain#3` représente une personne nommée Paul. L'objet correspondant possède tous les attributs de sa classe auxquels des valeurs sont affectées.

Les attributs de l'objet modélisent les traits pertinents de son correspondant dans le domaine, sa *dénotation*. L'ensemble d'attributs de l'objet sont fixés par sa classe d'appartenance à laquelle il est lié par le lien `est-un`. Toutefois, certains attributs peuvent ne pas être renseignés (instance incomplète). Un attribut renseigné d'un objet ne peut avoir d'autres valeurs que celles autorisées par la description de l'attribut au sein de la classe. Sur l'exemple ci-dessus, la valeur de l'attribut `âge` est inclue dans l'intervalle indiqué dans `Humain`.

Le lien `est-un` n'est pas toujours fixe, contrairement à l'instanciation dans les LPOO. Ainsi, certains SRCO autorisent un changement de la classe pour un objet. Cela veut dire en particulier que l'objet peut exister indépendamment d'une classe particulière. Le déplacement d'un objet dans la taxonomie est effectué par le mécanisme de *classification* qui détermine la ou les classes les plus spécifiques auxquelles l'objet peut appartenir.

Attributs

Un attribut est l'expression d'une caractéristique d'un seul ou d'un ensemble d'individus du domaine modélisé (selon l'entité qui le possède, un objet ou une classe). Cette caractéristique peut être une propriété qui permet de décrire les individus indépendamment de tout autre individu. Dans ce cas, l'attribut admet des valeurs appartenant soit à un type de données offert par le langage hôte du SRCO (qualifié de type *simple*, par exemple un entier ou une chaîne de caractères) soit à un type plus élaboré comme date, séquence d'ADN, etc. Sur la figure 1.3, ce sont les attributs `âge` et `salaire`. Les relations inter-individu constituent une autre sorte de caractéristiques qui peut être représentée par un attribut. Les relations exprimées sont en général binaires et orientées : l'attribut lie l'objet dont il fait partie de la description à un ou plusieurs autres objets. Dans le cas de la classe `Adulte`, l'attribut `conjoint` décrit une relation entre deux instances de la classe. Une place particulière parmi les relations entre objets est occupée par la relation de composition ou *partie-de*. L'attribut sous-jacent lie un objet *composite* représentant le tout à un ou plusieurs objets *composants* représentant les parties.

La valeur d'un attribut peut donc être soit un objet, soit une valeur d'un type de données. De plus, un attribut peut avoir une seule valeur pour un objet particulier, auquel cas il est *monovalué*. Il peut avoir pour valeur une collection (ensemble, liste, séquence, etc.) de valeurs et donc il est *multivalué*. On rencontre souvent des caractéristiques à valeurs multiples dans les domaines à modéliser.

> *Exemple* Les éléments chimiques ont plusieurs valences possibles. Dans une base de connaissances à objets cette multiplicité peut être représentée par un attribut `valences` dont la valeur serait un ensemble d'entiers, dénotant les différentes valences. Du même, au sein d'un modèle à objets du système solaire, si les planètes et leurs satellites sont modélisés par des objets, alors la classe `Planète` aura un attribut `satellites` dont la valeur sera l'ensemble d'objets chacun représentant un satellite. Un autre exemple est constitué par les acides aminés dont la structure est assimilée à une séquence de nucléotides. Ceux-ci peuvent être vues comme des mots sur un alphabet de quatre lettres {A,G,C,T}[1]. Dans un modèle à objets, la suite des nucléotides pourrait être modélisée par une liste de caractères.

Facettes

Les attributs d'une classe possèdent une structure propre, ils sont décrits par un ensemble de facettes (descripteurs). Les facettes apportent des précisions sur le contenu de l'attribut. Ainsi, elles peuvent servir à délimiter l'ensemble des valeurs admissibles pour l'attribut (dans les instances de la classe), à associer un comportement à cet attribut (lors d'une lecture ou écriture), à spécifier diverses manières d'inférer la valeur de l'attribut lorsque celle-ci n'est pas connue, etc. Les facettes de typage fixent le co-domaine d'un attribut si l'on admet que celui-ci est vu comme une fonction d'une classe vers un ensemble de valeurs. En cas d'un attribut monovalué, c'est la facette `$un` qui donne son type. Si l'attribut est multivalué, une facette `$ens-de` ou `$liste-de` est spécifiée signifiant que la collection est un ensemble ou une liste. En fonction de la nature des valeurs d'un attribut, des facettes peuvent exprimer une énumération de valeurs autorisées (`$domaine`) ou, au contraire, interdites (`$sauf`), une liste d'intervalles (`$intervalle`) ou encore une restriction de cardinalité pour les collections (`$card`). Par exemple, les attributs `nom` et `âge` sont de type chaîne de caractères et entier respectivement. Ils sont introduits dans la classe

1. pour **A**denine, **G**uanine, **C**itosine, **T**hymine

`Humain` de la figure 1.3 à l'aide d'une facette `$un` car tous les deux n'admettent qu'une seule valeur au plus par objet. L'attribut `âge` est également contraint par la classe à des valeurs dans l'intervalle [18 130] par la facette `$intervalle`.

Les restrictions en cas de valeurs objets peuvent être statiques, c'est-à-dire la donnée d'un ensemble de classes auxquelles ces objets doivent appartenir (`$type`) ou dynamiques, par la spécification d'un ensemble de critères que les objets doivent satisfaire (`$filtre`, un filtre évalué dynamiquement).

Un sous-ensemble de facettes décrit des inférences possibles pour la valeur d'un attribut au cas où cette valeur n'est pas précisée. En effet, elle est calculée, dans les limites du possible, en fonction du reste de caractéristiques de l'objet (valeurs d'autres attributs ou des attributs des objets liés). Les principaux mécanismes concernés sont la spécification d'une valeur par défaut (`$défaut`), l'attachement procédural (`$si-besoin`) et le filtrage (`$sib-filtre`).

1.3.2 Relations entre entités de description

Les relations entre les expressions d'un langage à objets sont réglementées par une sémantique dénotationnelle du langage qui mène à une interprétation ensembliste des classes.

Appartenance d'une instance à une classe

Les objets membres d'une classe sont liés à celle-ci par une relation d'appartenance : l'entité représentée appartient à l'interprétation de la classe. Le lien d'appartenance impose des contraintes portant sur la description de la classe et de ses instances. Ainsi, les valeurs d'un attribut dans les instances de la classe doivent être compatibles avec les restrictions spécifiées par les facettes de la classe.

Selon que la description de la classe est considérée comme une condition suffisante ou non, on distingue les classes *définitionnelles* des classes *descriptives* [Euzenat, 1993b]. Cette dualité est l'analogue de la distinction entre concept primitif et concept défini dans les langages terminologiques.

Spécialisation entre classes

Les classes d'une base de connaissances sont organisées hiérarchiquement selon une relation de généralité, la *spécialisation*. La spécialisation est une relation d'ordre partiel. Elle distingue les classes plus spécifiques, les sous-classes des classes plus générales, les super-classes. La spécialisation a une sémantique qui repose sur l'inclusion ensembliste : l'extension d'une classe est incluse dans celle de chacune de ses super-classes.

La spécialisation entraîne des contraintes sur les descriptions des classes : les restrictions de valeurs imposées par une sous-classe sont au moins aussi fortes que celles de ses super-classes. En d'autres termes, l'ensemble des valeurs d'un attribut autorisées par une classe est inclus (pas forcément de manière stricte) dans l'intersection des domaines spécifiés par les super-classes.

La spécialisation structure l'ensemble des classes en une hiérarchie (parfois appelée graphe d'héritage). En général, seule la réduction transitive de la relation de spécialisation est exprimée (lien *sorte-de*). Lorsque la hiérarchie résultante est un arbre (au plus une super-classe), on parle de *mono-spécialisation*, dans le cas contraire on est en présence d'une *multi-spécialisation*.

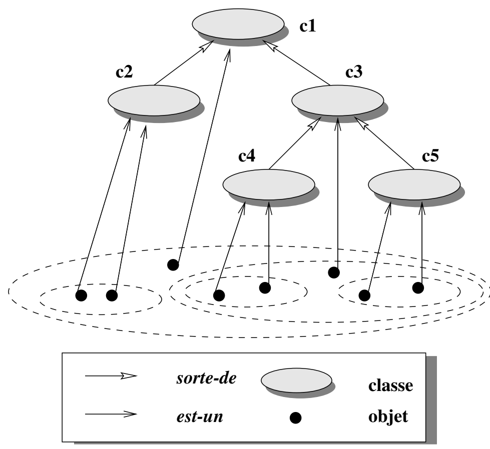


FIG. 1.4: *Exemple de l'effet de la spécialisation sur les extensions des classes : l'extension d'une classe plus spécifique est incluse dans l'extension de la classe plus générale.*

Héritage et spécialisation

La spécialisation est souvent, à notre avis, confondue avec l'*héritage* qui est un mécanisme permettant de profiter de l'organisation hiérarchique des classes pour optimiser le stockage et les modifications des connaissances exprimées. Ainsi, une classe hérite de ses super-classes l'ensemble des attributs que celles-ci définissent avec la totalité des facettes. La classe peut reprendre tel quel un attribut, le masquer ou le raffiner (redéfinir). Du coup, les descriptions des attributs hérités d'une super-classe n'ont pas le même statut : celles héritées d'une super-classe plus spécifique «masquent» les descriptions de ses super-classes. En cas de monospécialisation, l'ordre entre les super-classes d'une classe est total et permet une combinaison entre descriptions héritées sans conflits (sauf inconsistance). En revanche, si l'on autorise une classe à avoir plusieurs super-classes incomparables, on parle d'héritage multiple, la situation est beaucoup plus complexe car des conflits peuvent apparaître entre définitions héritées de super-classes indépendantes. Les problèmes posés par le multi-héritage sortent du cadre de cette thèse, le lecteur intéressé peut se rapporter à [Masini *et al.*, 1989; Ducournau *et al.*, 1995] pour une discussion générale sur la nature de l'héritage.

Relations entre objets

À part les relations dont la sémantique est fixée par le SRCO comme l'appartenance ou la spécialisation, un certain nombre d'autres relations peuvent être exprimées entre les entités de représentation. Celles-ci sont en général des relations binaires. Par exemple, sur la figure 1.3, la relation conjugale entre deux personnes est exprimée à l'aide de l'attribut `conjoint` de la classe `Adulte`. Pour une instance donnée la valeur de l'attribut indique l'instance ou l'ensemble d'instances avec lesquelles cet instance est en relation. Au niveau de la classe, un tel attribut, on parle de *lien* de la relation, désigne la classe qui est le co-domaine de la relation, c'est-à-dire la classe à laquelle les valeurs de l'attribut doivent appartenir. Un exemple de relation entre deux classes est donné sur la figure 1.5. Les classes `Homme` et `Femme` héritent l'attribut `conjoint` de la classe `Adulte` qu'elles raffinent. Chacune des deux classes joue le rôle de domaine pour l'attribut qu'elle définit et de co-domaine pour l'attribut défini dans l'autre classe.

Cela introduit la problématique plus générale de la représentation de deux relations inverses,

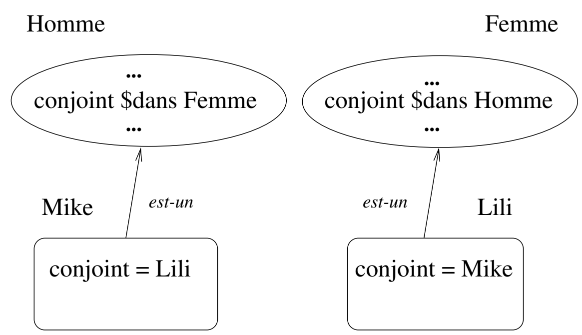


FIG. 1.5: *Exemple de deux classes liées par un attribut lien. La relation* `conjoint` *désigne pour une instance d'une des classes, l'instance de l'autre classe qui est en relation conjugale avec celle-ci.*

c'est-à-dire des relations R et R' pour lesquelles lorsqu'un objet o' est la valeur de R dans un autre objet o, l'objet o est systématiquement désigné par R' dans o'. Des systèmes comme YAFOOL et OBJLOG+ représentent une relation par les deux liens mutuellement inverses qu'elle induit (facette `$inverse`). L'utilisation d'un attribut est une manière directe pour représenter une relation.

Des systèmes comme SRL, SHOOD [Rieu *et al.*, 1992] ou VIEWS [Davis, 1987] ont adopté une toute autre approche vers la représentation des relations entre individus qui consiste à les modéliser par des objets de type particulier. Ainsi, il devient possible d'exprimer des relations entre plusieurs objets. Qui plus est, les relations peuvent être organisées dans une hiérarchie selon une spécialisation où un mécanisme d'héritage permet le partage des propriétés.

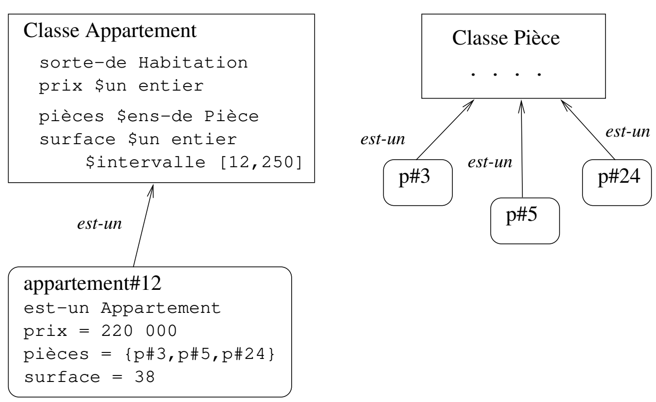


FIG. 1.6: *Exemple de relation de composition entre instances de deux classes.*

Parmi les relations dans les objets, la relation de composition ou *partie-de* occupe une place particulière. Elle lie un objet composite, le tout, à ses parties, les objets composants. Différentes sémantiques peuvent être attachées à la composition, une étude très détaillée sur le sujet peut être trouvée dans [Winston *et al.*, 1987]. Nous signalons ici que dans toutes les formes que la composition peut revêtir, elle demeure une relation : *non-réflexive*, *anti-symétrique* et *transitive*.

Du moment que les divers liens de composition respectent une certaine homogénéité, ce qui est le cas dans la plupart des SRCO qui distinguent la composition, ceux-ci peuvent être combinés. Ainsi, les composants des composants d'un objet composite sont aussi des composants de l'objet.

Il devient donc possible de parler de graphe de décomposition. Ce graphe est acyclique et admet donc un parcours par niveaux.

1.3.3 Mécanismes d'exploitation

Un mécanisme d'exploitation des connaissances fournit des moyens algorithmiques pour la consultation des connaissances disponibles dans une base ou bien pour l'extraction de nouvelles connaissances qui ne sont pas explicitement représentées dans la base. Chaque formalisme de représentation définit un ensemble de mécanismes concrets. Au sein d'un système à objets, les mécanismes principaux qui opèrent sont l'inférence de valeur, l'héritage, le filtrage et la classification.

Les mécanismes d'inférence de valeur permettent de calculer la valeur d'un attribut quand celle-ci n'est pas spécifiée pour un objet donné. Les moyens effectifs de calcul qui peuvent varier fortement de par leur nature (procédures, méthodes, contraintes...) sont rattachés aux classes par le biais des facettes inférentielles des attributs. L'invocation des mécanismes d'inférence se fait à la consultation de la valeur manquante.

L'héritage favorise la factorisation des connaissances au sein d'une taxonomie de classes. Ainsi, les caractéristiques communes aux objets (restrictions sur les valeurs, par exemple) sont placées le plus haut possible dans le graphe, c'est-à-dire dans les classes les plus générales. Ces connaissances sont héritées par toutes les sous-classes et cela de manière dynamique. Par conséquent, les modifications dans une classe se répercutent facilement dans le sous-graphe qui est dominé par celle-ci.

Le mécanisme de filtrage permet la formulation des requêtes sur le contenu d'une base de connaissances : il consiste en la spécification des critères (filtres), puis en la recherche des objets de la base qui les satisfont. Un filtre peut être vu, de par sa description, comme une classe dynamique, à interprétation définitionnelle.

La classification est parmi les mécanismes d'inférence les plus sophistiqués qu'un modèle à objets peut offrir. Son rôle est la complétion d'une base de connaissances par l'introduction d'une nouvelle connaissance : la classification recherche la place de l'entité sous-jacente, instance ou classe, dans la hiérarchie de spécialisation [Napoli, 1992; Ducournau *et al.*, 1999].

Synthèse

Les formalismes à objets offrent un mode de représentation conceptuel car les connaissances relatives à une entité de l'univers du discours sont concentrées à l'intérieur d'une unité du langage de description. Ils combinent la représentation des connaissances purement déclaratives, c'est-à-dire celles qui concernent les descriptions des individus du monde modélisé et des connaissances de nature opérationnelle qui concernent la manière d'obtenir la valeur de certaines caractéristiques. Celui-ci facilite l'inspection des connaissances représentées et son évolution grâce à la spécificité des langages de description des connaissances et aux mécanismes d'exploitation associés.

En conclusion, on doit souligner le rôle primordial de la taxonomie des classes dans la manipulation des connaissances d'une base. En effet, les différents mécanismes d'exploitation s'y appuient dans leur manière de fonctionner.

1.4 TROEPS

Le système de gestion de bases de connaissances à objets TROEPS [Mariño *et al.*, 1990; SHERPA, 1995] est développé dans l'équipe SHERPA, à l'INRIA Rhône-Alpes. TROEPS est un représentant de la RCO dont on présentera les particularités en insistant sur celles liées à la notion de taxonomie et à sa construction. C'est dans ce contexte précis que se place notre étude bien que de nombreux résultats peuvent être appliqués à d'autres SRCO, aux bases de données orientées-objet et, dans une moindre mesure, à certains langages de programmation par objets.

TROEPS est le successeur du système SHIRKA [Rechenmann *et al.*, 1990]. Sa conception a été motivée par le besoin d'apporter une solution à un ensemble de problèmes existant alors dans les SRCO (voir [Carré *et al.*, 1995]). Tout d'abord, la nécessité de **séparer** les connaissances selon leur évolutivité a été ressentie : alors qu'une partie des caractéristiques d'une entité reste immuable, d'autres peuvent changer traduisant ainsi le changement dans le monde ou dans la vision du monde. La solution apportée par TROEPS consiste à partager les connaissances génériques entre le concept et la classe. Un objet est ainsi instance d'un seul concept, mais peut changer de classe, voire en avoir plusieurs simultanément. Le besoin de fixer la signification des classes et des instances à conduit à l'établissement d'une sémantique descriptive du langage. Ainsi, les descriptions des classes en TROEPS ne sont que des **conditions nécessaires** (mais pas suffisantes) pour l'appartenance d'un objet à la classe.

Dans une base de connaissances TROEPS, les connaissances sont organisées à l'aide d'abstractions du langage de description qui ont une granularité variable : concepts, classes, objets, attributs, descripteurs. Dans la suite, nous décrivons le modèle d'objet que ces abstractions constituent dans TROEPS.

1.4.1 Présentation intuitive du modèle d'objets

Une base de connaissances TROEPS contient l'expression des connaissances sur un domaine particulier, tel la biologie moléculaire ou la synthèse organique, et constitue ainsi (une partie d') un modèle du domaine. La partie qui nous concerne est l'ensemble des descriptions des entités qui apparaissent dans le modèle.

Concepts

Les objets d'une base de connaissances, TROEPS, sont partitionnés en un ensemble de *concepts*. Chaque concept représente une famille d'individus du monde modélisé qui partagent l'ensemble de leurs caractéristiques (exprimées moyennant les attributs des objets). Ainsi, le concept est muni de prérogatives ontologiques (au sens propre du terme) sur l'ensemble de ses instances car il défini leur structure, leur intégrité et leur identité. Un objet appartient à exactement un concept de la base.

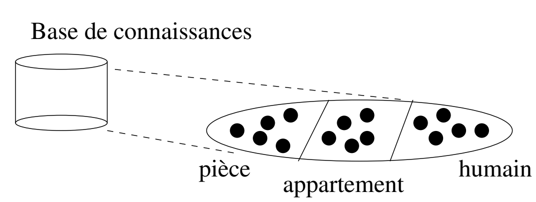


FIG. 1.7: *Exemple d'une base immobilière composée par trois concepts :* `humain`, `appartement` *et* `pièce`. *Les concepts sont des familles d'objets disjoints.*

Exemple Dans une base de connaissances portant sur le domaine immobilier, trois sortes d'éléments du domaine peuvent apparaître (voir figure 1.7) : les personnes sont modélisées comme des instances du concept `humain`, les habitations regroupées dans le concept `appartement` et les pièces des habitations, instances du concept `pièce`

Le concept détermine la structure des objets par la donnée de l'ensemble de leurs attributs et des types de ces attributs. Ces attributs et leur définition au sein du concept seront référés dans la suite par *attributs de concepts* par opposition aux *attributs de classe*. Le concept constitue l'espace des noms pour les attributs, les objets et les points de vue. En effet, au sein du concept, tout nom est unique et deux entités portant le même nom sont considérées comme identiques. En dehors des aspects structurels, le concept détient des connaissances opératoires car une partie de mécanismes d'inférence y sont rattachés.

Attributs de concept

Un concept fixe l'ensemble des attributs qui décrivent les instances du concept. Chaque attribut défini par le concept s'applique à toutes les instances de celui-ci. Ainsi, l'attribut a peut être vu comme une fonction qui associe à tout objet une (ou un ensemble de) valeur(s). Cette fonction a pour domaine le concept de définition, soit k et donc $dom(a) = k$, et pour co-domaine l'ensemble de valeurs possibles, $cod(a) = D$. Pour un attribut donné, sa description au sein du concept contient des informations précisant son *type*, sa *nature* et son *constructeur* qui sont mises à profit lors de diverses opérations de manipulation des objets telles que la vérification des types, la gestion de contraintes d'intégrité et le calcul de la spécialisation et la classification d'objets et de classes.

Le type d'un attribut $\tau(a)$ décrit l'ensemble de valeurs possibles que l'attribut peut prendre dans les instances du concept, c'est-à-dire le co-domaine de la fonction sous-jacente. De plus, il définit la structure de cet ensemble (ordre entre valeurs, ensemble de groupes de valeurs significatifs). Finalement, un certain nombre d'opérations et de relations sur les valeurs est associé à chaque type. Le type d'un attribut peut être soit un concept Troeps, soit un type de données simple T, auquel cas ses valeurs n'appartiennent pas au modèle d'objets. Pour harmoniser la gestion de types et la rendre extensible, Troeps ne comporte pas de types primitifs : tout type de données est importé de l'extérieur. Ses valeurs sont introduites et manipulées moyennant une interface comportant des prédicats de comparaison (appartenance, identité, ordre, etc.) et des fonctions utilitaires (lecture, écriture, etc.) appelée *type de données abstrait*[2] ou ADT. De cette manière de nouveaux types peuvent être intégrés facilement au système et gérés par le système des types [Capponi, 1997] de manière uniforme.

La nature d'un attribut offre des précisions d'ordre sémantique sur la manière dont les valeurs de l'attribut caractérisent les objets sous-jacents. Les attributs de nature `propriété` modélisent des caractéristiques propres à l'individu. Ils sont de type simple, c'est-à-dire que leurs valeurs ne sont pas des objets, mais des valeurs d'un type de données. Les attributs décrivant des relations entre les individus du monde modélisé sont de nature `lien` ou `composant`. Leur type est un concept et donc leurs valeurs sont des objets.

Une place particulière est réservée à la relation de composition entre objets car elle joue un rôle important dans la création des objets et le maintient d'intégrité d'une base.

2. car indépendante de l'implantation des primitives.

Le constructeur d'un attribut est une information relative au nombre de valeurs que l'attribut peut prendre dans un objet. Ainsi, un attribut monovalué ne peut avoir qu'une valeur unique dans un objet particulier. Ce type d'attributs est construit à l'aide du constructeur `un`. Par opposition aux monovalués, les attributs multivalués ont pour valeur une collection de valeurs de leur type de base. Dans ce cas, la fonction représentée par l'attribut est une fonction ensembliste. Selon que la collection de valeurs est munie d'une structure séquentielle (l'exemple des nucléotides) ou non (les valences d'un élément), le constructeur de la collection est `liste` ou `ensemble`.

Point de vue de concept

Un concept de la base de connaissances peut être manipulé sous diverses perspectives qui traduisent des aspect complémentaires de ce concept. La notion de perspective permet de mettre l'accent sur un sous-ensemble des caractéristiques des instances du concept, alors que le maintient de multiples perspectives sur le même concept rend possible l'expression de visions alternatives sur le domaine modélisé, propres à des spécialistes indépendants.

Une perspective est modélisée par un *point de vue* de concept. Celui-ci définit un ensemble d'attributs visibles, ainsi qu'une manière spécifique d'organiser les instances du concept en classes. Ainsi, un concept se voit associer un ensemble de points de vue chacun muni de sa propre taxonomie de classes.

La classe racine de la taxonomie contient la totalité des objets du concept et est progressivement spécialisée en sous-classes plus restreintes.

La définition d'un point de vue comporte un ensemble d'attributs pertinents, sous-ensemble des attributs du concept. Seuls ces attributs sont visibles, le reste demeurant masqué. En cela les points de vue sont similaires aux vues des bases de données relationnelles. Toutefois, l'intérêt d'une vue est dans l'agrégation de plusieurs tables et la mémorisation des calculs sur les colonnes de ces tables. Quant aux points de vue, ils ont le mérite de permettre la cohabitation de connaissances de provenance variée.

> *Exemple* Un ensemble d'appartements peut être organisé en classes selon plusieurs critères. Un des critères peut refléter l'attrait d'un appartement pour un éventuel acheteur, un autre critère l'intérêt potentiel pour une location, alors qu'un troisième critère peut résumer les risques en vue d'une assurance immobilière. Sur la figure 1.8 le concept `appartement` est visualisé avec trois de ses points de vue : `prix`, `taille` et `localisation` (représentés comme des plans). Chaque point de vue est muni d'une taxonomie de classes. L'unique instance du concept visualisée est liée à ses classes d'attachement sous chaque point de vue. Par exemple, il est membre de la classe `bon-marché` sous le point de vue `prix`, et de `très-petit` sous `taille`.

Classes

Les classes TROEPS décrivent des sous-ensembles significatifs d'instances d'un concept. Ainsi, une classe est associée à un concept TROEPS unique. Les classes décrivent les caractéristiques communes des objets membres : plages de valeurs pour les attributs, dépendances entre les valeurs de différents attributs (contraintes), etc. Ces connaissances ne sont pas intrinsèques à un objet particulier et peuvent évoluer dans le temps. La notion de significativité pour un groupe d'objets étant fortement contextuelle (il n'y a pas de classes universellement valides), plusieurs structures

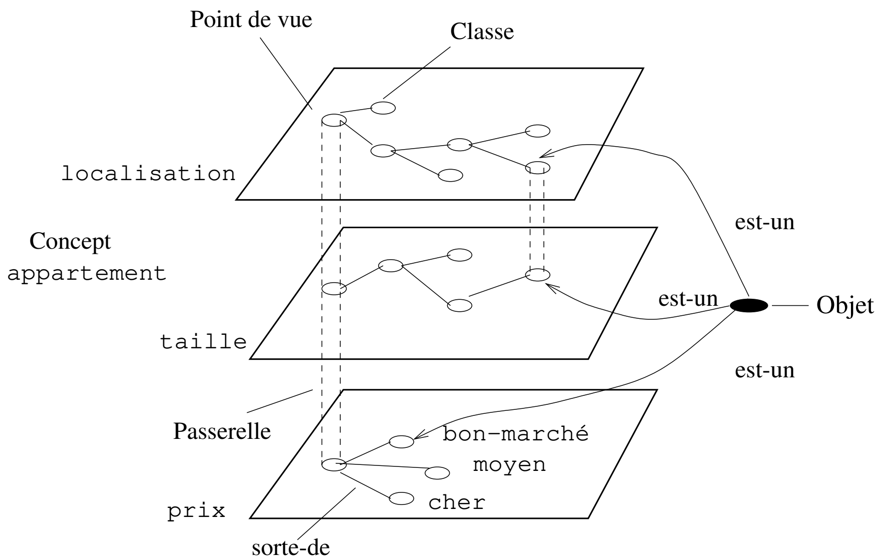


FIG. 1.8: *Exemple de trois points de vue sur le concept* `appartement` *:* `prix`*,* `taille` *et* `localisation`*. Une instance du concept est visualisée avec ses liens d'attachement sous chaque point de vue.*

de classes peuvent coexister sur les mêmes objets. En TROEPS, les classes sont associées à un point de vue unique sur le concept sous-jacent.

Vue sous un aspect structurel, la classe est un couple constitué d'un ensemble d'identificateurs (ceux des instances) et d'une *description*. La description, comme celle des instances, fait apparaître un ensemble d'attributs de classe. Ceux-ci constituent un sous-ensemble des attributs du concept sous-jacent mais possèdent une structure propre qui est détaillée dans le paragraphe suivant.

> *Exemple* La description de la classe `moyen` de la figure 1.8 pourra comporter les attributs `nb-pièces` et `surface`.

La description d'une classe a un statut descriptif et non définitionnel vis-à-vis des instances potentielles de celle-ci. Ainsi, les attributs de la classe ne définissent que des conditions nécessaires pour l'appartenance d'un objet à la classe. Par conséquent, l'appartenance d'un objet à une classe ne peut être calculée automatiquement, elle doit être précisée par le concepteur d'une base de connaissances.

Attributs de classe

Un attribut de classe définit des restrictions sur les valeurs de l'attribut de même nom dans les objets membres de la classe. Les restrictions sont décrites en termes d'un langage de description des types. Chaque restriction s'interprète comme un ensemble de valeurs admissibles appartenant au type de l'attribut, défini dans le concept. L'ensemble effectif des valeurs admissibles pour l'attribut dans une classe donnée est obtenu en combinant les restrictions sur cet attribut définies par la classe et par chacune de ses super-classes. TROEPS n'admet pas d'exception.

Le langage des descriptions des types fournit un ensemble de descripteurs. Ceux-ci précisent des restrictions de nature différente sur les valeurs. Les divers descripteurs sont applicables en fonction du type et du constructeur de l'attribut. Leur liste exhaustive est donnée dans la table 1.9. Ici, v dénote une valeur d'un type de données, r est un entier non négatif et c un nom de classe.

nom	valeur	fonction
`domaine`	$\{v_1, ..., v_n\}$	introduit une liste de valeurs d'un type de données non-ordonné
`type`	$\{c_1, ..., c_n\}$	introduit une liste de classes d'un concept appartenant à des points de vues disjoints
`intervalles`	$\{[v_1 v_1']...[v_n v_n']\}$	introduit un ensemble d'intervalles d'un type de données ordonné
`card`	$[r\ r']$	introduit un intervalle dont la cardinalité des collections d'un type construit doit être comprise
`sauf`	$\{v_1, ..., v_n\}$	introduit un ensemble de valeurs interdites

TAB. 1.9: *Ensemble des descripteurs du langage des types*

Exemple La classe `de-luxe` sous le point de vue `confort` dans `appartement` peut spécifier les contraintes suivantes pour les attributs nb-pièces, loyer et pièces (le point ou l'indirection $u.a$ est une notation standard dans les formalismes d'objets qui est à lire comme «l'attribut a de l'unité de représentation u») :

```
grand.nb-pièces   =   intervalles = { [4 12] }
grand.loyer       =   intervalles = { [5.5 8.8] }
grand.pièces      =    type = { grande, climatisée}
```

Passerelles

Les passerelles sont un moyen d'établir des liens entre les classes appartenant à des points de vue différents sur le même concept. Elle sont composées par un ensemble de classes de départ et une classe d'arrivée. Leur signification est la suivante : tout objet qui appartient à l'ensemble des classes de départ, appartient nécessairement à la classe d'arrivée.

Les passerelles peuvent être uni- ou bi-directionnelles.

Objets

Les objets TROEPS pris comme des structures de données partagent une grande partie de leurs caractéristiques avec les objets de programmation. En effet, un objet peut être vu comme un ensemble de couples attribut-valeur muni d'une identité. En TROEPS, l'identité est établie grâce à une clé. L'ensemble des attributs possédés par l'objet, dont la clé constitue un sous-ensemble, est défini au sein de l'unique concept dont l'objet est instance. Les valeurs des attributs clés permettent l'identification de l'objet à l'intérieur du concept.

Un objet est rattaché à une seule classe plus spécifique sous chaque point de vue de son concept (voir figure 1.8). L'objet est membre de (appartient à) ces classes, ainsi que toutes leurs super-classes. Pour chaque classe d'appartenance, les valeurs des attributs de l'objet appartiennent aux plages spécifiées par les attributs de la classe.

1.4.2 Formalisme objets de Troeps

Après cette description des éléments constitutifs du langage de Troeps, nous présentons ci-dessous une vision plus formelle. Elle s'inscrit dans une mouvance au sein du domaine de la RCO dont les objectifs incluent la formulation d'une sémantique dénotationnelle pour les langages d'objets (voir [Ducournau, 1998; Crampé et Euzenat, 1996]).

Syntaxe

Les unités du langage de description sont des concepts, des classes et des objets. Chacune des trois sortes possède des attributs. Les attributs des concepts sont typés, les attributs des classes précisent des domaines de variabilité, alors que les attributs des objets sont valués. Les valeurs peuvent être uniques ou multiples (collections de valeurs construites à l'aide d'un constructeur), des valeurs simples d'un type de données ou des objets. Les expressions du langage de description de connaissances sont constituées à partir de cinq ensembles :

- K — un ensemble de noms de concepts k
- C — un ensemble de noms de classes c
- O — un ensemble de noms d'objets o
- A — un ensemble de noms d'attributs a
- T — un ensemble de types qui est externe au modèle de Troeps t

Dans la suite, la notation traditionnelle des langages à objets, le point « . », est utilisée pour dénoter les attributs des entités. Ainsi, $o.a$ signifiera l'attribut a dans l'objet o, du même que $c.a$ l'attribut dans la classe et $k.a$ dans le concept. De plus, les deux constructeurs de collections sont introduits par les ensembles de collections qu'ils permettent de créer sur un type de base τ. Ainsi, l'ensemble des collections de constructeur `ensemble` est noté 2^τ et l'ensemble des collections de constructeur `liste` est noté Λ^τ.

<base de connaissances>	::=	{<assertion>*}
<assertion>	::=	<instanciation> $\mid$ <attachement> $\mid$ <slot-concept> $\mid$ <spécification> $\mid$ <spécialisation> $\mid$ <slot-classe> $\mid$ <slot-classe-card> $\mid$ <slot-instance>
<spécification>	::=	$c \subseteq k$
<spécialisation>	::=	$c \subseteq c'$
<instanciation>	::=	$o \in k$
<attachement>	::=	$o \in c$
<slot-concept>	::=	$k.a \subseteq k \mid t \mid 2^k \mid 2^t \mid \Lambda^k \mid \Lambda^t$
<slot-classe>	::=	$c.a \subseteq$ <domaine> $\mid$ $c.a \subseteq c'$
<slot-classe-card>	::=	<*entier-positif*> $\leq \lvert c.a \rvert \leq$ <*entier-positif*>
<slot-instance>	::=	$o.a = o' \mid o.a = v$
<domaine>	::=	$[v_1\ v_2] \mid \{v_1, ..., v_n\}$

Tab. 1.10: *Syntaxe abstraite du langage de description dans* Troeps

Une base de connaissances est un ensemble fini d'assertions dont le graphe de la relation de spécialisation est sans circuit. La base est bien typée, ce qui veut dire que chaque fois que les assertions suivantes sont présentes :

$$o.a = \omega, \quad c.a \subseteq \sigma, \quad k.a \subseteq \chi,$$

alors les deux assertions suivantes sont présentes aussi :

$$\omega \in \chi, \quad \sigma \subseteq \chi.$$

Sémantique

La sémantique du langage est donnée par rapport à un domaine d'interprétation. Comme dans les cas précédents, une interprétation spécifie un domaine et une fonction qui renvoie les éléments du langage vers les constituants du domaine : éléments, groupes, relations. Les objets sont interprétés par des éléments du domaine, les classes et les concepts par des ensembles d'éléments.

Soit B une base de connaissances dont O, C, K et A sont les ensembles des objets, des classes, des concepts et des attributs respectivement. Soit encore l'ensemble de toutes les valeurs τ. Une *interprétation* $\mathcal{I} = \langle D, .^{\mathcal{I}} \rangle$ est donnée par son domaine D et sa fonction d'interprétation $.^{\mathcal{I}}$. L'interprétation de chacun des ensembles est fixée comme suit :

$$\begin{array}{llll}
\mathcal{I}: & O & \longrightarrow & D \\
& C & \longrightarrow & 2^D \\
& K & \longrightarrow & 2^D \\
& A & \longrightarrow & (D \rightarrow \tau) \\
& & & \mid (D \rightarrow D) \\
& & & \mid (D \rightarrow 2^D) \\
& & & \mid (D \rightarrow 2^\tau) \\
& & & \mid (D \rightarrow \Lambda^D) \\
& & & \mid (D \rightarrow \Lambda^\tau)
\end{array}$$

Les descriptions des domaines s'interprètent comme suit (T est un type quelconque) :

$$\begin{array}{lcl}
\mathcal{I}([v_1 \ v_2]) & := & \{v \in T; \mathcal{I}(v_1) \leq_\tau v \leq_\tau \mathcal{I}(v_2)\} \\
\mathcal{I}(\{v_1, ..., v_n\}) & := & \{\mathcal{I}(v_1), ..., \mathcal{I}(v_2)\}
\end{array}$$

Dans le cas d'un intervalle, la relation $\leq_\tau$ dépend de la nature du type et peut être substituée par l'ordre naturel entre les nombres, mais aussi par l'inclusion entre ensembles ou listes de valeurs ou d'objets.

La fonction $.^{\mathcal{I}}$ est injective sur O ce qui assure l'unicité des noms des objets. Chaque fonction $\mathcal{I}(a)$ est totale sur D. Il est maintenant possible d'établir la satisfaction des assertions d'une base de connaissances relativement à une interprétation fixée. L'assertion α est satisfaite dans l'interprétation $\mathcal{I}$ (noté $\models_\mathcal{I} \alpha$) si une des règles ci-dessous s'applique :

$\models_\mathcal{I}$	$c \leq c'$	$\Leftrightarrow$	$\mathcal{I}(c) \subseteq \mathcal{I}(c')$
$\models_\mathcal{I}$	$c \subseteq k$	$\Leftrightarrow$	$\mathcal{I}(c) \subseteq \mathcal{I}(k)$
$\models_\mathcal{I}$	$o \in k$	$\Leftrightarrow$	$\mathcal{I}(o) \in \mathcal{I}(k)$
$\models_\mathcal{I}$	$o \in c$	$\Leftrightarrow$	$\mathcal{I}(o) \in \mathcal{I}(c)$
$\models_\mathcal{I}$	$o.a = o'$	$\Leftrightarrow$	$\mathcal{I}(a)(\mathcal{I}(o)) = \mathcal{I}(o')$
$\models_\mathcal{I}$	$o.a = v$	$\Leftrightarrow$	$\mathcal{I}(a)(\mathcal{I}(o)) = \mathcal{I}(v)$
$\models_\mathcal{I}$	$k.a \subseteq k'$	$\Leftrightarrow$	$\mathcal{I}(a\|k) \subseteq \mathcal{I}(k')$
$\models_\mathcal{I}$	$k.a \subseteq t$	$\Leftrightarrow$	$\mathcal{I}(a\|k) \subseteq \mathcal{I}(t)$
$\models_\mathcal{I}$	$c.a \subseteq c'$	$\Leftrightarrow$	$\mathcal{I}(a\|c) \subseteq \mathcal{I}(c')$
$\models_\mathcal{I}$	$c.a \subseteq d$	$\Leftrightarrow$	$\mathcal{I}(a\|c) \subseteq \mathcal{I}(d)$
$\models_\mathcal{I}$	$i_1 \leq \|c.a\| \leq i_2$	$\Leftrightarrow$	$\mathcal{I}(a\|c) \subseteq \{M \in 2^D \| i_1 \leq \|M\| \leq i_2\}$

L'opération $|.|$ dans la dernière règle peut être interprétée comme la prise de cardinalité d'un ensemble ou de longueur d'une liste.

Raisonnement

En Troeps, les principaux mécanismes de raisonnement valides sont l'héritage, le contrôle et l'inférence de type et le test d'inconsistance d'une classe. L'héritage permet de calculer les restrictions effectives sur un attribut a dans une classe c. La classe c peut définir de telles restrictions de manière explicite. Toutefois, l'ensemble des valeurs autorisées dans les instances de c, est calculé par rapport aux restrictions imposées par toutes les super-classes de c. Le contrôle de type est un autre mécanisme de raisonnement qui teste la satisfaction des restrictions de type définies dans les classes. Finalement, le test d'inconsistance est destiné à détecter l'absence de modèle d'une base de connaissances. Une base inconsistante est une base qui ne possède pas de modèle, c'est-à-dire que l'ensemble des assertions logiques qui en résulte n'admettent aucune interprétation. La correction du test d'inconsistance a été démontrée dans [Crampé, 1997].

D'autres mécanismes de raisonnement, par example la classification de classes ou d'instances, sont utilisés dans Troeps. Ils n'ont pas le statut d'inférences valides car leurs résultats ne sont pas valides dans tous les modèles. Ils démeurent néanmoins des inférences consistantes car ils possèdent au moins un modèle. Le rôle de ces mécanismes est de suggérer des complétions de la base à l'utilisateur.

1.5 Comparaison entre formalismes

Après avoir exposé les aspects descriptif et inférentiel de plusieurs approches vers la RC, nous proposons une brève synthèse. Celle-ci nous aidera à mettre en évidence les rapports entre formalismes afin de mieux cerner l'utilité des solutions retenues pour le regroupement dans ces formalismes discutées plus loin dans ce mémoire. En même temps, il sera examiné l'intérêt d'emprunter des solutions déjà existantes.

– expression des connaissances individuelles : les entités individuelles

  Les SRCO offrent une expression structurée dans la mesure où l'identité et la description des entités sont regroupées en une seule expression, l'objet. Au sein de cette expression, chaque caractéristique de l'entité donne suite à un attribut. Dans les langages de la LPO et dans les logiques de description, les entités individuelles sont dénotées par des constantes.

La description est composée par l'ensemble des atomes qui font apparaître la constante (un premier argument pour les logiques de description). L'ensemble d'atomes est implicitement lié par des connecteurs conjonctifs. En graphes conceptuels, les individus sont dénotés par des marqueurs individuels. Quand un individu possède une description, celle-ci est un graphe, expression unique.

Le nombre et la nature des caractéristiques des individus en RCO et DL sont déterminés par leur appartenance à une classe/concept. En particulier, les relations à d'autres individus sont fixées. De telles restrictions ne font pas partie des modèles des graphes conceptuels, ni de la LPO. Du coup, ces deux formalismes se prêtent mieux à la description d'entités agrégats composées d'un ensemble d'entités et un ensemble de relations entre celle-ci. Le fait de ne pas devoir fixer l'ensemble des caractéristiques des individus permet de décrire facilement des configurations fort variables, par exemple des positions mutuelles dans l'espace d'un ensemble d'objets physiques (voir Section 1.2.2). Plus généralement, on utilise ces deux formalismes pour la description des situations impliquant des entités en relations diverses.

– rapports entre individus et génériques : comment les individus sont-ils décrits par le générique

  Les connaissances génériques dans les SRCO sont organisées en classes. Les classes définissent, dans certains cas, la structure des objets membres, mais surtout la variabilité dans les valeurs des attributs. Ainsi, les attributs des classes fournissent des domaines de valeurs, qui sont soit des types, soit des classes, pour les attributs des objets. L'instanciation qui lie les objets à leurs classes se traduit donc, sur chaque attribut, par une relation d'appartenance de sa valeur dans l'objet au domaine défini par la classes.

  De même, les concepts en logiques de description caractérisent leurs instances avec l'ensemble des rôles et des restrictions sur ces rôles. Une restriction est la donnée d'un concept pour le co-domaine du rôle. L'instanciation a une interprétation similaire à celle dans les SRCO.

  En LPO (graphes conceptuels), la distinction entre les expressions décrivant une entité individuelle et un groupe n'est pas très nette. Si tout de même, une telle distinction existe, les expressions génériques en LPO, les classes, décrivent les entités membres par l'ensemble de leurs caractéristiques communes, des prédicats avec variables. L'appartenance à une classe équivaut à l'existence d'une substitution qui, quand elle est appliquée aux variables dans la description de la classe, transforme celle-ci en un sous-ensemble de la description de l'individu. Dans les graphes conceptuels, une classe décrit ses membres par un sous-graphe qui est plus général, au sens décrit dans Section 1.2.2, que chacune des descriptions des membres. La classe définit donc une sous-structure commune, avec des possibles variations pour chaque nœud conceptuel. L'appartenance à la classe signifie l'existence d'une projection du graphe de la classe dans le graphe de l'individu.

D'autres notions qui ne seront pas détaillées ici permettent encore de distinguer les formalismes : expressivité/complexité, ensemble d'opérations de manipulation disponibles, types d'héritage. On peut se référer à [Brachman et Levesque, 1985] sur ce sujet.

1.5.1 Conclusion

Dans les familles de langages présentées auparavant, deux modes de représentation se retrouvent : le mode relationnel où les unités de représentation sont des relations entre les entités du domaine représenté et le mode structurel qui rassemble la connaissance sur une entité du monde au sein d'une seule unité. Les langages logiques et les réseaux sémantiques suivent le mode relationnel, alors que les langages à objets s'appuient sur un mode structurel. Les graphes conceptuels et les logiques de descriptions occupent une place au milieu : bien que le mode de base reste relationnel, des composantes structurelles sont incluses dans les modèles respectifs.

Les formalismes logiques rendent possible l'expression de la connaissance à un degré de granularité très élevé. Ils conviennent particulièrement à la représentation de la connaissance sur des domaines peu étudiés et donc peu structurés.

A l'autre extrême, les langages à objets structurés concentrent les informations à propos d'une entité du domaine au sein de la même unité syntaxique. Cette concentration assure un accès rapide aux caractéristiques recherchées, ainsi qu'une efficacité plus grande de toute manipulation liée à l'évolutivité de la connaissance : modification, complétions, etc. Elle est aussi à la source d'une meilleure adéquation entre une expression du langage et sa dénotation dans l'univers.

Les divers formalismes se distinguent par la richesse et la finesse des langages de description des connaissances, par l'efficacité des mécanismes d'inférence intégrés, par leur capacité à faire évoluer la connaissance représentée.

1.6 Conception d'une base de connaissances à objets

Le processus de conception d'une base de connaissances est long et comporte des étapes d'acquisition de connaissances, formalisation, validation. Il a souvent un caractère itératif, avec de multiples remises en cause du travail préalablement effectué. Des méthodologies de conception de systèmes à base de connaissances ont été proposées au sein du domaine de l'Acquisition de Connaissances (AC). Plusieurs méthodologies générales semblent avoir du succès actuellement dont *CommonKADS* [Schreiber *et al.*, 1994] et *PROTÉGÉ-II* [Puerta *et al.*, 1992].

La conception d'une base de connaissances à objets quant à elle, s'articule autour de la définition de la structure hiérarchique de classes. Celle-ci englobe la définition des classes, avec leurs attributs et, éventuellement, méthodes, et l'organisation des classes selon la relation de spécialisation. Des méthodologies générales pour la conception de hiérarchies de classes ont été proposées dans [Brachman *et al.*, 1991] en RCO et dans [Booch, 1994] en programmation par objets. L'opération globale est manuelle, le concepteur de la hiérarchie est donc entièrement responsable du résultat car il est censé refléter sa propre vision sur le domaine d'application.

Dans le cas d'un modèle d'objets comme celui que nous avons présenté précédemment, la conception d'une base de connaissances est quelque peu différente. Bien qu'elle n'est sujet d'aucune formalisation, il est possible d'identifier un certain nombre d'étapes spécifiques (d'après [Euzenat, 1998]) :

1. identification des concepts du domaine et de leurs instances.
2. inventaire des relations entre concepts (attributs à valeurs objets : liens et composants).
3. inventaire des attributs propriétés.
4. définition des points de vue sur chaque concept avec leurs attributs visibles.
5. construction des taxonomies sur chaque point de vue.
6. spécification des liens d'inclusion entre classes de divers points de vue, les passerelles.

Nous nous intéressons à l'étape de conception de taxonomies de classes. Le processus de construction d'une taxonomie nécessite l'identification d'un ensemble de classes significatives (selon la perspective adoptée) qui regroupent les instances déjà existantes dans la base. C'est le point essentiel sur lequel notre modèle diverge d'un modèle traditionnel à objets : les objets existent indépendamment de leurs classes. On en déduit donc que les classes sont créées pour refléter une manière particulière d'organiser les objets, c'est-à-dire les connaissances individuelles.

Notre intérêt est plus particulièrement attiré par les situations dans lesquelles le concepteur n'est pas en mesure de spécifier directement les classes. Cette impossibilité ne signifie pas nécessairement l'absence de toute connaissance sur les classes, il peut par exemple s'agir d'une difficulté d'expression en termes d'un langage formel. Nous faisons donc l'hypothèse que le concepteur est au moins capable de reconnaître une taxonomie qui reflète correctement sa vision.

Il n'y a pas de procédé d'identification de classes que l'on pourra suivre à la lettre afin d'aboutir au résultat désiré ([Masini *et al.*, 1989]). Dans [Booch, 1994], l'auteur suggère de « regarder dans l'ensemble des individus du domaine déjà identifiés » pour trouver des sous-ensembles correspondant aux classes potentielles. Ces sous-ensembles se caractériseraient par une homogénéité interne, c'est-à-dire le partage d'un grand nombre de propriétés entre leurs membres. Au contraire, les sous-ensembles correspondant aux classes disjointes n'ont que peu de caractéristiques en commun. En ce qui concerne la manière concrète de détection des sous-ensembles, il est suggéré de s'inspirer des résultats sur la catégorisation en psychologie cognitive. Nous en déduisons la nécessité d'une étape d'analyse de la collection d'individus. Son but est d'en extraire une organisation taxonomique de classes. L'analyse cherche à détecter des régularités existantes dans les caractéristiques des individus qui peuvent indiquer l'appartenance à une classe commune.

Dans notre modèle, les individus sont déjà décrits sous forme d'objets et donc accessibles à une analyse au sein de la base. Cela rend possible l'application de procédés d'analyse automatique, une solution qui est sans alternative en cas de description complexes ou de collections d'objets de grande taille. Ainsi, notre but sera d'étudier les possibilités d'aide automatique à la construction de taxonomies au sein des objets. Plus concrètement, nous allons définir une stratégie de construction qui comporte, d'une part, des méthodes concrètes de regroupement des objets et d'autre part, des procédés d'interaction avec le concepteur. Les résultats de cette étude seront donc matérialisés sous forme d'un environnement de construction de taxonomies.

Le problème que nous chercherons à résoudre peut être décrit comme suit :

Étant donné :

une base de connaissances avec concepts $\{C_i\}$ *chacun muni d'un ensemble de points de vue* $\{pv_i^j\}$

un ensemble d'objets O *d'un concept* C^*

un point de vue pv^* *avec un ensemble d'attributs visibles* A,

un critère d'homogénéité des groupes d'objets

Trouver :

une taxonomie de classes les plus homogènes

La définition ci-dessus comporte un point essentiel, la donnée d'un critère d'homogénéité qui guiderait la construction automatique des classes. Afin d'aboutir à des classes pertinentes, un critère concret doit refléter au maximum la vision du concepteur sur les objets à regrouper. Le chapitre suivant va mettre en évidence les difficultés qui surgissent lorsque l'on veut définir un tel critère ainsi que construire une taxonomie qui l'optimise. Il va également examiner les divers types de critères qui existent et les procédés de construction que ceux-ci entraînent. Dans la suite, nous établissons un ensemble d'exigences qui guideront la définition de notre stratégie.

Tout d'abord, il est nécessaire que la méthode de construction autorise l'expression de visions divergentes sur la structure des classes recherchées. En d'autres termes, une certaine *configurabilité* est nécessaire : le concepteur doit pouvoir traduire sa vision en termes de paramètres du processus de construction. Ensuite, la question de l'exploration de l'espace des solutions, c'est-à-dire des taxonomies, doit être traitée. Il est rare qu'une construction aboutisse dès le premier coup. Au contraire, le processus est *itératif* : il commence avec une première structure qui est ensuite évaluée et modifiée et cela se répète jusqu'à l'obtention d'une taxonomie satisfaisante. La stratégie doit donc offrir des possibilités de modification, globale ou locale, d'une taxonomie. Finalement, en lien avec le point précédent, le processus de construction ne peut être séparé de la base de connaissances, il doit y être complètement intégré car, d'une part, les objets à traiter sont liés à d'autres objets et ces liens font partie de leurs caractéristiques. D'autre part, les résultats d'une construction peuvent remettre en cause d'autres connaissances de la base.

1.7 Conclusion

Les formalismes de RC sont destinés au stockage et à la manipulation des connaissances dans la machine. Parmi eux, les formalismes à objets apparaissent comme un moyen particulièrement adapté à la représentation des connaissances complexes sur un domaine en cours d'étude. Les unités de représentation de base sont les objets qui dénotent des entités individuelles du domaine. Ils sont décrits en termes d'attributs qui modélisent leurs propriétés et leurs relations à d'autres objets. Les objets sont organisés en classes qui expriment les catégories du domaine. Au sein d'une classe, les objets partagent des caractéristiques, exprimées par les attributs de la classe. Les classes sont organisées en une structure hiérarchique, la taxonomie, selon une relation de généralité appelée spécialisation.

Une BCO s'articule autour de la ou les taxonomies de classes que la majorité des mécanismes d'exploitation utilisent massivement. L'importance de la taxonomie au sein de la RCO en fait un terrain d'application naturel pour des procédés de construction automatique de taxonomies. Avant de passer en revue en certain nombre d'approches vers la construction de classes, voici un ensemble d'exigences qu'une méthode doit satisfaire. Celles-ci sont liées à la nature du processus de construction d'une base de connaissances, mais aussi au modèle de données particulier que les objets constituent.

D'une part, le processus s'inscrit dans un contexte de construction de bases de connaissances. Tout d'abord, la méthode doit être configurable afin de s'adapter à la vision particulière d'un concepteur. Ensuite, elle doit tolérer des remises en cause partielles et totales des résultats, c'est-à-dire de la taxonomie obtenue. Le processus total d'obtention d'une taxonomie est donc une recherche dans l'espace des solutions effectuée par la système et en interaction avec le concepteur. Finalement, pour garder les liens avec les autres éléments de la base, ainsi que pour pouvoir bénéficier des moyens de gestion des taxonomies offerts par le gestionnaire de base de connaissances, il est nécessaire d'assurer une intégration complète de la méthode au sein de ce gestionnaire.

D'autre part, le but de l'opération est de construire des classes homogènes, c'est-à-dire qui se basent sur le partage d'un grand nombre de caractéristiques entre les objets membres. Ces caractéristiques peuvent inclure la totalité des attributs des objets, et en particulier ses relations, ainsi que les attributs multivalués. La méthode doit donc avoir la puissance requise pour le traitement de toute la variété de connaissances que l'on peut exprimer dans le modèle. Afin de faciliter l'interprétation des classes construites, il est également nécessaire d'assurer un traitement uniforme aux diverses sortes d'attributs des objets, ce qui assure également l'extensibilité de la méthode. Finalement, le souci d'efficacité nous obligera d'exiger un coût algorithmique raisonnable.

Le chapitre suivant présente un survol des principales approches vers le problème du regroupement afin d'envisager celles qui peuvent s'appliquer aux RCO.

2

La construction automatique de classes

L'objectif de notre travail est la définition d'une méthodologie d'aide à la construction de taxonomies de classes dans les formalismes à objets. L'élément principal de cette méthodologie est une stratégie de construction automatique de classes. C'est pourquoi nous nous intéressons aux méthodes effectives de construction automatique de taxonomies. De telles méthodes ont été développées au sein de l'analyse de données (AD) en Statistique et au sein de l'apprentissage symbolique automatique (ASA) en Iintelligence Artificielle (IA). Nous situant nous-mêmes dans le cadre de l'IA, nous allons porter notre attention essentiellement sur les méthodes d'apprentissage dites de regroupement conceptuel permettant de construire une taxonomie à partir d'un ensemble d'individus non-classifiés. Ce type d'apprentissage est qualifié de *non-supervisé*, car le système ne bénéficie pas d'indications sur l'appartenance d'un individu à une classe, il découvre les classes lui-même. Il est à opposer à l'apprentissage de concepts qui lui est supervisé, les classes sont déjà connues et le but est de les discriminer. Un ensemble de méthodes en provenance de l'analyse de données, dites de *classification automatique* ont également trouvé une place dans notre exposé à cause de leur lien étroit avec certaines méthodes d'apprentissage, les dernières s'inspirant des premières.

Au long de ce chapitre, l'accent sera mis sur les aspects des méthodes qui sont pertinents dans le contexte décrit par le chapitre précédent : leur configurabilité et leur aptitude au traitement des descriptions complexes. Comme nous allons le voir, aucune des approches considérées ne satisfait la totalité de nos exigences, essentiellement à cause de l'expressivité limitée des langages de description admis, mais aussi à cause d'un mode autonome de fonctionnement privilégié en apprentissage automatique.

Au début, nous décrivons la problématique de la construction de taxonomies de manière automatique (section 2.1). Un ensemble d'aspects des méthodes de construction sont énumérés qui facilitent la présentation de celles-ci (section 2.2). Ensuite, trois approches sont présentées que nous avons distinguées selon leur critère effectif de formation de classes. Ainsi, au sein de l'approche par proximité les similarités inter-individu sont quantifiées et guident la construction des classes (section 2.3). L'approche par généralisation consiste à engendrer un ensemble complet de classes (généralisations des objets) vis à vis d'un langage de description (section 2.4). Finalement, l'approche probabiliste cherche à construire des classes ayant un pouvoir prédictif maximal (section 2.5). Les mérites et les insuffisances des approches par rapport à notre problématique sont discutés (section 2.6) . La discussion se termine par l'ébauche d'une méthode de construction de taxonomies adaptée au contexte des objets complexes (section 2.7).

2.1 Problématique

Avant de présenter les grandes familles de méthodes, il sera utile de fixer un cadre pour leur comparaison. Ainsi, nous précisons, tout d'abord, les notions liées à la construction de taxonomies. Les objectifs poursuivis par la construction d'une taxonomie sont énumérés ensuite, car ceux-ci sont déterminants par rapport à sa construction. Finalement, nous précisons quelques problèmes clés liés à l'automatisation de la construction de taxonomies.

2.1.1 Terminologie

Il y a une grande variété dans l'emploi des termes liés à la construction automatique de classes : classe, classification, concept, catégorie. Dans la suite, nous continuerons de parler d'objets qui sont les individus d'un domaine. Rappelons qu'une *classe* désigne un ensemble d'individus. Elle est liée donc à un ensemble d'objets qui constituent son *extension* connue. Elle peut posséder également une *description* qui résume les caractéristiques communes à ses instances. Les classes sont supposées représenter des *concepts du domaine*, cela explique l'emploi du terme *concept* pour désigner une classe dans certains contextes. La description d'une classe est souvent assimilée au concept.

Une taxonomie est définie par un ensemble de classes $\mathcal{C}$, muni d'une relation de généralité $\leq$ qui est un ordre partiel et d'une fonction $\mathcal{I}$ d'interprétation des classes vers un domaine X quelconque.

Définition 2.1. *Soit $\mathcal{C}$, un ensemble de classes muni d'une fonction $\mathcal{I}$ d'interprétation vers un domaine $\mathcal{X}$ d'individus :*

$$\mathcal{I} : \mathcal{C} \rightarrow 2^{\mathcal{X}}$$

Une taxonomie par rapport à $\mathcal{I}$ est un couple $\mathcal{H} = \langle C, \leq \rangle$ où C est un ensemble de classes, $C \subseteq \mathcal{C}$, et $\leq$, ou « plus général que », une relation d'ordre partiel sur C compatible avec $\mathcal{I}$:

$$\forall c, c' \in C, c \leq c' \Rightarrow \mathcal{I}(c) \subseteq \mathcal{I}(c')$$

Une classe c est donc plus générale qu'une classe c' si l'interprétation de c inclut celle de c'. Cette définition est suffisamment générique pour s'appliquer dans un grand nombre de cas. Dans un contexte de construction automatique, l'interprétation d'une classe est l'ensemble des objets membres.

L'ordre $\leq$ n'étant pas contraint *a priori*, le graphe représentant sa réduction transitive est donc un graphe orienté sans circuit (DAG). Dans la pratique, une taxonomie se présente le plus souvent comme un arbre, c'est le cas des taxonomies biologiques. D'autres formes peuvent être rencontrées : pyramides [Bertrand et Diday, 1990; Brito, 1991], treillis [Ganter et Wille, 1989; Godin *et al.*, 1991] ou DAG [Bisson, 1992; Bournaud et Ganascia, 1995].

Le terme *classification* est employé tantôt avec le sens de taxonomie, tantôt pour désigner le processus de sa construction. Pour garder une consistance avec la terminologie du chapitre précédent, nous allons parler de taxonomie et de construction de taxonomie.

Dans la même veine, former une classe signifiera donner l'ensemble des membres de la classe, son extension. Caractériser une classe, en revanche, fera référence à la description de la classe. Le terme construction sera employé pour désigner le processus global qui aboutit aux classes sous leur forme désirée, c'est-à-dire soit avec leurs extensions seules, soit avec extensions et descriptions. Le processus de construction de classes sera appelé regroupement.

Finalement, nous allons préciser quelques termes courants dans le domaine du regroupement auxquels nous réservons un emploi particulier :

Méthode de regroupement un procédé algorithmique permettant d'aboutir à un ensemble de classes à partir d'un ensemble d'individus, un processus de construction de classes est l'exécution d'une ou plusieurs méthodes ;
Exemples : la classification hiérarchique ascendante, la méthode des plus proches voisins, etc.

Outil de construction un mécanisme de bas niveau qui réalise un seul aspect d'une méthode ; une méthode met en œuvre un ou plusieurs outils ;
Exemples : la distance Euclidienne dans $\mathbb{R}^2$, le cirtère des l'inertie, l'indice du lien minimal, etc.

Approche un ensemble de méthodes qui partagent les mêmes fondéments théoriques (principes, hypothèses, etc.) et par là les mêmes outils de construction ;
Exemples : l'approche par similarité, l'approche probabiliste incrémentale, etc.

Souvent nous parlerons de *technique* pour désigner un sous-ensemble d'une méthode qui est chargé de la résolution d'un problème particulier. Une technique peut aussi bien s'inscrire dans le cadre d'un seul outil que représenter un ensemble de tels outils.

2.1.2 Objectifs d'une taxonomie

La construction de taxonomies est une opération courante en sciences : en médecine pour découvrir des classes de malades ayant les mêmes combinaisons de symptômes, en botanique, pour mettre en évidence des sous-espèces d'une même variété, etc.

Une taxonomie scientifique peut être motivée par deux objectifs principaux [Bournaud, 1996] . D'une part, le but originel de tout processus de regroupement est la **structuration** d'un ensemble de données. Par là, une amélioration de la visibilité (possibilité de **visualisation**) est réalisée. En outre, une taxonomie offre une **description** synthétique des données et donc facilite l'accès vers celles-ci. La connaissance structurée peut appuyer la **découverte** de concepts sur un domaine : les classes présentes dans la collection d'individus peuvent traduire un phénomène ou une loi jusqu'à lors inconnus. D'autre part, la taxonomie peut servir à des fins de **prédiction** : prédire une caractéristique particulière d'un nouvel individu selon sa classe dans la taxonomie.

L'objectif visé par une taxonomie influe sur le processus de sa construction. Celui-ci consiste à identifier les classes et à les caractériser. Intuitivement, on regroupe les individus qui exhibent des similitudes selon un critère. C'est ce critère qui reflète le but de la construction, il a donc souvent un caractère subjectif. La difficulté essentielle d'une automatisation du processus de construction est l'expression du critère de regroupement « approprié ».

2.1.3 Automatisation de la construction

Le problème de l'automatisation de la détection des groupes homogènes a été abordé dans le cadre des Statistiques ainsi que dans le cadre de l'IA. La classification automatique (*automatic classification* ou *clustering* en anglais) en analyse de données (AD) [Jain et Dubes, 1988] et le regroupement conceptuel en apprentissage automatique [Michalski et Stepp, 1983] sont les deux courants qui ont donné naissance à un nombre considérable de méthodes du regroupement. L'approche de l'AD est souvent qualifiée de « numérique » alors que l'approche de l'apprentissage est dite « symbolique ». L'opposition prend ses racines dans la nature des données sur lesquelles les méthodes opèrent : décrites par des variables numériques en AD et par des propriétés symboliques en apprentissage.

Bien que notre problématique se place *a priori* dans le cadre de l'IA, nous nous intéressons aussi bien au regroupement conceptuel qu'à la classification automatique car les méthodes de l'AD ont souvent été reprises et adaptées dans le cadre symbolique.

Il est à noter que des méthodes de regroupement automatique ont été proposées dans d'autres contextes, comme la reconnaissance de formes [Duda et Hart, 1973] et la recherche d'informations [Carpineto et Romano, 1996]. Par ailleurs, le regroupement automatique est, parmi les outils de la découverte de connaissances dans les bases de données [Kodratoff, 1997], une discipline relativement nouvelle dont le but est de mettre en évidence des connaissances non-explicites et potentiellement utiles dans les grands ensembles de données [Piatetsky-Shapiro et Frawley, 1991].

Nous avons évoqué le caractère subjectif du processus de construction de taxonomies qui rend difficile son traitement informatique. D'une part, il est nécessaire de formuler un critère permettant de juger la « qualité » d'une taxonomie. Ceci n'est pas une tâche aisée car la qualité n'est souvent apparente qu'après la construction, lors de l'utilisation de la taxonomie. Par défaut, l'hypothèse est faite que les classes d'un niveau de la taxonomie doivent maximiser la similarité intra-classe, c'est-à-dire entre les membres de la même classe (cohésion) et minimiser les similarités inter-classe (contraste).

D'autre part, une procédure algorithmique pour la recherche de la ou une structure qui optimise ce critère parmi l'ensemble des taxonomies possibles (il y en a de l'ordre de 2^{2^n} pour n objets) doit être fixée. Celle-ci fait souvent usage d'un critère de regroupement qui guide la construction des classes. Il n'est pas toujours possible de distinguer entre le critère de qualité et le critère de regroupement dans une méthode.

Le problème générique du regroupement conceptuel peut être défini ainsi :

Étant donné :

- *un ensemble d'individus* O,
- *un ensemble de caractéristiques* des individus A,
- *des connaissances sur le domaine considéré*,
- *un critère de qualité de la taxonomie* ;
-

Trouver :

- *un ensemble de classes* d'individus,
- *une description pour chaque classe*,
- *une organisation hiérarchique des classes.*

Les deux tâches à effectuer sont donc [Michalski et Stepp, 1983] :

1. construction d'un ensemble de partitions récursives

2. inférence de description pour chaque classe

Ces tâches se retrouvent au sein de chaque système de regroupement conceptuel, bien que les étapes correspondantes ne soient pas forcément disjointes. Elles peuvent être complètement séparées comme dans le processus habituel d'analyse de données effectuées en alternance [Bisson, 1992] ou simultanément [Fisher, 1987; Michalski et Stepp, 1983].

La construction de «bonnes» taxonomies de manière automatique suppose une possibilité d'évaluer la qualité de celles-ci. Toutefois, la qualité d'une taxonomie est fonction du résultat de son utilisation. En effet, d'après Lance et Williams [Lance et Williams, 1967b],

> ...there are no «good» or «bad», «probable» or «improbable» classifications, there are only useful or unuseful ones[3].

L'utilité d'une taxonomie peut se manifester sur plusieurs plans. Par l'ensemble des classes, elle peut contribuer à confirmer ou infirmer des hypothèses sur le domaine considéré, ainsi que suggérer de nouvelles hypothèses. Dans un plan pratique, la taxonomie offre une représentation synthétique et structurée de l'ensemble des données. Elle peut aussi être utilisée à des fins de prédiction. Dans sa thèse [Bournaud, 1996], Bournaud souligne à juste titre le fait que la majorité des méthodes de regroupement conceptuel sont orientées vers la recherche de classes ayant un bon pouvoir prédictif. Il est, en effet, relativement aisé de mesurer la capacité prédictive d'une taxonomie [Fisher, 1996]. Cependant, les taxonomies obtenues par ces méthodes ne reflètent pas toujours la structure de classes sous-jacente dans les cas où le but de la construction n'est pas la prédiction [McKusick et Langley, 1991].

2.2 Aspects d'une méthode de regroupement

La définition d'une méthode de regroupement conceptuel passe par la réponse à certaines questions. Les différents langages utilisés pour la description des connaissances manipulées — les objets, les classes et les connaissances du domaine — constituent un aspect primordial. Les deux autres aspects concernent les tâches de la construction et de la caractérisation des classes. Avant de présenter les approches existantes en regroupement conceptuel, nous essayerons de donner un apperçu de la variabilité que les méthodes exhibent sur chacun des aspects.

Langages de description

Le *langage de description des individus* offre des moyens d'expression pour les caractéristiques des individus. Les langages utilisés en regroupement sont de deux types. Le premier décrit les objets sous forme d'un ensemble de couples attribut-valeur où les valeurs appartiennent à un type de données. Ce type de langages a la puissance de la logique propositionnelle. Le second groupe est composé par des langages autorisant l'expression de relations entre individus. En général, il s'agit d'un sous-ensemble de la logique des prédicats [Bisson, 1992]. Mais les graphes conceptuels ont également trouvé une application dans ce domaine [Mineau, 1990]. Les objets manipulés sont des agrégats composés par un ensemble de parties et de relations entre parties.

Le *langage de description des classes* constitue, en général, une extension du langage des objets. Sa forme exacte reflète déjà une vision sur la nature des classes recherchées.

Un groupe de systèmes utilise des langages dits conjonctifs. Les classes sont décrites par des conjonctions de propriétés interprétées comme des conditions nécessaires et suffisantes. En cas d'un langage attribut-valeur, comme dans le système Cluster/2, les propriétés sont des couples attribut-domaine. Le domaine décrit l'ensemble des valeurs de l'attribut rencontrées dans les objets de la classe. Avec un langage de prédicats, les classes peuvent être décrites par des conjonctions d'atomes prédicatifs [Bisson, 1992].

3. il n'y a pas de taxonomies vraies ou fausses, probables ou improbables, mais seulement utiles ou inutiles.

Les systèmes de *formation de concepts* de la mouvance de COBWEB, (cf. Section 2.5.1) utilisent des descriptions de classes en termes de probabilistes. Une classe, appelée *concept probabiliste*, définit pour chaque attribut une distribution de probabilités sur ses valeurs. Le système LABYRINTH [Thompson et Langley, 1991] qui appartient à cette famille admet des descriptions relationnelles attributives qui sont semblables à celles des objets en représentation de connaissances (RC) vues précédemment. Ainsi, les attributs relationnels dans les classes construites sont décrits par des distributions de probabilités sur un ensemble de classes.

Le choix des langages de description pour les objets et pour les classes est étroitement lié aux autres éléments d'une méthode de regroupement conceptuel, le critère de regroupement et la procédure de construction des descriptions des classes. Dans notre contexte, les deux langages sont fixés et les contraintes liées à leur nature et leur richesse vont nous guider dans le choix d'une méthode appropriée de regroupement.

Finalement, une construction de taxonomie manuelle ne se passe jamais isolément, sans recours à d'autres connaissances. Souvent, les méthodes offrent un formalisme de représentation des connaissances sur le domaine. Celui-ci peut être une variante des langages de description des individus ou des classes comme dans CLUSTER/2 ou un formalisme à part. Par exemple, CLUSTER/S utilise un réseau sémantique pour représenter les objectifs du regroupement et les liens entre les attributs des individus. Dans KBG des règles de production sont utilisées pour représenter les connaissances permettant de compléter les descriptions des individus.

Processus de regroupement

Au sein d'une méthode, le critère de regroupement guide la construction des classes. Soulignons que la majorité des critères existants favorise la recherche de classes *compactes* et *contrastées*. Le critère de regroupement est étroitement lié à l'évaluation d'une taxonomie, souvent le même critère est utilisé pour regrouper et évaluer.

Nous distinguons trois types de critères dans la suite. Les critères *globaux* portent sur un ensemble de classes, généralement constituant une partition d'une sur-classe. Le critère *LEF* (Lexical Evaluation Function) employé par le système CLUSTER [Michalski et Stepp, 1983] est un critère complexe, composé par une liste de critères élémentaires. Parmi ces critères, citons l'ajustement entre les descriptions et les données, la disjonction entre classes, la simplicité de leurs descriptions, etc. Le critère utilisé par le système COBWEB [Fisher, 1987] mesure la compacité et le contraste d'une partition en termes de probabilités. Il se base sur la mesure *Category Utility* issue des travaux en psychologie cognitive [Gluck et Corter, 1985]. Les critères basés sur les ressemblances portent sur un couple d'entités, deux objets, une classe et un objet ou deux classes d'une partition. Un tel critère mesure, sous forme d'une valeur numérique, le degré de ressemblance entre ceux-ci. Finalement, certaines méthodes construisent des taxonomies exhaustives, c'est-à-dire contenant la totalité de classes vérifiant un critère qui lie leurs descriptions aux objets. Nous avons appelé ces critères *exhaustifs*.

Le problème du regroupement peut être vu comme une recherche d'une taxonomie dans l'espace des hiérarchies de classes possibles [Fisher, 1987]. Un ensemble de méthodes utilise des procédés de recherche de nature gloutonne, les taxonomies obtenues ne sont pas remises en cause (voir 2.3.5). Cela n'empêche pas le processus global d'obtention d'une classification d'être itératif. Toutefois, chaque exécution de l'algorithme de construction est indépendante. D'autres méthodes procèdent par une recherche heuristique dite d'*optimisation itérative* dans l'espace des solutions (voir 2.3.2). A partir d'une taxonomie initiale, un certain nombre d'itérations sont effectuées visant à améliorer le résultat.

Une forme particulière de recherche est réalisée par les méthodes *incrémentales* (voir 2.5).

Celles-ci sont capables d'acquérir les individus un par un en intégrant à chaque étape le nouvel individu dans la hiérarchie courante. Grâce à des opérateurs d'intégration spécifiques, le résultat des étapes préalables peut être remis en cause.

Dans la suite, nous allons distinguer les diverses méthodes selon leur manière de former les regroupements.

2.2.1 Caractérisation des classes

L'inférence de descriptions des classes est un processus de généralisation : à partir des description des objets de la classe, il abstrait un ensemble de caractéristiques communes. La construction de descriptions discriminantes à partir d'une collection d'objets appartenant à des classes différentes fait partie de la problématique de l'apprentissage supervisé. Nous n'allons pas détailler ici les différentes méthodes existantes, mais invitons le lecteur à consulter [Mitchell, 1997].

Pour notre étude, il est utile d'examiner les différences existant entre la généralisation dans un langage propositionnel et un langage admettant des relations. Dans le premier cas, la description de la classe se résume à une synthèse, attribut par attribut, de l'ensemble des valeurs rencontrées dans les objets. La complexité du processus varie en fonction des contraintes sur les descriptions des classes (simplicité, disjonction).

Avec des descriptions relationnelles, la généralisation est plus délicate. Rappelons que les objets décrits sont des ensembles de parties liées par des relations. La description d'une classe spécifie les caractéristiques communes qui s'expriment par un ensemble de relations entre parties. Pour extraire un ensemble de telles relations, il est nécessaire d'apparier les parties. Cet appariement est déterminant par rapport à la généralisation obtenue. Pour que celle-ci soit maximalement spécifique, c'est-à-dire qu'elle décrit au mieux les objets, un appariement optimal doit être trouvé. Le nombre d'appariements possibles croît de manière exponentielle avec le nombre des parties des objets. La recherche d'un appariement optimal entre deux structures relationnelles est donc de nature NP-difficile. C'est pourquoi, les systèmes d'apprentissage, aussi bien supervisé que non-supervisé, qui admettent des descriptions relationnelles font appel à des techniques heuristiques pour trouver un appariement convenable lors de la généralisation. Signalons que dans le cas d'un langage exprimant les relations par des attributs, par exemple les objets en RC, la généralisation n'inclut pas forcément la recherche d'un appariement. Celui-ci est fixé par les attributs qui repèrent les parties. Le seul cas où un appariement peut s'avérer nécessaire est le traitement des attributs multi-valués [Ketterlin, 1995]. En effet, la comparaison entre collections nécessite, comme nous allons le voir, un appariement entre leurs membres.

2.3 Regroupement selon des mesures de ressemblances

Dans cette section nous décrivons un ensemble de méthodes dont le critère de regroupement repose sur une mesure de ressemblance. Dans la grande majorité des cas, la mesure est définie sur les couples d'objets et le but du regroupement est de rassembler au sein d'une classe les objets les plus similaires tout en gardant une similarité faible entre les objets de classes différentes.

2.3.1 La notion de mesure de ressemblance

Les *mesures de ressemblance* sont des fonctions qui quantifient les similitudes entre objets. Ainsi, elles associent à un couple d'individus une valeur réelle positive exprimant le degré de leur similitude. Cette *évaluation* par une mesure est en quelque sorte duale à la *caractérisation* des similitudes qui est obtenue par la généralisation [Bisson, 1997].

Il sera question de *similarité* si la mesure associe les valeurs élevées au couples d'objets qui se ressemblent plus, sinon on parlera de *dissimilarité*.

La notion de mesure de ressemblance est inspirée de celle de distance. Les propriétés des fonctions de ressemblance constituent un champ d'étude très actif. Nous citons ici les principaux types de mesure, le lecteur intéressé trouvera une étude complète des diverses fonctions et leurs équivalences dans [van Cutsem, 1994], chapitre 1.

Une fonction d, définie sur un ensemble d'objets O et à valeurs réelles non-négatives, $d : O \times O \rightarrow \mathbb{R}_+$, est un *indice de dissimilarité* si étant donné deux objets o_1, o_2 dans O :

1)	$d(o_1, o_1)$	$=$	0	(minimalité)
2)	$d(o_1, o_2)$	$=$	$d(o_2, o_1)$	(symétrie)

Afin d'obtenir une *métrique*, les propriétés suivantes sont exigées (pour un triplet o_1, o_2, o_3 de O quelconque) :

3)	$d(o_1, o_2)$	$=$	0 implique $o_1 = o_2$	(identité)
4)	$d(o_1, o_2)$	$\leq$	$d(o_2, o_3) + d(o_1, o_3)$	(inégalité triangulaire)

Finalement, si la fonction d vérifie :

5)	$d(o_1, o_2)$	$\leq$	$Max(d(o_2, o_3), d(o_1, o_3))$	(inégalité ultramétrique)

on dira qu'elle est une *ultramétrique*. Diverses combinaisons de propriétés donnent suite à des fonctions différentes : d est seulement un *indice de distance* ou *semi-métrique* si elle vérifie les propriétés 1,2 et 3. et un *écart* ultramétrique si elle vérifie 1,2 et 5.

Toutes ces propriétés ont une interprétation intuitive. Par exemple, la symétrie assure que l'ordre des arguments n'a pas d'importance. Ce fait semble naturel dans le contexte de comparaison entre individus de même statut. L'inégalité triangulaire nous garantit, à son tour, que si pour un triplet d'individus quelconques, soit a, b et c est valide que a est proche de b et b est proche de c, alors a ne peut être très différent de c. Ceci a une application directe lors de la construction des groupes d'individus similaires : lors de l'examen de l'appartenance d'un individu à un groupe, il n'est pas indispensable de tester la similitude de celui-ci avec tout autre membre du groupe. Il suffit de vérifier qu'il est proche d'un seul individu, le plus central. Si l'individu examiné lui est proche, alors on peut l'affecter au groupe sous-jacent car il ne peut pas diverger beaucoup du reste des membres du groupe.

En ce qui concerne la dualité entre les notions de similarité et dissimilarité, le passage entre une mesure de *dissimilarité* d et une mesure de *similarité* s se fait naturellement par une soustraction :

$$s(o_1, o_2) = \max_{o,o' \in O}(d(o, o')) - d(o_1, o_2)$$

Les mesures de dissimilarité sont soit directement *prélevées* sur les individus, soit *calculées* sur leurs caractéristiques, appelées *variables*. Dans la suite de ce chapitre nous nous intéresserons au calcul de dissimilarité. Pour cela, nous supposons que les individus sont décrits par un ensemble de variables, toutes de nature numérique. L'ensemble des individus est introduit sous forme de tableau à deux dimensions où les lignes correspondent aux individus et les colonnes aux variables. Les entrées du tableau sont les valeurs décrivant les individus.

Le calcul nécessite une manière d'évaluer les différences entre les individus sur chaque variable, que nous allons appeler *écarts*, puis une manière de combiner celles-ci en une valeur unique. Les écarts entre valeurs numériques sont calculés par la valeur absolue de leur différence. Afin d'assurer une équité entre les variables à étendue différente, une procédure de *standardisation* sur les variables est effectuée. Celle-ci transforme chaque valeur d'une variable donnée par la soustraction de la moyenne de la variable (*centrage*) et la division par son écart type (*réduction*). Les nouvelles variables sont de moyenne zéro et d'écart-type un.

Un grand nombre de mesures ont été définies au sein de l'AD [Diday *et al.*, 1982]. Parmi elles, nous allons nous intéresser aux mesures de la famille dite de Minkowski. Soit d_p la fonction définie sur un ensemble d'objets O décrit par n attributs de type réel $\{a_i\}_{i=0,n}$ par :

$$d_p(o_1, o_2) = (\sum_{i=1}^{n} |o_1.a_i - o_2.a_i|^p)^{1/p} \tag{2.1}$$

Avec des valeurs du paramètre p de 1, 2 et ∞ on obtient respectivement la distance de Manhattan (simple somme des écarts), la distance Euclidienne classique et la distance de Chebichev (le maximum des écarts). Afin de doter le calcul d'une plus grande flexibilité, indispensable lorsque les caractéristiques ont de comportements disparates vis-à-vis de la similarité entre individus, les contributions des attributs dans la formule 2.1 sont pondérées. Il est admis, que le poids λ_i utilisé sur un attribut a_i exprime le degré de sa pertinence. L'expression de la mesure globale devient donc :

$$d_p(o_1, o_2) = (\sum_{i=1}^{n} \lambda_i |o_1.a_i - o_2.a_i|^p)^{1/p}$$

En général, les poids sont tels que leur somme est égale à un ($\sum_{i=1}^{n} \lambda_i = 1$).

D'autres mesures sur des descriptions purement numériques qui sont souvent employées, sont les distances *quadratiques* (distance de Mahalanobis, de χ^2 [Saporta, 1990]). Des mesures sur des attributs binaires ont également été définies, mais nous ne nous y arrêterons pas. Une description de celles-ci peut être trouvée dans [Saporta, 1990].

Afin d'illustrer la notion de mesure, nous prenons un ensemble de six objets, instances du concept `humain`, décrits par deux attributs : `âge` et `salaire` (voir table 2.1). L'attribut `âge` est de type entier, prenant ses valeurs dans `intervalle [20 35]`, alors que `salaire` est de type réel, variant dans `intervalle [1,8 3,3]` (il indique le revenu mensuel d'une personne en Euros).

TAB. 2.1: *Objets du concept* `humain`

attribut	o_1	o_2	o_3	o_4	o_5	o_6
âge	26	23	27	30	32	35
salaire	3	2,3	2,2	2,7	2,9	3,2

Sur les données de la table ci-dessus, nous allons calculer une distance de Manhattan. Les valeurs des attributs ne seront pas modifiées comme en analyse de données, mais leur contributions seront normalisées, en divisant chacune par l'étendue de l'attribut. Pour le calcul d_p, fixons des poids égaux pour les deux attributs. Ainsi, la dissimilarité pour le couple (o_1, o_4), devient :

$$d_p(o_1,\ o_4) = 0,5\ |26 - 30|/(35 - 20) + 0,5\ |3 - 2,7|/(3,3 - 1,8) = 0,28 \tag{2.2}$$

Les valeurs de la dissimilarité pour l'ensemble des objets sont données dans la table 2.2.

Dans la suite nous décrivons deux groupes de techniques de regroupement basées sur des mesures de ressemblance. Ils se différencient dans la stratégie de construction des classes : dans

TAB. 2.2: *Valeurs des dissimilarités d_p pour les objets de table 2.1*

	o_1	o_2	o_3	o_4	o_5	o_6
o_1	0					
o_2	0,38	0				
o_3	0,36	0,26	0			
o_4	0,28	0,43	0,31	0		
o_5	0,5	0,61	0,46	0,33	0	
o_6	0,56	0,74	0,57	0,44	0,25	0

le premier cas elle partitionne les objets en classes dissemblables, alors que dans le second cas elle consiste à agréger progressivement les objets similaires. Chacune de ces techniques a été reprise et étendue dans le contexte de l'ASA. Nous décrivons brièvement les deux système les plus proéminents des approches descendante et ascendante, CLUSTER et KBG.

2.3.2 Classification par partitionnement

Les méthodes de classification par partitionnement divisent l'ensemble O en k classes où k est un paramètre externe de la méthode.

Dans la méthode dite *des centres mobiles* [Forgy, 1965], le processus de formation des classes est fondé sur la notion de *noyau* : chaque classe est formée autour de son noyau. Au départ, un ensemble initial de noyaux est déterminé. Ensuite, des étapes de partition et de sélection des nouveaux noyaux se suivent. Lors d'une partition, chaque individu qui n'est pas noyau est associé à la classe dont le noyau est le plus proche au sens de la moindre distance d. Lors d'une sélection, les nouveaux noyaux sont choisis parmi les membres des classes. Ceux-ci sont typiquement les individus « moyens » des classes. Par exemple, en cas de caractéristiques exclusivement numériques, on prend l'individu le plus proche du centre de gravité de la classe. Le critère d'arrêt teste une stabilité dans la partition en évaluant les changements entre deux itérations.

REGROUPEMENT-ASC(O,k)
%entrées : un ensemble de n individus, matrice de dissimilarité d
%sorties : une partition en k classes
$i := 0$
$\mathcal{N}_i :=$ CHOIX-NOYAUX-INITIAUX(O)
repeter
    $i := i + 1$
    pour tout $n_j \in \mathcal{N}$ **faire**
        $c_j := \{o \mid j = \text{argmin}(d(n_l, o))\}$
    $\mathcal{N}_i :=$ CHOIX-NOYAUX($\{c_1, ..., c_k\}$)
jusqu'a $\mathcal{N}_i = \mathcal{N}_{i-1}$
retourner $\{c_1, ..., c_k\}$

Algorithme 1: Regroupement par partition

La méthode réalise une stratégie d'*optimisation itérative*. Sa convergence dépend des paramètres comme la mesure de distance d et la sélection des noyaux de départ. Afin d'échapper aux problèmes de minima locaux, il est recommandé d'effectuer plusieurs exécutions, puis de rechercher des « formes fortes », des ensembles d'individus souvent regroupés.

La méthode des «nuées dynamiques»[Diday *et al.*, 1982; Celeux *et al.*, 1989] est une généralisation de la précédente. Elle se base sur deux fonctions f et g dont la première permet d'affecter une classe à un individu et la seconde choisit un représentant à un partir d'une classe. Le représentant n'est plus un seul individu, mais une «nuée centrale», un ensemble d'individus.

Un des handicaps majeurs de l'approche par partition est le besoin de fixer le nombre de classes *a priori*. La solution proposée est de faire varier k pour retrouver la valeur «convenable».

2.3.3 Cluster

Les membres de la famille Cluster sont parmi les premiers systèmes de regroupement conceptuel.

Les données du système Cluster/2 sont décrites à l'aide d'un formalisme logique, chaque individu est représenté par une conjonction de *sélecteurs*, termes prédicatifs associant une valeur à un attribut. L'algorithme construit un nombre fixe de classes également décrites par des conjonctions de sélecteurs et disjointes de par leurs descriptions.

Les processus de regroupement et de caractérisation sont indissociables : le premier est guidé par un critère qui mesure la qualité des descriptions des classes, la «fonction d'évaluation lexicographique» (LEF). LEF est un critère complexe, composé par une liste de critères élémentaires, chacun muni d'un seuil. Lors d'une évaluation, les critères sont appliqués à un ensemble de descriptions candidates dans l'ordre. Pour chaque critère on ne retient que les descriptions dont l'évaluation dépasse le seuil. Parmi ces critères, citons l'ajustement entre les descriptions et les données, la disjonction entre classes, la simplicité de leurs descriptions, etc.

Cluster/2, emprunte sa structure de contrôle à l'algorithme des «nuées dynamiques» décrit dans la section 2.3.2. On retrouve donc les notions de noyaux et de construction des classes en plusieurs itérations. On peut également identifier, d'une manière approximative, les deux fonctions préalablement citées. Ainsi, la fonction f est basée sur l'évaluation des descriptions des classes, alors que la fonction g produit une description de classe à partir d'un ensemble d'individus membres et un ensemble de noyaux des classes.

L'algorithme commence par choisir les k noyaux initiaux. Ensuite, un certain nombre d'itérations est effectué. En début de chaque itération, l'algorithme construit un ensemble de classes à descriptions disjointes. L'étape suivante est le choix des nouveaux noyaux des classes. L'algorithme s'arrête dès la satisfaction d'un critère complexe basé, entre autre, sur le critère de qualité LEF.

Cluster(O,k)
%entrées : un ensemble de n individus, nombre de classes k
%sorties : ensemble de k descriptions de classes
$\mathcal{N} := $ Choix-Noyaux-Initiaux(O)
repeter
    pour tout $n_j \in \mathcal{N}$ **faire**
        $c_j := Star(n_i|\mathcal{N} - \{n_i\})$
    $\mathcal{N} := $ Choix-Noyaux($\{c_1, ..., c_k\}$)
jusqu'a ARRET($\mathcal{N}, \{c_1, ..., c_k\}$)
retourner $\{c_1, ..., c_k\}$

Algorithme 2: Algorithme dans Cluster

Le choix des noyaux est aléatoire au départ. Ensuite, ceux-ci sont choisis en fonction du résultat du LEF par rapport au résultat de l'itération précédente : individus centraux si amélioration, individus marginaux si détérioration.

Les descriptions pour les k noyaux sont construites en deux étapes. Dans la première, pour chaque noyau n_i, son *étoile* $G(n_i|\mathcal{N} - \{n_i\})$ est construite à l'aide de l'algorithme Star qui est la description la plus générale qui couvre le noyau n_i et qui ne couvre aucun des autres noyaux. Les individus sont classés selon leur appartenance aux étoiles des noyaux. La seconde étape consiste en la construction de descriptions disjointes. Chaque étoile est réduite à la description des individus qui n'appartiennent qu'à la classe correspondante. Le reste des individus est distribué sur les classes en faisant attention à la disjonction de leurs descriptions. À chaque étape, la décision de l'affectation dépend du critère LEF évalué sur les descriptions produites.

2.3.4 Classification hiérarchique ascendante

Les méthodes ascendantes travaillent sur une matrice de dissimilarité. Elles construisent une classification binaire par des agrégations successives à partir de l'ensemble des classes-singletons d'objets dans O. À chaque pas, les deux classes de la partition courante les plus proches selon un critère D dit *indice d'agglomération* sont agrégées en une nouvelle classe. Celle-ci devient leur super-classe dans la classification partiellement construite. La construction s'arrête lorsque la classe unique englobant tous les objets est atteinte.

L'indice d'agglomération est une fonction basée sur la distance entre individus membres de chaque classe. Parmi les indices le plus souvent utilisés, citons :

l'indice du lien *minimal* : le minimum des distances entre deux membres
l'indice du lien *maximal* : le maximum des distances entre deux membres
l'indice du lien *moyen* : la moyenne des distances entre deux membres

```
Regroupement-Asc(O)
%entrées : un ensemble de n individus, matrice de dissimilarité d
%sorties : une classification
H : = {{o_1}, ...{o_n}}
tant que |H| > 1 faire
    soit h_1, h_2 dans H tel que D(h_1, h_2) soit minimal
    H : = H ∪ {h_1 ∪ h_2} − {h_1, h_2}
retourner H
```

Algorithme 3: Regroupement ascendant

L'algorithme générique de la méthode commence par la partition triviale construite par les singletons. A chaque pas, les deux ensembles les plus proches selon l'indice d'agrégation sont réunis en un seul. Les valeurs de la dissimilarité dans la matrice sont mises à jour à partir des anciennes valeurs. L'algorithme continue les fusions jusqu'à ce qu'il ne reste plus qu'un seul élément dans $\mathcal{H}$. De cette manière, il construit progressivement tout un ensemble de partitions intermédiaires qui constituent une hiérarchie binaire.

La formule de Lance-Williams [Lance et Williams, 1967b] permet d'exprimer de manière homogène et quasiment linéaire le calcul de l'ensemble des liens les plus couramment utilisés. Ainsi, pour tout élément h de la partition courante, la distance entre celui-ci et le nouvel ensemble, $D(h, \{h_1 \cup h_2\})$, est calculé comme suit.

$$D(h, \{h_1 \cup h_2\}) = \alpha_1 D(h, h_1) + \alpha_2 D(h, h_2) + \beta D(h_1, h_2) + \gamma |D(h, h_1) - D(h, h_2)|$$

Les divers liens sont obtenus en faisant varier les paramètres $\alpha_1, \alpha_2, \beta$ et γ. Par exemple, avec $\alpha_1 = \alpha_2 = 0,5$, $\beta = 0$ et $\gamma = -0,5$ on obtient le lien minimum.

Le résultat du travail d'un algorithme ascendant est couramment visualisé sur un diagramme. Pour chacune des classes, sa «hauteur» est proportionnelle à la valeur du lien lors de sa constitution. Le résultat est une *hiérarchie indicée*.

Le résultat de l'application d'un algorithme de construction hiérarchique de type *lien simple* sur les valeurs de la dissimilarité de la table 2.2 est dessiné sur la figure 2.1.

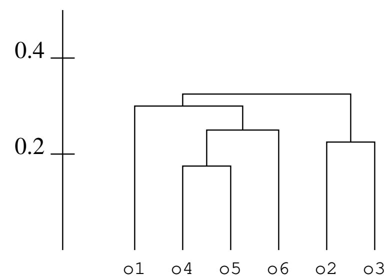


FIG. 2.1: *Une hiérarchie binaire indexée obtenue par classification ascendante avec le lien simple.*

2.3.5 KBG

KBG [Bisson, 1993] est un représentant des systèmes de regroupement conceptuel qui emploient des techniques basées sur la similarité entre individus. Le système présente de nombreuses fonctionnalités originales, nous n'allons nous intéresser ici qu'à la mesure de similarité utilisée par le système. D'une part, la mesure est définie sur des descriptions en LPO et permet donc de comparer des objets complexes. D'autre part, elle est utilisée non seulement lors du regroupement, mais aussi lors du processus de caractérisation. La manière de prendre en compte les relations inter-individus dans KBG nous servira pour établir notre propre approche vers le regroupement d'objets complexes.

Les données du système sont des descriptions d'entités complexes ou «exemples», chacune faisant apparaître un ensemble d'individus. Les propriétés des individus et leurs relations sont exprimées uniformément par des termes d'un langage de prédicats. La description d'un exemple est une conjonction de termes prédicatifs dont les arguments sont des symboles ou des valeurs. Les symboles, notés ici X_i, représentent des individus, appelés «entités». La sémantique des symboles est complètement définie par l'ensemble des termes dans lesquels ils apparaissent. Les valeurs, notées v_j, sont typées. Ainsi, un terme est l'instanciation d'un prédicat p :

$$p(X_1, \ldots, X_n, v_1, \ldots, v_m)$$

Tous les prédicats ont un type qui impose les types des arguments-valeurs : à la même place ne peuvent apparaître que des variables ou constantes de même type.

Le but de l'étape de regroupement conceptuel dans KBG est la construction d'une taxonomie sur l'ensemble des exemples. Les classes sont décrites dans le même langage conjonctif que les exemples, à cette différence près que les symboles sont remplacés par des variables et les valeurs par des ensembles de valeurs.

Le regroupement est guidé par la mesure de similarité entre exemples qui repose sur une mesure entre entités. Les entités sont comparées par rapport à leurs propriétés et par rapport aux relations qui les lient aux autres entités des exemples respectifs. Signalons que lors de la comparaison de deux entités, leurs relations communes sont exprimées par tous les termes où celles-ci apparaissent à la même position.

On parlera d'occurence d'une entité X et de l'ensemble des occurrences, $OCC(X)$. Pour un couple X, Y, $OCC_{com}(X, Y)$ est l'ensemble des termes à occurrences communes de X et Y, c'est-à-dire les termes où les deux entités apparaissent à la même position :

$$t_1 = p(X_1, \dots, X, \dots, X_n, v_1, \dots, v_m)$$
$$t_2 = p(Y_1, \dots, Y, \dots, Y_n, w_1, \dots, w_m),$$

Pour tous les couples (X_i, Y_i) on dira désormais qu'ils sont *connectés* par la relation p. Le principe fondateur de la similarité dans KBG est l'influence mutuelle entre les couples connectés. Ainsi, lors de la comparaison d'un couple X, Y, sa similarité dépend de la similarité de tous les couples connectés par une relation. Ce principe s'étend transitivement sur toute la structure relationnelle des exemples comparés, de sorte que les similarités de tous les couples connectés par des séquences de relations contribuent à la similarité de X, Y. L'influence de chaque couple est inversement proportionnelle à la longueur de la séquence de relations.

D'une manière plus formelle, la similarité entre exemples sim_E s'appuie sur la similarité entre termes sim_T qui, à son tour, dépend de la similarité entre entités sim_O et entre valeurs sim_V. La dernière fonction est une fonction générique, instanciable selon le type des valeurs. De plus, à chaque prédicat p, un poids $\omega(p)$ exprimant son importance dans le calcul de la similarité est associé.

La similarité de deux termes $t_1 = p(X_1, \dots, X_n, v_1, \dots, v_m)$ et $t_2 = p(Y_1, \dots, Y_n, w_1, \dots, w_m)$, relativement à un couple d'entité apparaissant au rang r, X_r et Y_r, est la somme des similarités des couples d'arguments respectifs, valeurs ou individus :

$$sim_T(t_1, t_2) = \omega(p) * (n^{-1} * \sum_{i=1}^{n} sim_O(X_i, Y_i)) * (m^{-1} * \sum_{j=1}^{m} sim_V(v_j, w_j))$$

Dans la formule ci-dessus, la similarité entre les entités initiales $sim_O(X_r, Y_r)$ est fixée à un. La similarité entre deux entités X et Y est calculée comme une combinaison linéaire entre les similarités des termes dans $OCC_{com}(X, Y)$. Celle-ci est normalisée par le maximum des deux sommes des poids des prédicats, l'une sur $OCC(X)$ et l'autre sur $OCC(Y)$.

$$sim_O(X, Y) = \frac{\sum_{t_1, t_2 \in OCC_{com}(X,Y)} sim_T(t_1, t_2)}{max(\sum_{p(..,X,..)} \omega(p), \sum_{p(..,Y,..)} \omega(p))}$$

Cette manière de définir les similarités entre entités et entre termes introduit des dépendances circulaires dans le calcul de la similarité : le calcul de $sim_O(X, Y)$ nécessite la similarité des couples connectés qui, à leur tour, dépendent de la valeur de $sim_O(X, Y)$. Cette auto-dépendance est résolue à l'aide d'une transformation vers un système d'équations. Les variables du système sont les similarités des couples d'entités. Pour chaque variable, une équation est obtenue à partir de l'expression de la similarité sous-jacente. Le système obtenu a un nombre égal de variables et d'équations. Il est linéaire sous certaines conditions et la forme spécifique des fonctions sim_O et sim_T assure l'existence d'une solution unique. Le système peut être résolu directement ou de manière itérative. En cas de multiples occurrences d'une entité dans l'exemple, avec même prédicat et à la même position, le système n'est plus linéaire. Une solution peut tout de même être calculée par une méthode itérative comme celle de Jaccobi.

Le calcul des similarités entre entités est suivi par une étape d'appariement. Le processus est guidé par les similarités, l'appariement établi cherche à maximiser la similarité totale. Le problème est comparable à la recherche d'un appariement de poids maximal dans un graphe bi-parti. Il est résolu de manière heuristique, en prenant les n couples d'entités exclusifs de similarité maximale. La similarité entre les exemples est la somme des similarités sur les couples de l'appariement, normalisée par le nombre d'entités dans un exemple.

Le fonctionnement de la mesure dans KBG, est illustré par la comparaison de deux exemples (selon [Bisson, 1992]) donnés ci-dessous (les noms des entités commencent par des majuscules).

```
E1 : père (Paul, Yves) sexe(Yves, male) âge(Yves,22) français(paul)
E2 : père (John, Ann) sexe(Ann, femelle) âge(Ann,12)
```

Pour les calculs, la similarité de valeurs sur `âge` est la valeur absolue de la différence, normalisée par l'étendue du type, 100.

$$sim_V(v, v') = (100 - |v - v'|)/100$$

Sur `sexe`, la similarité est de 1 si les deux valeurs sont identiques, zéro sinon. Les expressions des similarités entre termes relatives aux divers couples d'entités sont données dans la table ci-dessous.

`père`	sim_T sur `(Paul, John)` est $0,5(1 + sim_O(Yves, Anne))$
	sim_T sur `(Yves,Anne)` est $0,5(1 + sim_O(Paul, John))$
`âge`	sim_T sur `(Yves,Anne)` est $sim_V(22, 12)$
`sexe`	sim_T sur `(Yves,Anne)` est $sim_V(male, femelle)$

À partir des expressions des similarités entre termes, les similarités entre entités sont composées. Les poids des prédicats sont supposés égaux. Notons par x_1 et x_2 la valeur de $sim_O(Paul, John)$ et de $sim_O(Yves, Anne)$ respectivement. Ainsi, on obtient le système d'équations suivant :

$$x_1 = (0,5 + x_2/2)/2 \tag{2.3}$$

$$x_2 = (x_1/2 + 0,5 + 0,9 + 0)/3 \tag{2.4}$$

Le système est résolu soit directement, soit par une méthode itérative. Les solutions dans ce cas sont $x_1 = 0,38$ et $x_2 = 0,53$.

La méthode de regroupement dans KBG est ascendante, à chaque étape, les exemples les plus proches sont regroupés. Les groupes formés sont généralisés et le résultat est un nouvel exemple. Cet exemple remplace les exemples généralisés. Le processus de généralisation repose sur un appariement structurel également guidé par les similarités entre entités et entre termes. Le but est de préserver le plus grand nombre de relations communes aux exemples et de manière la plus connectée possible.

En somme, dans la méthode KBG la comparaison de deux individus nécessite la contribution d'un grand nombre d'autres couples d'individus. Cela revient à une exploration exhaustive de la structure relationnelle des exemples. Un tel procédé est justifié par la recherche du meilleur appariement entre individus : l'influence mutuelle conduit à des pôles de couples de similarité élevée qui permettent de conserver des « systèmes de relations », c'est-à-dire des ensembles de relations connexes. En cela, l'approche est conforme aux travaux effectués en psychologie cognitive sur l'analogie [Gentner, 1988].

2.3.6 Conclusion

Le système Kbg repose sur une approche algorithmique simple, la classification ascendante hiérarchique. Toutefois, comme l'auteur le souligne, d'autres méthodes basées sur les similarités pourraient aussi bien s'appliquer. La mesure de similarité offre des possibilités de paramétrisation, notamment par les poids des prédicats. Nous insistons sur cette universalité de la notion de similarité qui permet un large choix de stratégies de construction de classifications. De plus, dans Kbg , le même mécanisme est utilisé pour guider les processus de regroupement et de caractérisation, ce qui devrait assurer l'adéquation entre ceux-ci.

La méthode que Kbg emploie pour le traitement des relations a l'avantage de la précision : les individus sont comparés en fonction de leur « contexte ». De plus, elle est la seule qui permette de traiter de manière directe les dépendances circulaires entre couples d'individus induites par les relations.

Cependant, il faut noter le coût algorithmique considérable dû au besoin systématique de résoudre un système d'équations. De plus, l'approche concentre tout son intérêt sur le traitement des exemples, en négligeant les individus qui y apparaissent. Seuls les exemples sont organisés en classes. Il est difficilement imaginable d'appliquer le même procédé à des exemples dont les entités sont elles-mêmes complexes, c'est-à-dire composées par d'autres entités.

Une des sources de complexité est l'interprétation bi-directionnelle des relations entre entités. Dans le système Ribl [Emde et Wettschereck, 1996], une variante non-récursive de la similarité entre entités a été intégrée. Celle-ci profite d'une interprétation à sens unique de toutes les relations qu'on donne à partir du couple d'individus à comparer. L'exploration de la structure relationnelle se fait donc dans un seul sens. De plus, un seuil définit la profondeur maximale d'exploration de la structure lors de la comparaison.

2.4 Regroupement par généralisation

Cette section regroupe des méthodes qui ont un trait en commun : afin de décider si deux ou plusieurs individus constituent un groupe intéressant, on se réfère à la description du groupe qui constitue une généralisation des descriptions des individus.

La forme exacte et le principe d'obtention de la généralisation diffèrent selon les méthodes. Il va de même pour le jugement de la qualité d'une ou un ensemble de généralisations.

Les méthodes citées ne sont pas les seules de produire une généralisation, mais à utiliser cette généralisation dans l'appréciation des groupes.

2.4.1 Regroupement en des treillis de Galois

L'utilisation des treillis de Galois en analyse de données a débuté avec les travaux de Barbut et Monjardet [Barbut et Monjardet, 1970] sur l'interprétation des questionnaires. L'approche a pris un nouvel essor au début des années 80 avec les travaux de Wille et ses collaborateurs sur l'*analyse des concepts formels*[4].

Description de la technique

L'analyse de concepts formels consiste en la construction d'une correspondance de Galois entre deux ensembles, A et O, identifiés respectivement comme les attributs et les objets, munis

4. Formal Concept Analysis en anglais

d'une relation d'incidence I. Les groupes retenus correspondent à des concepts formels, c'est-à-dire à des ensembles d'individus fermés pour la correspondance de Galois.

Un concept est défini par rapport à un contexte formel, le triplet (O, A, I). Il est constitué par un ensemble d'objets X et un ensemble d'attributs Y, appelés respectivement *extent/intent* du concept. L'idée intuitive est que les objets membres de X partagent tous les attributs dans Y. De plus, les deux ensembles sont maximaux, dans la mesure où aucun ajout ne peut être fait à l'un sans être obligé de réduire l'autre.

Pour une définition rigoureuse de la notion de concept formel, la définition de deux applications est nécessaire. L'application f fait correspondre à tout sous-ensemble X de O le sous-ensemble Y de A tel que tout les objets dans X ont tous les attributs dans Y et Y est maximal. L'application g a un effet dual sur A et O.

- $f : \mathcal{P}(\mathcal{O}) \rightarrow \mathcal{P}(\mathcal{A})$ avec $X \subseteq O, f(X) = \{a \in A | \forall o \in X, oIa\}$
- $g : \mathcal{P}(\mathcal{A}) \rightarrow \mathcal{P}(\mathcal{O})$ avec $Y \subseteq A, g(Y) = \{o \in O | \forall a \in Y, oIa\}$

Les deux applications f et g, prises avec l'ordre induit par l'inclusion constituent une *correspondance de Galois* [Guénoche, 1990] entre les ensembles des parties de A et O. Sur cette base, deux opérateurs de *fermeture* [5] sont définis :

- $h : \mathcal{P}(\mathcal{O}) \rightarrow \mathcal{P}(\mathcal{O})$ avec $h = g \circ f$
- $h' : \mathcal{P}(\mathcal{A}) \rightarrow \mathcal{P}(\mathcal{A})$ avec $h' = f \circ g$

Un concept C est donc un couple (X, Y) où X est un ensemble d'objets et Y d'attributs pour lequel $X = g(Y)$ et $Y = f(X)$. Il est à noter que les deux ensembles sont *fermés*, c'est-à-dire $X = h(X)$ et $Y = h'(Y)$.

L'ensemble $\mathcal{C}(\mathcal{O}, \mathcal{A}, \mathcal{I})$ de tous les concepts sur un contexte (O, A, I) est muni d'un ordre partiel $\leq$ qui suit l'inclusion entre extents :

$$(X_1, Y_1) \leq (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2$$

De même, l'ordre entre concepts correspond à l'inverse de l'ordre entre intents :

$$(X_1, Y_1) \leq (X_2, Y_2) \Leftrightarrow Y_2 \subseteq Y_1$$

L'ensemble des concepts $\mathcal{C}(\mathcal{O}, \mathcal{A}, \mathcal{I})$ et l'ordre $\leq$ définissent une structure de treillis dit de Galois. Dans le treillis, les opérations de borne supérieure et borne inférieure sont définies comme suit : $\forall (X_1, Y_1), (X_2, Y_2) \in \mathcal{C}(O, A, I)$

- $(X_1, Y_1) \wedge (X_2, Y_2) = (X_1 \cap X_2, h'(Y_1 \cup Y_2))$
- $(X_1, Y_1) \vee (X_2, Y_2) = (h(X_1 \cup X_2), Y_1 \cap Y_2)$

Nous illustrons la construction de treillis sur l'exemple ci-dessous. La relation I est donnée par la table 2.3 où les lignes sont des objets ($O = \{a, b, c, d, e\}$) et les colonnes des attributs ($A = \{1, 2, 3, 4\}$). La présence d'un attribut pour un objet est marquée par un X dans la case respective. Par exemple, l'individu b possède tous les attributs alors que d ne possède que les attributs 1,3,4.

5. applications monotones croissantes, idempotentes et extensives

TAB. 2.3: *Une relation binaire entre deux ensembles*

	1	2	3	4
a		X	X	
b	X	X	X	X
c		X		
d	X		X	X
e	X	X		X

L'analyse en concepts formels consiste à rechercher les ensembles fermés de la correspondance de Galois induite par I. Ceci revient à rechercher tous les rectangles maximaux de la table composés de cases 'pleines' (notées par X). Par exemple, le rectangle $(\{a, b\}, \{2, 3\})$ est maximal, alors que $(\{b, d\}, \{3, 4\})$ ne l'est pas, mais $(\{b, d\}, \{1, 3, 4\})$ l'est. Les rectangles maximaux s'ordonnent naturellement par l'inclusion ensembliste entre ensembles d'individus. Ainsi, $(\{b, d\}, \{3, 4\})$ est dit plus spécifique que $(\{b, d, e\}, \{1, 4\})$. Le treillis de Galois, induit par la relation I est dessiné sur la figure 2.2.

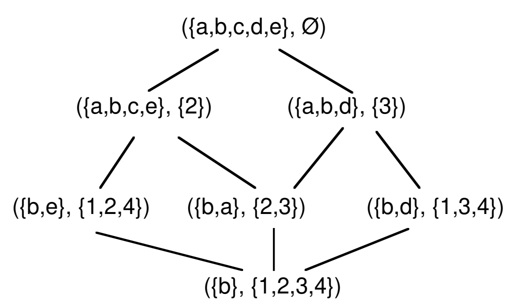


FIG. 2.2: *Le treillis de Galois obtenu à partir des données de la table 2.3*

Aspects algorithmiques de l'approche

Les concepts d'un contexte (O, A, I) sont représentés, sur la table de la relation I par des rectangles maximaux (de marques «présent», soit X ou 1). La recherche de tous les rectangles maximaux d'une relation binaire est un problème pour lequel plusieurs solutions algorithmiques ont été proposées. On dénombre au moins quatre algorithmes existants qui trouvent l'ensemble $\mathcal{C}(O, A, I)$ à partir de la table binaire : les algorithmes de Chein [Chein, 1969], de Ganter [Ganter, 1984], de Bordat [Bordat, 1986] et de Noris [Norris, 1978]. Parmi ces algorithmes, seul celui de Bordat construit explicitement le treillis de Galois sous forme de diagramme de Hasse. Une étude comparative des quatre algorithmes peut être trouvée dans [Guénoche, 1990].

Une autre direction dans les travaux sur le sujet est la conception d'algorithmes incrémentaux. L'incrémentalité, dans le contexte de construction de structures hiérarchiques est la capacité d'acquérir les données à la suite, une après l'autre, et d'adapter la structure existante plutôt que de recommencer sa construction de zéro. En général, les algorithmes incrémentaux, opposés aux algorithmes «par lots» [6], se comportent mieux sur les grands ensembles de données car ils n'explorent qu'une sous-structure de la hiérarchie entière. Des stratégies de construction incrémentale

6. batch en anglais

ont été proposées dans [Godin *et al.*, 1995b] et dans [Carpinetto et Romano, 1996].

Aspects description des données

Dans leur forme initiale, les algorithmes de construction de treillis s'appliquent exclusivement sur une table binaire. Toutefois, ce mode de description est de loin trop contraignant pour des applications réelles. Des efforts ont été fournis dans la direction de l'enrichissement de l'éventail des structures admissibles.

- Dans un premier temps, le **codage** : des descriptions plus complexes sont ramenées à des tables de booléens. C'est la méthode envisagée par les initiateurs de l'analyse de concepts formels [Ganter et Wille, 1989; Wille, 1992].

- Plus tard, l'application **directe** : le passage par des descriptions binaires n'est pas une obligation. En effet, la notion de correspondance de Galois n'implique pas nécessairement une telle nature des données. Il est envisageable de définir des fonctions f et g sur des structures plus riches. Une définition de la correspondance de Galois au sein des objets symboliques en analyse de données a été présentée dans [Brito, 1991] et étendue dans [Polaillon, 1998]. La structure du treillis de Galois sur des données structurées en arborescence (termes structurés) a été étudiée dans [Daniel-Vatonne, 1993]. Finalement, [Carpinetto et Romano, 1996] décrit une manière simple de joindre l'approche incrémentale vers la construction du treillis avec le traitement direct des attributs hiérarchisés.

Simplification de la structure du treillis

Le treillis de Galois a l'avantage incontestable de la complétude : il inclut tous les groupes d'individus potentiellement intéressants. En même temps, la structure de treillis dépasse celle d'un arbre de classes en complexité. Elle peut atteindre, au pire cas, la taille du treillis des sous-ensembles de O. On pourra remarquer que les expériences rapportées dans [Carpinetto et Romano, 1996; Godin *et al.*, 1995b] ont montré une croissance quasi-linéaire avec le nombre des objets à traiter. Toutefois, ces expériences ont été effectuées sur des données non-structurées et essentiellement discrètes.

L'analyse de données plus riches nécessite des phases de post-traitement qui visent à extraire une sous-structure du treillis de Galois en réduisant l'ensemble des nœuds [Girard, 1997].

2.4.2 Traitement des relations

L'approche par treillis a été adaptée au traitement de descriptions relationnelles sous forme de graphes conceptuels [Mineau, 1990; Bournaud, 1996]. Le principe de la construction exhaustive a été préservé : les systèmes aboutissent à une structure englobant la totalité des classes du langage de description fixé que nous allons détailler plus bas. La structure globale, appelée *Espace de connaissances* dans [Mineau, 1990] et *Espace des Généralisations* dans [Bournaud, 1996] contient l'ensemble des taxonomies possibles sur les objets considérés.

Le processus de construction des classes repose sur la généralisation : une classe est définie par sa description qui est la généralisation d'un ensemble d'autres descriptions, de classes et d'individus. La construction suit une direction ascendante, à partir des descriptions des individus jusqu'à la classe universelle.

Le problème de la généralisation dans les graphes conceptuels relève d'un isomorphisme de graphes et est donc un problème NP-difficile. Afin d'éviter une complexité déraisonnable, les méthodes en question ont recours à une restriction dans le langage de descriptions des classes qui assure une complexité polynomiale de la généralisation. La technique repose sur la notion d'arc conceptuel qui est composé d'un nœud relationnel et ses deux nœuds conceptuels adjacents[7]. La généralisation d'un arc est faite en par l»'augmentation» d'un ou plusieurs types dans ses nœuds. L'augmentation peut directement substituer à chaque type un type universel et générique noté « ?» dans [Mineau, 1990].

Une généralisation plus fine qui exploite l'ordre dans le treillis des types est proposée dans [Bournaud, 1996]. Ainsi, les graphes décrivant les individus sont décomposés en un ensemble d'arcs dont une partie appartiennent au graphe, alors que les autres sont des généralisations de ceux-ci. Les classes sont décrites par un ensemble d'arcs généralisés. L'appariement entre deux graphes devient un appariement entre les ensembles d'arcs étendu par un test de spécialisation entre arcs.

L'ensemble des classes construites est doublement contraint : d'une part, par le langage associé au *support* (voir section 1.2.2 du chapitre 1) et d'autre part, par l'exigence de non-redondance (deux classes ne peuvent avoir la même extension). Signalons au passage, que cette transformation vers un ensemble d'arcs, bien qu'elle constitue un gain important en efficacité, n'admet pas toujours une transformation inverse, c'est-à-dire qu'il n'est pas possible de restituer la structure du graphe d'origine dans tous les cas.

D'autres techniques permettant un traitement direct des relations au sein des données sont en cours de développement : la construction de treillis sur des descriptions en sous forme de graphes est esquissée dans [Liquiere et Sallantin, 1998], alors que [Simon, 1999] et [Simon et Napoli, 1999] présentent des adaptations des treillis de Galois aux objets complexes.

2.4.3 Conclusion

La construction de treillis de Galois est une technique de regroupement :

- **objective** : elle comporte *tous les groupes* d'individus qui satisfont une condition, en l'occurrence celle de la fermeture. En comparaison, les critères basés sur les ressemblances ou sur le pouvoir prédictif des groupes offrent des moyens de comparaison (en termes numériques) entre différentes alternatives, mais ils ne retiennent qu'une seule taxonomie [Mineau, 1990].
- **exacte** : tous les algorithmes arrivent à la même structure, elle ne dépend pas des choix de l'utilisateur et des biais des algorithmes.

Cependant, la complexité structurelle du treillis peut rendre difficile son interprétation : les classes intéressantes sont là, mais elles sont enterrées dans une multitude de nœuds et liens. Les méthodes de simplification n'apportent qu'une réponse partielle à ce problème : toute classe a une chance d'être retenue, encore faut-il pouvoir formuler un critère qui la sélectionne.

En ce qui concerne les descriptions relationnelles, les techniques employées pour restreindre la complexité de l'appariement sont propres au formalisme des graphes conceptuels. Elles sont comparables, d'une manière très approximative, à la réduction de la structure des graphes en un ensemble de propriétés booléennes. Comme dans le cas de KBG , les méthodes construisent une seule hiérarchie sur les graphes. Les individus apparaissant dans les graphes ne sont pas comparés explicitement. Cependant, dans [Bournaud, 1996] le treillis des types est utilisé comme une taxonomie de classes sur ces individus.

7. Les seules relations considérées sont binaires.

2.5 Formation de concepts

Les méthodes probabilistes se distinguent par le langage de descriptions des classes. Les classes ne sont pas de listes de couples attribut-valeur, mais de triplets attribut-valeur-probabilité qui définissent, pour chaque attribut une distribution de probabilités sur ses valeurs. Nous présentons dans la suite COBWEB l'algorithme référence de *formation de concepts*.

2.5.1 COBWEB

Dans la mouvance de formation de concepts, les méthodes manipulent des individus décrits par des couples attribut-valeur et des classes dont les descriptions sont des distributions de valeurs d'attributs. Une telle description ne permet d'établir l'appartenance d'un (nouvel) individu à une classe que de manière probabiliste. En revanche, elle offre une précision sur la répartition des valeurs au sein de la classe et permet de prédire, sous certaines hypothèses, les valeurs des membres. La prédictivité des classes est le facteur principal qui guide leur construction. La stratégie générale de construction est une incorporation progressive et descendante d'un objet dans l'arbre des *concepts probabilistes*. À chaque niveau, un choix parmi un ensemble d'opérateurs d'incorporation locale est effectué.

Dans COBWEB, le critère permettant de trancher est un critère probabiliste basé sur un compromis entre la similarité intra-classe et les dissimilarités inter-classe. Les deux sont à exprimer, d'après [Gluck et Corter, 1985], en termes de probabilités comme suit :

- la probabilité $P(c|o.a = v)$ qu'un objet o appartienne à la classe c en sachant que sa valeur pour a est v exprime la *prédictivité* de v pour la classe c.

- la probabilité $P(o.a = v|c)$ qu'un objet o ait la valeur v pour a en sachant qu'il est dans la classe c exprime la *typicité* de v pour la classe c.

Les classes homogènes ont une *typicité* forte des valeurs rencontrées, alors que des *prédictivités* fortes des valeurs parlent en faveur de classes contrastées. En poursuivant ce raisonnement jusqu'au bout, Fisher formule un critère de qualité pour une partition de la classe c en k sous-classes $\{c_1, c_2, ..., c_k\}$, l'utilité de partition :

$$PU(c, \{c_1, c_2, ..., c_k\}) = \frac{1}{k}(\sum_k P(c_k) \sum_i \sum_j P(o.a_i = v_{ij}|c_k)^2 - \sum_i \sum_j P(o.a_i = v_{ij}|c)^2)$$

L'intégration d'un nouvel objet dans la hiérarchie constitue une recherche heuristique dans l'espace des solutions selon la méthode du gradient[8]. Les opérateurs possibles à chaque étape portent sur la partition de la classe courante qui accueille le nouvel objet. Quatre opérateurs sont donnés dans la version originale de COBWEB, mais d'autres peuvent être facilement imaginés :

Incorporation : simple mise-à-jour des distributions de la classe courante, sans restructuration,

Création : une nouvelle classe est construite, sous-classe de la classe courante avec pour unique membre le nouvel objet,

Fusion : création d'une nouvelle classe à partir des deux classes les plus « ressemblantes » au nouvel objet, celle-ci devient leur sur-classe directe,

Division : suppression de la classe la plus ressemblante à l'objet par ses sous-classes.

8. *hill climbing* en anglais

Le sens général est descendant, l'incorporation de l'objet se fait depuis la classe-racine et suit un chemin dans l'arbre qui est éventuellement modifié. À chaque pas, le critère PU aide à choisir le résultat d'un opérateur, une partition donc, et la classe par laquelle la descente continue. Le processus s'arrête lorsqu'une classe feuille est créée.

Extensions

L'algorithme COBWEB étant défini à l'origine pour le traitement des attributs nominaux dans les individus, son application à des problèmes réels a rapidement conduit à la définition de certaines extensions destinées à combler ce manque de puissance. Parmi l'ensemble large et toujours grandissant des descendants de COBWEB, nous avons choisi deux systèmes qui sont proches de nos préoccupations.

L'algorithme CLASSIT [Gennari *et al.*, 1989] est une extension de COBWEB capable de prendre en compte les attributs *continus*. Il utilise une hypothèse de normalité pour la distribution des valeurs de ces attributs à l'intérieur d'un concept probabiliste. Le système LABYRINTH [Thompson et Langley, 1991] est destiné à la formation de concepts structurés à partir de descriptions relationnelles. Nous reviendrons sur la manière de décrire et traiter les relations dans LABYRINTH. Le système ADECLU [Decastaecker, 1993] se distingue des autres cités ici par le critère qui guide les regroupements. Celui-ci, appelé *critère d'adéquation*, se base sur une mesure de similarité entre un individu et un concept. Il favorise le *contraste* entre les concepts d'une partition dans l'arbre. Un autre aspect original du système est la possibilité de déterminer la pertinence relative des attributs dans les descriptions de chaque concept. Le même principe d'adéquation entre l'individu à classer et un concept est utilisé par le système INC [Hadzikadic et Yun, 1989].

La méthode de LABYRINTH

Les données de la méthode sont des entités composites avec un nombre fixe de composants et des relations entre ceux-ci. Les composants sont cités au sein du composite sans être repérés par des attributs particuliers. Les relations entre composants font partie de la description du composite. Chaque entité, composant ou composite, peut posséder ses attributs propres. Les composants peuvent, à leur tour, avoir des composants et ainsi de suite. L'ensemble des composants d'un individu composite est organisé en un arbre appelé *partonomie*. La relation de composition est homogène, dans la mesure où tous les composants au sein du même composite ont le même rôle et sont donc interchangeables. On distingue des niveaux dans la partonomie en fonction de la profondeur de chaque élément. Les composants d'un même niveau sont de nature identique et différente des autres niveaux.

Les éléments de chaque niveau sont organisés en une hiérarchie à part. Les classes, *concepts probabilistes*, de chaque hiérarchie sont décrites par des attributs simples ainsi que par deux types d'attributs liés aux composants. Le premier type d'attribut désigne les composants directs, ceux du niveau immédiatement au-dessous de la partonomie. Leur nombre est égal au nombre des composants et l'association d'un composant à un attribut est faite par un appariement. La valeur d'un tel attribut est un ensemble de classes sur les composants. Ces classes font partie de la hiérarchie du niveau inférieur. La probabilité associée à chaque classe de composants au sein de l'attribut fait référence à sa prédictivité au sein de sa propre hiérarchie. Le second type d'attribut décrit les relations entre composants. Ils sont de nature binaire et font référence à une relation et deux attributs. La probabilité associée indique la fréquence des cas où la relation particulière existe entre les composants associés aux attributs.

De cette manière, la structure compositionelle des individus résulte en une description des classes en termes d'autres classes, ce qui n'est pas sans rappeler la structure d'une base de connaissances à objets.

Le but de la méthode est donc de construire de manière incrémentale un ensemble de taxonomies, une par niveau de la partonomie. L'algorithme de regroupement englobant procède par intégration d'un individu complexe à l'ensemble des taxonomies. Les partonomies sont parcourues en sens ascendant : à chaque niveau, un algorithme de type COBWEB permet d'intégrer les individus dans leur hiérarchie tout en prenant en compte le résultat de l'intégration au niveau inférieur.

La structure relationnelle des individus manipulés intervient à plusieurs endroits de l'algorithme de base, COBWEB. Nous avons déjà évoqué la *description* particulière des classes composites. Lors de l'*intégration* d'un nouvel individu à une telle classe, les composants de l'individu sont déjà intégrés dans leur hiérarchie et le but est de les apparier aux attributs de la classe. Le critère guidant la recherche de l'appariement est l'augmentation de la prédictivité totale de l'ensemble des attributs actualisés. Rappelons qu'un tel attribut a pour valeur un ensemble de classes. Il est actualisé avec un nouveau composant en prenant la généralisation de sa valeur actuelle et du nouvel individu. Cette généralisation est un ensemble de classes qui «couvre» les anciennes classes et l'individu dans la taxonomie. La sélection de cet ensemble et la recherche d'un appariement sont deux procédures à forte complexité car un grand nombre de combinaisons sont examinées.

L'approche de LABYRINTH vers le traitement des données relationnelles possède des aspects intéressants pour notre propre problématique. D'une part, la méthode organise les individus de type différent dans des taxonomies séparées. Les classes sur un type d'individus font référence aux classes sur les type des composants. D'autre part, lors du regroupement, qui est une phase d'intégration d'un individu à une classe, les composants de l'individu participent non pas avec leurs caractéristiques comme dans KBG, mais avec leur classe, et donc leur position, dans la taxonomie sous-jacente. De cette manière les connaissances sur la structure conceptuelle de l'ensemble des composants sont utilisées dans le regroupement des composites.

Cependant, l'approche de LABYRINTH n'offre pas toute la généralité du traitement des relations entre individus. Si la composition est le pivot de l'algorithme de regroupement global, le reste des relations n'a qu'un rôle secondaire. Ainsi, les relations entre composants ne sont pas prises en compte lors de leur propre regroupement. Une limitation supplémentaire est le nombre fixe des composants. Mais le problème majeur avec LABYRINTH est sans doute sa complexité accrue qui vient des recherches exhaustives d'appariement optimal entre composants et attributs.

La méthode originelle a été reprise dans [Ketterlin, 1995] et adaptée à d'autres types de descriptions. Ici les relations entre individus, composition et autres, sont représentées par un attribut relationnel. Les relations sont orientées et induisent une structure, appelée espace des abstractions, qui lie les différents types d'individus. L'espace se présente comme un DAG où les attributs sont des arcs et les types d'objets des nœuds. L'absence de circuit assure l'application de la stratégie de LABYRINTH, c'est-à-dire le parcours du DAG par niveau en intégrant les individus d'un niveau dans leurs hiérarchies avant de passer au niveau supérieur. La nouvelle méthode est de loin plus efficace que LABYRINTH même si son formalisme de description des données est moins expressif.

2.5.2 Autres méthodes probabilistes

La formation des concepts a inspiré des méthodes utilisant des représentations et critères de regroupement de nature probabiliste. Ainsi, l'algorithme WITT [Hanson et Bauer, 1989] utilise une stratégie ascendante de construction de classes dont l'indice d'agrégation est basé sur les corrélations entre attributs (nominaux). Globalement, le critère, appelé *indice de cohésion conceptuelle*, tente de maximiser la corrélation au sein des classes construites (*cohésion*) et de minimiser les corrélations inter-classe (*contraste*). L'algorithme AUTOCLASS [Cheeseman *et al.*, 1988] emploie une stratégie probabiliste Bayesienne pour découvrir la partition la plus *probable* d'un ensemble d'individus.

2.5.3 Conclusion

Les méthodes incrémentales représentent une approche efficace vers le regroupement conceptuel. Du coup elles sont bien adaptées au traitement de grands ensembles de données. Leur fonctionnement autonome fait d'elles un bon outil d'extraction de connaissances dans les bases de données.

En même temps, le biais vers la détection de classes maximalement prédictives ne facilite pas leur application dans un contexte de structuration d'un ensemble de données où la prédiction n'est pas le but. En effet, ces méthodes cherchent à fixer une seule taxonomie par un processus relativement fermé, ne permettant pas l'intervention d'un humain. De plus, la forme définitive d'une taxonomie dépend de l'ordre dans lequel les individus sont acquis par le système.

2.6 Synthèse

Nous avons présenté un survol du vaste domaine du regroupement, partagé essentiellement entre l'apprentissage automatique et l'analyse de données dont le but est de situer les paradigmes suscitant le plus grand nombre de travaux à l'heure actuel.

Rappelons les caractéristiques de la construction de taxonomies dans les objets que nous avons dégagées préalablement (voir 2.1.3). Celles-ci sont liées d'une part à la nature du processus, c'est-à-dire à son aspect *exploratoire*, *interactif* et *itératif*, et à son *intégration* au sein d'une base de connaissances. D'autres exigences proviennent du langage de description que les objets offrent : traitement des relations, des valeurs multiples, des ensembles de types extensibles etc. Ainsi, une méthode de regroupement conceptuel sur les objets doit être suffisamment *puissante* pour traiter la structure complexe des objets et cela de manière *directe* et en s'appuyant au maximum sur les connaissances existant dans une base. En même temps une *flexibilité* est exigée : la construction de diverses taxonomies sur le même ensemble d'objets doit être possible afin d'assurer une meilleure exploration de l'espace des taxonomies possibles. Cela veut dire, en clair, que les outils de construction à adopter sont paramétrisables, mais aussi qu'une interprétabilité des paramètres est exigée.

La formation de concepts semble le plus éloignée de nos préoccupations. Ceci peut sembler paradoxal, compte tenu des ressemblances qui existent entre les processus de classification, ou le raisonnement classificatoire [Napoli, 1992], dans les systèmes de RCO et le processus général de formation de concepts. Force est de reconnaître également que les seuls systèmes de regroupement conceptuel qui ont connu un certain succès sont de cette famille et cela grâce à leur efficacité et leur mode de fonctionnement autonome. Toutefois, leur atout majeur, à savoir le pouvoir prédictif des concepts probabilistes, peut devenir un handicap quand il ne s'agit pas de chercher à prédire, mais d'organiser les connaissances sur un domaine [McKusick et Langley, 1991; Fisher, 1996]. Un autre inconvénient lié à la nature incrémentale et autonome du processus de formation de concepts : les méthodes de la famille COBWEB construisent une *taxonomie unique* dont la forme exacte dépend de *l'ordre* de présentation des individus.

Cependant, nous allons retenir les principes employés par les méthodes de formation de concepts pour le regroupement d'individus complexes. D'une part, chaque type d'individu est organisé dans une hiérarchie propre. Les classes d'une hiérarchie sont décrites par des classes des hiérarchies sur les types liés par des relations. D'autre part, lors du regroupement sur un type d'individus, tous les individus qui lui sont liés par une relation sont pris en tant que membres de leur classes dans la hiérarchie sous-jacente. Ce principe exclut la circularité dans les descriptions des individus : chaque individu référencé par l'attribut d'un autre individu doit être incorporé dans sa hiérarchie avant que le second individu ne le soit.

Le regroupement par généralisation ne nécessite qu'une opération de synthèse de description de classes. Celle-ci peut être aisément définie à partir des deux relations qui sont disponibles dans la plupart des systèmes à objets, à savoir la spécialisation et le sous-typage. Cependant, le mode de regroupement par généralisation n'autorise pas les dépendances circulaires entre entités regroupées. En effet, comme dans le cas précédent, il exige que les classes des objets référencés soient connues au préalable.

D'autre part, la taille du treillis et sa complexité structurelle constituent un obstacle considérable lorsqu'il s'agit de laisser l'interprétation des résultats à un spécialiste humain. L'extraction d'arbres à partir d'une hiérarchie exhaustive est certes possible. Elle peut être effectuée à l'aide d'un critère paramétrable [Bournaud et Ganascia, 1995] et dans un mode interactif, ce qui répond effectivement à l'aspect exploratoire du processus dans les objets. Toutefois, la définition d'un tel critère n'est pas une tâche aisée et revient souvent à réintégrer des critères de regroupement déjà existants [Girard, 1997].

Quant à l'approche par ressemblances, elle donne plus de possibilité de contrôle sur le processus de regroupement : par les poids des diverses caractéristiques, par les mesures et les algorithmes choisis. De plus, une mesure peut être définie sur un grand nombre de types de données et structures d'individus. L'approche permet de combiner les propriétés et les relations lors de la comparaison des individus. C'est la seule technique effective qui traite les dépendances circulaires entre individus de manière directe. Toutefois, les avantages de la ressemblance sont contre-balancés par le besoin de connaître les valeurs numériques des poids des attributs.

En somme, aucune des approches ne convient parfaitement dans notre contexte. La similarité et la généralisation permettent de faire varier le critère de regroupement/extraction. La formation de concepts est apte à intégrer facilement les connaissances taxonomiques dans le regroupement. Les métriques s'adaptent facilement au traitement de divers types de données et de structures dans les objets.

2.7 Notre approche

Notre choix, pour des raisons de complétude s'est arrêté sur une approche par proximité avec une mesure à la KBG, mais mieux adaptée aux objets. En particulier, nous voulons appuyer, autant que possible, la mesure sur les taxonomies existantes plutôt que sur les valeurs des attributs ainsi obtenant un double avantage. D'une part, les taxonomies existantes se retrouvent naturellement incluses dans le processus de construction de nouvelles taxonomies, ainsi assurant une bonne réutilisation des connaissances sur le domaine disponible au moment du regroupement. D'autre part, cela permet d'éviter le calcul systématique d'un système d'équations ce qui représente un gain substantiel en efficacité.

Le choix d'une métrique comme celle de KBG et surtout sa variante «orientée» dans RIBL assure une flexibilité et surtout une complétude. Les alternatives qui consistent à construire les descriptions des classes en même temps que les ensembles de leur membres peuvent s'avérer

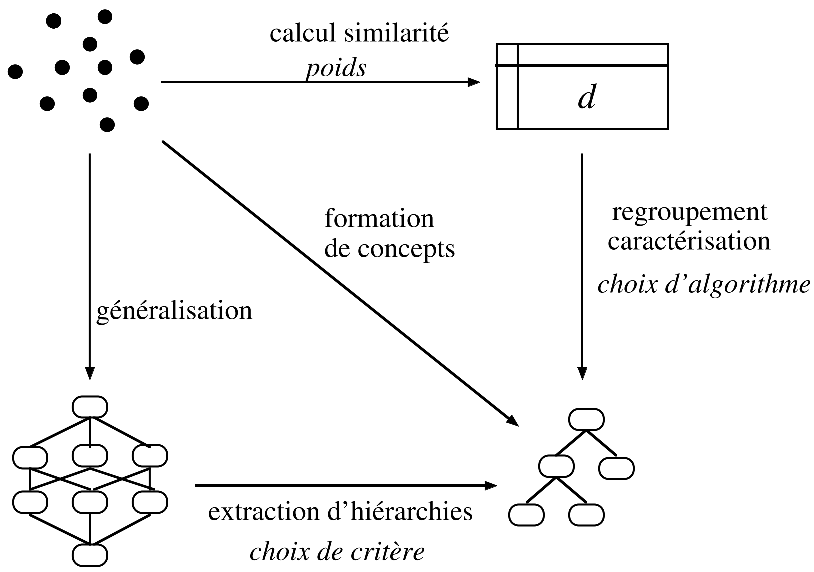


FIG. 2.3: *Représentation schématique des trois approches vers le regroupement conceptuel : le regroupement par généralisation, le regroupement par similarité et la formation de concepts. Les éléments nécessitant une intervention de la part de l'utilisateur sont donnés en italique.*

impossible dans deux cas : quand les co-domaines de certains attributs relationnels ne sont pas munis de taxonomies convenables [9] et quand les descriptions des objets exhibent des circuits.

Par ailleurs, l'utilisation systématique d'une métrique sur les attributs des objets obligerait à des explorations exhaustives, et donc coûteuses, dans les réseaux d'objets, les structures induites par les attributs relationnels. Il est de loin préférable de raccourcir les recherches effectuées pour une comparaison, d'autant plus que l'influence des objets liés diminue de manière quadratique avec leur éloignement. Une solution semble être de comparer les objets référencés non pas en tant que structures d'attributs, mais en tant que membres de leur classes au sein des taxonomies respectives. Ceci n'est pas toujours possible, la présence de circuits incluant des objets du même concept (comme celui induit par l'attribut `conjoint`, par exemple) est un des facteurs qui peuvent empêcher cette stratégie d'aboutir. En outre, rien ne garanti que les taxonomies existantes reflètent bien l'objectif du regroupement. Leur utilisation à des fins de comparaison ne peut donc être un principe.

Il s'en suit que la méthode doit être suffisamment flexible pour permettre d'inclure les connaissances taxonomiques quand celles-ci sont conformes au but de la comparaison et de les ignorer, en se rapportant aux descriptions propres des objets dans le cas inverse [10].

Une telle approche laisse une grande liberté quant à la configuration de l'outil de comparaison, la métrique. On pourrait imaginer que l'utilisateur ait à décider pour chaque concept lié au concept à regrouper si la ou les taxonomies sont convenables. En réalité, les décisions seront limitées, sauf exception, aux concepts directement liés, c'est-à-dire les co-domaines des attributs relationnels des objets à regrouper. En effet, la pertinence des classes construites de manière automatique sera jugée sur leurs descriptions qui, sur les attributs relationnels, seront produites

9. Ce danger est toujours présent avec LABYRINTH : rien n'assure que les hiérarchies qui évoluent en parallèle aillent vers le résultat désiré et les distorsions à un niveau affectent les niveaux supérieurs.

10. Évidemment, cette décision appartient à l'utilisateur ayant demandé le regroupement car lui seul détient le but.

à partir des classes sur les concepts co-domaines. Il y a donc une condition préalable au regroupement qui est la pertinence des classes accessibles sur les attributs relationnels sélectionné[11]. Dans la majorité des cas donc il y aura une adéquation entre les taxonomies sur les concepts référencés et les objectifs de la construction de classes sur les objets référant. Par conséquent, il sera utile d'appuyer le regroupement sur la structure de ces taxonomies qui serviront aussi pour la caractérisation des classes, assurant ainsi une meilleure cohésion entre les deux opérations, séparées dans notre approche.

Dans la suite de ce mémoire nous décrivons les outils nécessaires à la réalisation de cette approche. En premier, une mesure de dissimilarité est présentée qui compare les objets selon toutes leurs caractéristiques, propriétés et relations confondues. Ensuite, la manière d'étendre celle-ci sur des attributs multi-valués est précisée. Nous montrons également que la mesure peut être utilisée dans les cas de dépendances circulaires. Finalement, des opérations de synthèse de descriptions sont proposées.

11. Les seules exceptions sont les cas où un attribut est intéressant pour le regroupement, mais pas pour la caractérisation. À notre connaissance, il n'y a pas de cas exhibant cette particularité.

Deuxième partie

Construction automatique de classes à partir d'objets

3

Construction automatique de classes

Introduction

Dans le chapitre 2 nous avons vu que les méthodes de regroupement automatique permettent de construire des classes à partir d'un ensemble d'individus, avec une efficacité algorithmique qui dépend de la homogénéité dans les descriptions de ces individus. Les relations inter-individus ne sont que très rarement admises au sein des descriptions et, quand elles le sont, leur traitement est soumis à de multiples contraintes imposées par le souci de limiter la complexité algorithmique. Le chapitre 1 a dessiné un tout autre paysage : un large éventail de moyens d'expression des connaissances d'un domaine, avec un mode de représentation favorisant la distinction au sein de la base de connaissances de plusieurs familles d'individus. Sur le point du traitement de données très structurées il y a donc un retard des techniques d'analyse par rapport aux formats des données que l'on peut rencontrer en représentation de connaissances par objets (RCO).

Ce retard a été maintes fois constaté [Chouvet *et al.*, 1996; Kodratoff, 1997; Hand, 1997], cependant les travaux visant le rapprochement des deux champs sont encore rares [Brachman *et al.*, 1993; Kietz et Morik, 1994; Ketterlin, 1995; Han *et al.*, 1998]. Un problème reste en tout cas peu étudié, c'est l'impact de la forte *structuration* [12] dans les données induite par les relations inter-individus. La présence de ce genre de relations demande à ce que le processus de construction de classes soit repensé : le regroupement d'un ensemble d'objets n'est plus une tâche isolée, mais s'inscrit dans un processus global d'organisation de connaissances individuelles. De même son résultat, la taxonomie de classes, n'est plus une structure indépendante, mais doit s'intégrer au sein d'une organisation plus complexe liant des classes sur des familles d'objets différentes.

Dans ce chapitre, nous examinons de manière détaillée les exigences de l'environnement à objets envers l'application de méthodes d'analyse automatique. Nous mettons en évidence, en particulier, le besoin d'une discipline (appelée *stratégie*) pour la conduite successive de multiples tâches de regroupement au sein d'une base de connaissances dont le but principal est une utilisation optimale des connaissances disponibles sur le domaine. De même, nous motivons la définition d'une méthode de regroupement conceptuel capable de traiter un ensemble de familles d'individus à la fois tout en respectant les relations inter-individus. La réalisation de la méthode devient possible grâce à un ensemble d'outils concrets dont les caractéristiques répondent aux exigences de l'environnement à objets. Nous décrivons également un cadre unificateur pour les divers aspects de la description des objets qui permet, sous la forme d'un principe fondamental de comparaison, d'extraire puis de décrire les traits communs des objets. Ce cadre repose sur

12. Signalons qu'il ne s'agit pas ici du même type de structure que celle qui a motivé la création de systèmes comme KBG [Bisson, 1993] ou MSG [Mineau, 1990], une structure qui reste essentiellement intra-individu.

une double vision sur les objets : d'une part, comme des structures d'attributs et d'autre part, comme des membres des classes.

Au début du chapitre, un exemple illustre les diverses situations qui se présentent lors de la construction incrémentale d'une base de connaissances (section 3.1). Nous en tirons un ensemble d'exigences que notre approche vers la construction de taxonomies doit satisfaire. La section 3.2 présente brièvement chacun des aspects pertinents de cette approche avec les raisons qui ont motivé une définition explicite. Elle est suivie par la description détaillée de chaque élément constitutif de l'approche. Tout d'abord, nous présentons le principe fondamental de comparaison des objets, à savoir une décomposition récursive de tout traitement selon les niveaux d'imbrication dans la structure des descriptions des objets qui est, à son tour, appuyée sur une double vision sur les objets à comparer (section 3.3). Ensuite (section 3.4), nous décrivons une stratégie de construction successive de taxonomies basée sur le parcours du graphe composé par les familles d'objets et leurs relations. Son résultat étant le découpage de l'ensemble de toutes les familles en sous-ensembles qui sont indissociables pour le regroupement, nous suggérons une méthode de construction simultanée de taxonomies sur plusieurs familles (section 3.5). À la fin du chapitre, les outils nécessaires à la construction – une mesure de dissimilarité, des algorithmes de regroupement et des opérateurs de caractérisation – sont esquissés (section 3.6). Leur description détaillée est l'objet des chapitres suivants.

Dans la suite, afin de simplifier le discours, nous emploierons les notions de concept, classe, objet, composant, lien, type, instance, valeur telles qu'elles sont définies dans la section 1.4 du chapitre 1.

3.1 Exemple introductif

Nous allons considérer une base de connaissances modélisant un institut de recherche en informatique. La base traduit la structure de l'institut, elle contient des objets représentant des chercheurs, mais aussi des projets de recherche et des laboratoires. Dans la suite, nous présentons tout d'abord un état opérationnel de la base, pour revenir ensuite en arrière et retracer les étapes du processus de construction de la base. Cela nous permettra de fonder les exigences envers un outil d'aide à la construction auxquelles nous chercherons à répondre par la suite.

3.1.1 La base de connaissances

La base est composée de trois concepts, `laboratoire`, `projet` et `chercheur`. Chaque concept définit un ensemble d'attributs décrivant ses instances : par exemple, les laboratoires sont décrits pas leur `nom`, leur `adresse`, leur `directeur` et l'ensemble des `projets` qui en font partie. À leur tour, les projets sont caractérisés par leur `nom`, leur `chef` et l'ensemble des chercheurs `membres`. Finalement, les chercheurs ont un `nom`, un `grade`, un `nombre-de-publications`, un projet de rattachement (attribut `membre-de`) et un laboratoire où ils travaillent (attribut `affiliation`).

Un exemple de cinq objets des trois concepts de la base est montré sur la figure 3.1. Chacun des concepts est également représenté accompagné de la taxonomie de classes qui lui est associée. Sur le dessin, les objets sont liés à leurs classes de rattachement (lien `est-un`). Les relations entre concepts, classes et objets établies moyennant les attributs `projets` et `membres` sont indiquées. Ainsi l'objet représentant le laboratoire `Clonia` est lié par son attribut `projets` à deux objets, représentant ses projets, `Blast` et `Rotor`. Deux objets représentant des chercheurs de `Rotor`, `Paul` et `Louis` sont également présentés. Ils son référencés par l'attribut `membres` de l'objet `Rotor`.

À noter la présence des liens sur les trois niveaux de la structure de la base : les liens entre objets sont *définis* au sein des concepts sous-jacents et *contraints* dans les classes. De cette

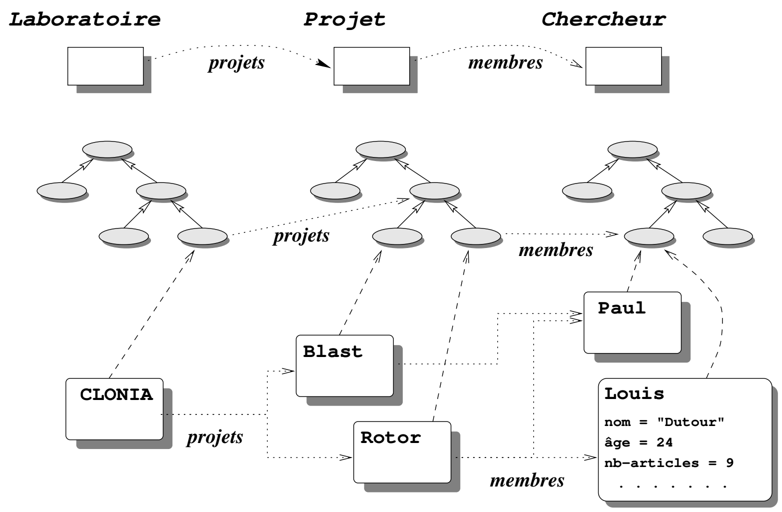


FIG. 3.1: *Exemple des liens entre unités de description dans une base de connaissances à objets : les concepts sont dessinés comme des rectangles, les objets comme des rectangles aux coins arrondis et les classes comme des ovales. Les objets sont liés par leurs attributs relationnels ; les concepts sont liés par les attributs de concept relationnels qui définissent les attributs des instances ; les classes sont liées par des attributs de classe qui contraignent les attributs des instances.*

manière, des liens entre les concepts et entre les classes sont établis. Par exemple, l'attribut `membres` de la classe `équipe-jeune` sur le concept `projet` pourra avoir comme valeur la classe `jeune-chercheur` sur le concept `chercheur`, indiquant ainsi que tous les membres d'un projet de la classe `équipe-jeune` doivent appartenir à la classe `jeune-chercheur`.

La construction de classes au sein d'une base à objets est essentiellement concernée par les liens inter-classes qui doivent être inférés automatiquement à partir des liens entre objets. Pour cela, elle peut déjà s'appuyer sur les liens entre concepts.

3.1.2 Vue dynamique sur la construction de la base

La figure 3.1 illustre une phase avancée de la construction de la base de connaissances. En effet, chacun des trois concepts possède une taxonomie de classes qui pourra être utilisée dans la consultation/complétion de la base. Toutefois, pour nous, cet état de la base est un but car nous nous intéressons au processus de complétion progressive de la base qui conduit à cet état.

En général, au début de sa construction, la base est peu structurée, c'est-à-dire qu'elle contient un nombre réduit de classes. Dans les cas qui nous intéressent, la base contient, en plus de ces classes, un bon nombre d'instances. Rappelons ici qu'en RCO, il est souvent possible de créer les objets en tant que membres d'une classe, puis de les rattacher à une autre classe, souvent une sous-classe de la première[13]. Notons au passage que dans certains formalismes dits à objets *semi-structurés* [Abiteboul, 1997; Buneman *et al.*, 1997], il est possible de commencer la constitution d'une base dans l'absence de toute classe.

Pour nos objectifs, les étapes de la complétion d'une base se résument en des ajouts de

13. Sur ce point, TROEPS enregistre une avancée car la création des objets se fait de manière complètement indépendante de leur appartenance à une classe particulière, elle fait partie des prérogatives du concept. Cela permet de déclarer les objets sans se préoccuper de leur classe exacte qui sera déterminée par la suite.

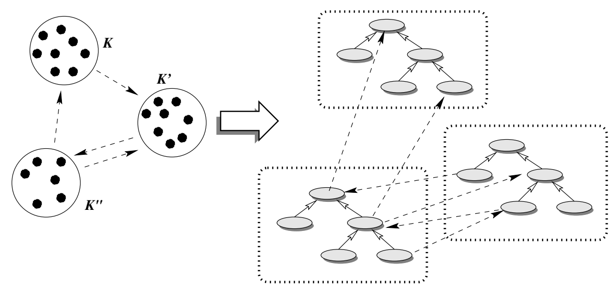


FIG. 3.2: *Construction de classes sur plusieurs concepts. L'ensemble des familles d'objets doit être organisé en classes. Les relations entre objets des diverses familles imposent des relations entre les classes à construire.*

nouvelles classes. Le processus global de sa construction peut donc être vu comme le passage progressif d'une base peu structurée à une base mieux structurée. Les étapes initiale et finale du processus sont visualisées sur la figure 3.3.

Afin de pouvoir livrer une aide effective lors du processus de complétion de la base, un outil de construction de classes doit pouvoir s'adapter à son degré de complétion. Lorsque la base est faiblement structurée, c'est-à-dire en présence de peu de classes, il est fort probable que plusieurs taxonomies doivent être construites. Dans ce mode de fonctionnement, l'outil devra pouvoir ne s'appuyer que sur les objets, c'est-à-dire sur les données disponibles. La tâche à accomplir est à rapprocher à la découverte des connaissances dans les bases de données.

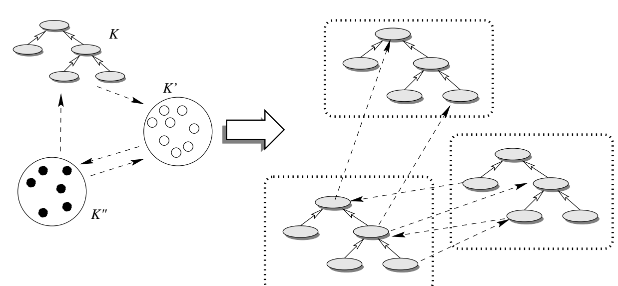


FIG. 3.3: *Définition de classes sur plusieurs concepts avec un certain nombre de taxonomies disponibles.*

En même temps, l'outil doit pouvoir s'adapter à des situations où des taxonomies existent déjà (voir figure 3.3). Dans ce mode de fonctionnement, plus proche de la *structuration de bases de connaissances* comme définie dans [Mineau, 1990], les taxonomies existantes sont des connaissances sur le domaine qu'il convient d'intégrer au processus de regroupement car elles peuvent mener à la constitution de classes pertinentes sur certaines familles d'objets. Par exemple, la classe des chercheurs appartenant à des équipes jeunes ne pourra très probablement pas être constituée avant que la classe `équipe-jeune` ne le soit.

3.1.3 Le problème

Afin de remplir notre objectif, à savoir l'aide à l'organisation des objets en classes au sein d'une base de connaissances, nous devons résoudre le problème suivant :

> *construire un ensemble de classes à partir d'objets sur plusieurs familles d'objets reliées par des attributs.*

Dans les deux premiers chapitres de ce mémoire nous avons défini un certain nombre d'exigences envers la méthode de construction de classes dans les objets. Celles-ci nous ont guidés dans le choix d'une approche basée sur les proximités entre objets. Dans la suite, nous rappelons ces exigences et les complétons par d'autres qui sont directement liées au contexte objets.

Intégration au sein de la base

Un ensemble d'objets à regrouper en classes constitue une partie de la base et peut entretenir des liens variés avec le reste de son contenu. Sur l'exemple ci-dessus, les chercheurs sont liés à leurs projets et *vice versa*. Pour cela, le processus doit être complètement **intégré** au sein de la base.

Forte structuration des connaissances manipulées

Ceci est de loin notre préoccupation principale. En effet, les objets constituent un formalisme très riche en moyens d'expression (types de données variés) et hautement structuré (collections, relations entre objets). Le processus de construction et plus précisément les mécanismes d'inspection et de comparaison d'objets doivent porter sur **l'intégralité des descriptions** des objets, indépendamment de leur degré de structuration.

Prépondérance des objectifs du concepteur

Il ne s'agit pas ici de *détecter* une taxonomie unique et prédéterminée, mais au contraire, d'*aider* le concepteur de la base de retrouver une **organisation de ses connaissances qui le satisfait**. Pour cela le processus de conception doit être conçu de telle manière que le concepteur puisse exprimer son «point de vue». Ainsi, dans le cas du concept chercheur, on doit pouvoir construire des classes qui prennent en compte leur affiliation, mais aussi des classes qui l'ignorent. Plus encore, la contribution des diverses caractéristiques doit être **paramétrable**.

Richesse de l'environnement

Qui plus est, les classes recherchées font référence à d'autres classes qui peuvent exister ou être recherchées à leur tour. Ce fait doit être pris en compte lors de la construction de classes : notre conviction est que les classes existantes expriment la vision de leur(s) concepteur(s) et qu'appuyer la construction d'autres classes sur celles-ci assure une meilleure adhérence au point de vue du concepteur de la base.

Évolution de la base

Le contenu de la base de connaissances reflète l'état de la formalisation des connaissances sur le domaine. Celui-ci s'enrichit progressivement et le processus doit pouvoir **s'adapter** à cette richesse variable dans la recherche de classes pertinentes.

3.2 Les constituants de l'approche

La prise en compte des exigences décrites dans la section précédente ne peut pas être réussie dans le cadre trop rigide offert par les systèmes conventionnels de regroupement. Pour remédier à cela, des aspects nouveaux doivent s'ajouter à ceux déjà identifiés auparavant (voir section 2.2 du chapitre 2). Par exemple, un processus global d'organisation des connaissances doit intégrer les tâches de regroupement sur les diverses familles d'individus afin d'assurer le respect des relations entre ces familles. D'autres aspects déjà identifiés, tels que les outils de construction effective de classes, doivent être redéfinis afin de s'adapter aux descriptions d'objets qui se distinguent par leur hétérogénéité et leur forte structuration. Dans la suite, nous passons en revue les aspects de notre approche sur lesquels elle se distingue d'une approche traditionnelle. Ils sont énumérés en fonction de la granularité des connaissances auxquelles ils s'appliquent, allant d'une base entière à une simple valeur d'attribut.

3.2.1 Stratégie de construction conjointe de taxonomies

Les formalismes structurés de représentation de connaissances permettent de manipuler plusieurs familles d'individus au sein d'une base de connaissances. Ceux-ci sont organisés en des taxonomies de classes qui, tout en étant disjointes, peuvent être liées par des attributs relationnels inter-classes.

L'existence d'attributs relationnels entraîne des contraintes que le processus de construction des classes ne doit pas ignorer. En effet, la constitution de classes homogènes sur une famille d'objets peut nécessiter la comparaison d'objets sur d'autres familles liées par des relations. Plus encore, le résultat de cette comparaison ne peut être rendu explicite qu'en l'exprimant en terme de classes sur les objets liés. On en déduit que la construction de classes sur une famille repose sur l'existence de classes significatives sur les familles auxquelles celle-ci est liée par des attributs. Ainsi, lorsque plusieurs tâches de construction de taxonomies sont à effectuer au sein d'une base de connaissances, l'ordre de leur exécution n'est pas anodin. En effet, les classes construites en respectant les contraintes de référence entre familles d'objets reflètent mieux les connaissances disponibles et par là ont plus de chances d'être significatives pour le concepteur de la base. En conséquence, l'établissement de l'ordre d'exécution des regroupements doit être considéré comme une des étapes du processus global d'organisation de connaissances. L'aspect sous-jacent de l'approche sera désigné par *stratégie* de regroupement multi-familles faisant ainsi référence à son rôle de cadre pour l'application successive de méthodes de construction de taxonomies.

La stratégie de construction ne peut pas être réduite à un simple ordonnancement. En effet, il n'est pas toujours possible de trouver un ordre qui satisfait toutes les contraintes de référence entre familles. Dans beaucoup de cas, notamment en présence de références circulaires, des contraintes peuvent se révéler contradictoires. Cette apparente contradiction se résout si l'on admet que le regroupement peut s'appliquer sur plusieurs familles à la fois (voir le paragraphe suivant pour les implications sur la méthode de regroupement). Ainsi, le rôle de la stratégie est, avant tout ordonnancement, de discerner les groupes de familles qui sont, de par leurs liens, indissociables pour la construction de taxonomies. Par la suite, l'ordre sera établi sur l'ensemble de ces groupes.

En somme, la stratégie offre un cadre à la construction de classes sur un ou plusieurs concepts afin d'optimiser les efforts fournis par le concepteur : en matière d'ordonnancement entre les tâches de construction (l'ordre optimal est retrouvé automatiquement), ainsi qu'en matière d'interprétation des résultats (le plus grand nombre possible de connaissances validées par le concepteur est utilisé lors de chaque construction des classes). La stratégie que nous avons conçue est présentée par la section 3.4 de ce chapitre.

3.2.2 Méthode de construction de classes

Le paragraphe précédent a indiqué le besoin d'une méthode multi-familles, et donc multi-taxonomies, de construction de classes. Cela constitue une généralisation des méthodes existantes qui ne construisent qu'une seule taxonomie à la fois [14]. Dans le cas d'un seul concept, on retrouve le problème classique du regroupement, il peut donc être résolu par une méthode de regroupement qui satisfait les exigences citées dans la section 3.1.3.

Dans le chapitre 2 (voir sections 2.3.5 et 2.4.2), nous avons discuté des difficultés que les méthodes des différentes approches en regroupement rencontrent dans leur application aux données structurées. Nous avons fait le choix d'une méthode par proximité pour l'adapter aux objets, à cause essentiellement de la flexibilité qu'elle offre pour la prise en compte d'objectifs variables lors d'une construction de taxonomies. Elle a également l'avantage de s'adapter facilement au traitement de descriptions hétérogènes (voir le paragraphe suivant).

Compte tenu des exigences formulées précédemment, une méthode basée sur la proximité présente des avantages supplémentaires. Elle permet de décomposer le processus de construction de classes en deux étapes disjointes – correspondant au regroupement et à la caractérisation – construisant respectivement l'extension et l'intention des classes. De cette manière, le traitement des liens circulaires entre objets peut être confiné à la première étape, le regroupement par similarité, lors de laquelle les objets ne sont comparés que deux à deux et l'existence de classes sur leurs attributs relationnels n'est pas requise. La caractérisation peut ensuite se baser sur les groupes de forte cohésion préalablement formés pour en fournir une description sommaire.

La séparation des deux étapes de la construction va à l'encontre d'un des principes du regroupement conceptuel, à savoir l'évaluation des classes selon leur description. Cependant, nous considérons que c'est un prix à payer pour un traitement de la structure relationnelle libéré de toute restriction.

3.2.3 Outils de construction

Les outils sont les éléments de notre approche en contact direct avec les descriptions des objets opérant des transformations sur celles-ci. Dans notre cas, les outils nécessaires sont deux, une mesure de proximité pour la formation de groupes de forte cohésion et un opérateur de généralisation pour doter chaque groupe d'une description intentionnelle, en faisant ainsi une classe à part entière. Dans leur définition, la satisfaction des exigences de la section précédente doit être assurée.

Mesure de proximité

La mesure de proximité traduit un couple de descriptions d'objets en une valeur quantitative. Les principaux défis à relever à travers sa définition proviennent de la richesse et la structuration des descripteurs utilisés, l'intégration de connaissances préalables et la flexibilité de la comparaison (objectifs du regroupement, état de la base de connaissances). Parmi eux, la flexibilité peut être réalisée à moindre coût, en choisissant une mesure pondérée sur les attributs des objets.

14. Même si le système Labyrinth construit plusieurs taxonomies en parallèle, ce parallélisme est la conséquence du mode incrémental d'acquisition des individus. Il est imaginable que dans un mode «par lot», chaque construction de taxonomie sur une famille soit effectuée indépendamment des autres, en respectant l'ordre induit par la composition.

En revanche, le reste des exigences demande des efforts plus conséquents de formulation pour aboutir à une mesure convenable.

D'une part, la présence de relations dans les descriptions des objets exige une définition propre de l'écart sur un tel attribut. En même temps, la définition envisagée ne peut ignorer une certaine analogie qui existe entre les valeurs simples et les objets. Pour doter la nouvelle mesure d'un assis théorique, il est indispensable de dégager les aspects communs que valeurs et objets présentent afin d'y appuyer la définition. Ainsi, nous formulons un principe de comparaison (voir le paragraphe suivant) qui permet d'intégrer l'ensemble des connaissances descriptives sur les objets. Le même principe permet d'intégrer dans le calcul de la proximité les taxonomies existantes sur certains concepts.

D'autre part, le besoin de flexibilité par rapport à l'état de la base ne permet pas de favoriser l'un ou l'autre des aspects de la description des objets. En conséquence, nous proposons une définition dans laquelle les écarts sur l'un et sur l'autre peuvent être substitués à volonté. Qui plus est, une attention spéciale est attribuée à l'imbrication dans la structure des descriptions et aux possibles références circulaires entre objets.

Opérateur de généralisation

Compte tenu de la séparation nette entre les deux étapes de la construction de classes, beaucoup des exigences que nous avons formulées ne s'appliquent pas à la définition de l'opérateur de généralisation. Seule la complétude par rapport aux descriptions des objets nécessite une attention particulière.

3.2.4 Principes de la comparaison

La comparaison d'objets se caractérise par la grande variété des connaissances qui doivent être prises en compte. Pour la plupart, celles-ci sont exprimées dans les descriptions des objets, mais une minorité est liée aux éventuelles taxonomies de classes et au concept des objets comparés[15]. Cette variété motive la formulation d'un principe rigoureux régissant les contributions des divers aspects de la connaissance sur les objets comparés et la manière dont celles-ci sont combinées pour donner lieu à une valeur unique quantifiant les similitudes d'un couple d'objets. L'avantage d'un tel principe par rapport à la conception immédiate d'une mesure concrète est double. En effet, outre la garantie d'une prise en considération de toute la connaissance sur les objets, il dessine les contours de toute une famille de mesures possibles contribuant ainsi à une plus grande flexibilité dans le choix d'outils, et donc dans les résultats du processus global de regroupement.

Le principe que nous défendons ici repose sur une double vision sur les objets à comparer. Ceux-ci sont vus d'une part, comme des structures complexes, composées par leurs valeurs d'attributs, et d'autre part, comme des membres de leurs classes (si de telles classes sont connues). Avec une vision structurelle, la comparaison des objets est décomposée en un ensemble de comparaisons, chacun sur un attribut particulier. Avec la deuxième vision, la comparaison peut être effectuée de manière directe en tenant compte de la structure de la taxonomie. Un cadre unificateur, basé sur l'opération de généralisation, abstrait la partie commune des deux visions afin d'y fonder le calcul de la proximité. L'application conjointe des deux visions sur un couple d'objets suit une décomposition récursive et simultanée des structures de données sous-jacentes.

La description détaillée du principe de comparaison ainsi que des autres aspects de l'approche évoqués dans cette section est donnée dans la suite du chapitre.

15. Dans notre cadre, seuls des objets du même concept peuvent être comparés.

3.3 Principes de comparaison d'objets

Le défi principal de la construction de classes dans les objets est la comparaison d'objets dont les descriptions font apparaître une grande variété d'attributs y compris des attributs relationnels faisant référence à d'autres objets. La structure induite par les références entre objets, même si elle ressemble fortement aux individus complexes dans KBG, ne favorise pas un traitement à la KBG. Outre la nature asymétrique des relations inter-individus dans le modèle à objets, ici l'accent est mis sur un couple particulier d'objets[16] et non pas sur les structures entières. Une comparaison plus adéquate consiste à explorer, en partant des objets à comparer et en allant dans le sens des références, les structures respectives et cela d'une manière progressive permettant de s'arrêter lorsque les objets atteints n'ont qu'une importance négligeable pour le couple initial. Cette démarche reflète mieux la régularité dans les structures induites par les deux objets à comparer induite par le typage strict des attributs (voir chapitre 1). Sa réalisation repose sur deux manières de confronter un couple d'objets, chacune correspondant à un aspect complémentaire de la connaissance sur les objets, réunies par un principe de comparaison unique basé sur la généralisation.

Les deux visions complémentaires sur les objets ainsi que le principe de comparaison qui permet de les réconcilier sont décrites dans ce chapitre.

3.3.1 Les deux visions sur les objets

Notre approche est fondée sur deux visions complémentaires sur les objets d'une base de connaissances. Alors que l'une se concentre sur la description des objets en termes d'attributs, l'autre met l'accent sur leur appartenance à des classes dans une taxonomie. La comparaison entre objets est faite en combinant les deux visions qui mènent à des manières spécifiques d'estimation pour les ressemblances entre les objets. Dans la suite nous présentons les deux visions, puis la manière dont elles interviennent dans la construction de classes.

Le concept et son espace de description

Un concept TROEPS a été défini (voir chapitre 1) comme la représentation d'une famille d'individus du domaine modélisé. Chaque objet instance du concept est la représentation d'un individu de la famille. Ainsi, le concept peut être vu comme un ensemble d'objets dont les classes définissent des sous-ensembles significatifs.

En même temps, les objets possèdent une description dont les parties constituantes, les attributs, sont définis au niveau du concept. En ce sens, le concept définit un espace de description dont chacun des points est la description d'une instance potentielle. L'espace est multidimensionnel, chaque dimension correspondant à un des attributs. Par exemple, le concept `chercheur` de l'exemple précédent définit un espace dont les dimensions correspondent au nom, l'âge, les articles publiés, etc. Alors que certaines dimensions sont des types de données, elles décrivent donc des ensembles de valeurs, d'autres, correspondant à des concepts, décrivant des ensembles d'objets (par exemple la dimension projet de rattachement de l'espace du concept `chercheur`). Un tout autre type de dimension est constitué par les attributs multivalués (membres d'un `projet`).

La figure 3.4 (à gauche) fournit une manière approximative de visualiser l'espace de description défini par un concept k. Celui-ci est présenté comme un espace à dimensions numériques. L'analogie avec $\mathbb{R}^n$ est forte car chaque instance présente, potentiellement, tous les attributs définis par le concept et seulement ces attributs. Ainsi, un objet correspondrait au point dont

16. Les objets qui sont à l'origine des structures.

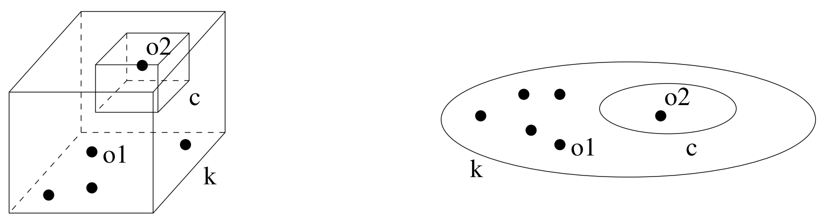


FIG. 3.4: *À gauche, espace de descriptions associé à un concept k. Chaque dimension correspond à un attribut, chaque instance à un point de l'espace et chaque classe à une région. Les objets membres d'une classe se situent dans la région décrite par celle-ci. À droite, le concept vu comme un ensemble d'entités individuelles, les classes sont des sous-ensembles et les objets sont les entités membres. L'objet o_2 appartient à la classe c, alors que l'objet o_1 n'y appartient pas. Sur les deux dessins, o_2 est inclus dans la région correspondante à c, alors que o_1 en est exclu.*

la coordonnée sur chaque dimension représentant un attribut est la valeur de cet attribut dans l'objet. De plus, les classes prises avec leurs descriptions, définissent des parties de l'espace et tout point correspondant à un objet membre d'une classe est situé à l'intérieur de la partie sous-jacente comme le montre la figure 3.4. Il est même possible de retrouver les restrictions de valeurs que les classes définissent sur les attributs des objets membres : celles-ci sont des «projections» de la partie de l'espace sur l'axe correspondant à l'attribut sous-jacent.

Cependant, toute analogie avec un espace numérique s'arrête là : le raisonnement traditionnel basé sur les notions de voisinage, contiguïté, hyper-rectangles, etc., bien qu'attirant n'est plus fondé. En effet, seuls les attributs totalement ordonnés admettent un tel raisonnement, alors que pour le reste des attributs la notion d'axe ne signifie rien d'autre qu'un ensemble de valeurs possibles.

Vision structurelle

Nous avons eu l'occasion de souligner par multiples reprises les avantages de la représentation *structurée* que les objets permettent de réaliser. Lorsqu'il est question de TROEPS, la structuration acquiert un sens nouveau, celui du *typage*. En effet, de point de vue de la théorie des types, les concepts TROEPS constituent la définition des nouveaux types de données. Les valeurs du type correspondent aux descriptions des objets du concept. Elles sont des n-uplets dont chaque champ correspond à un attribut et comporte sa valeur pour l'objet sous-jacent. Dans la suite, nous allons négliger les identificateurs des objets qui n'ont pas d'importance pour la comparaison, et confondre un objet avec sa description. Autrement dit, l'objet sera réduit au n-uplet constitués par les valeurs de ses attributs. Ainsi, les instances du concept `chercheur` seront assimilées à des n-uplets dont les trois premiers constituants sont de type chaîne de caractères, entier et entier correspondant respectivement au nom, à l'âge et au nombre des articles publiés.

Il est certes imaginable que les n-uplets d'un type particulier soient constitué exclusivement par des valeurs simples, cependant la plupart du temps, certains constituants des n-uplets vont avoir une structure propre. Cette structure peut être celle d'un n-uplet ou bien d'une collection. Les membres de la collection ou les constituants du n-uplet peuvent à leur tour être structurés et ainsi de suite. Le degré d'imbrication dans les objets d'une base de connaissances dépend des liens entre entités du domaine modélisé que l'on veut exprimer. Par exemple, dans le n-uplet de type `projet`, le constituant `membres` contient un ensemble de n-uplets de type `chercheur`.

Pour un concept donné, les n-uplets que constituent ses instances peuvent être progressivement décomposés en leurs constituants. En effet, en traversant un niveau d'imbrication à chaque pas, c'est-à-dire en éliminant un constructeur de n-uplet ou de collection, la décomposition va finalement aboutir à un ensemble de valeurs simples et donc indécomposables. Si les descriptions compostent des circuits, à cet ensemble de constituants atomiques s'ajoutera un ensemble de n-uplets du concept de départ. Par exemple, si dans la description d'un chercheur on indique son appartenance à un projet (moyennant un attribut), alors la décomposition d'un n-uplet de `chercheur` va aboutir à un ensemble d'autres n-uplets de ce type correspondant à tous les chercheurs qui partagent des projets, un ou plusieurs, avec celui-ci. Inversement, chaque n-uplet peut être vu comme le résultat de l'application successive de constructeurs sur un ensemble de valeurs simples. La composition de l'ensemble de valeurs simples, ainsi que l'ordre d'application des constructeurs qui mène à la constitution d'un n-uplet « valide » ne sont pas arbitraires : ils obéissent aux contraintes exprimées dans les descriptions des concepts.

La régularité dans les n-uplets devient une évidence lorsque l'on représente la structure des descriptions d'objets sous forme de graphe orienté. Dans ce graphe, les nœuds sont soit des valeurs simples soit des descriptions construites, n-uplets ou collections. Les arcs du graphe représentent les liens entre une description construite et ses constituants. Ainsi, chaque n-uplet est lié à ses constituants et chaque collection à ses membres. Le graphe correspondant à la décomposition de l'objet représentant le projet `Rotor` est dessiné sur la figure 3.5. Pour cette visualisation nous avons utilisé un formalisme graphique que est inspiré de [Abiteboul *et al.*, 1995].

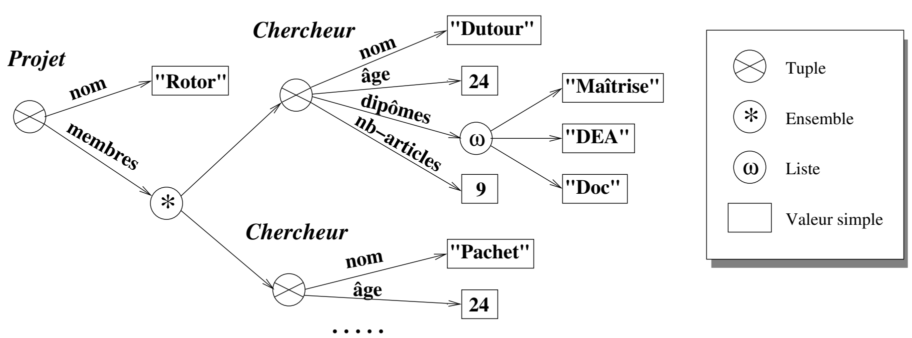


FIG. 3.5: *La décomposition de l'objet* `Rotor` *: le n-uplet est composé de deux valeurs dont une simple, le nom, et l'autre construite. La seconde valeur est un ensemble de n-uplets de type* `chercheur` *qui sont également décomposés. À noter la présence d'un constituant liste qui représente les diplômes détenus par un chercheur.*

Il est à noter qu'un raisonnement identique s'applique aux classes qui possèdent leurs propres descriptions composées par les restrictions de valeur. Chaque restriction est vue comme un ensemble de valeurs, les valeurs admissibles. Pour aboutir à un n-uplet complet par rapport au type défini par le concept, il suffit de compléter avec une restriction fictive pour chaque attribut que la classe ne restreint pas de manière explicite. Pour une classe c, ette restriction est la conjonction de toutes les restrictions définies par les super-classes de c et, dans le cas extrême, peut correspondre au co-domaine de l'attribut tel que défini par le concept (aucune restriction).

La décomposition récursive des objets peut être utilisée dans leur comparaison. Ainsi, pour comparer deux objets, il faudra définir des fonctions de comparaison à deux niveaux : sur les valeurs simples et sur les constructeurs.

Vision ensembliste

Avec la vision ensembliste, les objets ne sont plus considérés comme des n-uplets de valeurs, mais comme des entités atomiques. Les objets n'ont donc pas de structure interne et les concepts ne sont que des ensembles d'entités. Cependant, un concept peut être muni d'une autre sorte de structure, notamment en considérant l'ensemble de classes qu'il admet. Chaque classe décrit un ensemble d'objets, sous-ensemble de l'extension du concept. Par conséquent, à la taxonomie de classes ordonnées par la spécialisation correspond une collection d'ensembles liés par l'inclusion.

Une taxonomie n'offre en soi qu'une base relativement pauvre pour la comparaison d'objets. En effet, on peut voir la taxonomie comme le domaine d'une fonction de comparaison très rudimentaire faisant correspondre à un couple d'objets leur classe commune la plus spécifique. Afin de traduire ce principe en une valeur numérique, on pourrait préciser une application des classes vers les nombres positifs. Parmi les variantes d'une telle application, citons la cardinalité de l'extension d'une classe. Celle-ci peut être l'extension effective (les objets présents dans la base) ou potentielle (tous les objets pouvant être membre de la classe).

Les deux visions sur les objets, en tant que structures d'attributs et en tant que membres de classes, sont comparables à ce qui est appelé l'*interprétation intentionnelle* et l'*interprétation extensionnelle* des objets dans [Capponi *et al.*, 1995].

3.3.2 Vision structurelle et comparaison d'objets

Les objets sont assimilés à leurs descriptions construites au-dessus d'un ensemble de valeurs simples à l'aide de constructeurs. Celles-ci peuvent être représentées sous forme d'un graphe[17] dont les nœuds terminaux sont des valeurs simples et le reste des descriptions construites, n-uplets ou collections. Un tel graphe contient la totalité des connaissances descriptives sur un objet et par conséquent constitue un bon support pour la comparaison d'objets. En effet, la comparaison consiste en une mise en correspondance de deux graphes (ou des sous-graphes).

On observe une certaine analogie avec l'approche de KBG qui est basée également sur la transformation des descriptions des individus en une structure graphique. Cependant, des différences considérables existent entre les deux structures et elles sont reflétées dans les méthodes d'exploration qui déterminent le résultat de la comparaison. Ainsi, d'une part, les graphes que nous considérons sont orientés, alors que dans KBG ils ne le sont pas. De plus, la présence de constructeurs en tant que nœuds enrichit la structure, mais en même temps rend son exploration plus complexe. Finalement, le but de la comparaison dans notre cas n'est pas la détection de sous-structures communes entre graphes, mais sur l'établissement d'un degré de ressemblance entre deux nœuds.

Toutes ces particularités du modèle à objets déplacent le focus de la comparaison : celui-ci n'est plus mis sur un appariement optimal entre deux graphes, à la KBG, car le typage permet d'établir un tel appariement de manière directe. En revanche, la question de la contribution des paires de nœuds appariés dans le résultat global doit être considérée. Le principe de notre solution prévoit un traitement distinctif pour les valeurs simples et les descriptions construites qui revient à définir deux types de fonctions. Les fonctions d'*écart* sont calculées sur les constituants d'une description construite, alors que les fonctions d'*agrégation* sont appliquées sur les descriptions même. Une fonction d'agrégation ne fait que de combiner les valeurs rendues par les écarts en une seule valeur. Pour leur part, les écarts peuvent être des fonctions sur des valeurs simples ou sur des descriptions construites. Dans ce second cas, on leur substitue des fonctions d'agrégation.

17. En réalité, il s'agit d'un multi-graphe.

Une différence notable existe entre fonctions d'agrégation sur les n-uplets et sur les collections. Les premières combinent un nombre fixe de résultats qui ne dépend que du type du n-uplet. Les secondes s'appliquent sur un ensemble de résultats dont la taille peut varier fortement en fonction de la taille des collections.

Compte tenu des remarques précédentes, le principe de la comparaison pourrait être formulé ainsi : la comparaison consiste en une décomposition récursive et simultanée des structures de n-uplets. À chaque couple de constituants simples rencontré, une fonction d'écart est appliquée selon son type. À chaque couple de constituants structurés, une fonction d'agrégation est appliquée. Le typage assure que le principe est toujours applicable et le calcul abouti, si les conditions suivantes sont respectées :

– les valeurs manquantes ($\imath$) sont admises dans le calcul,
– une fonction de composition peut être substituée à un écart.

Avec des descriptions circulaires, la décomposition n'aboutit pas toujours à un ensemble de comparaisons de valeurs simples. Dans certains cas, elle peut conduire à la comparaison des structures du départ. Les travaux sur le système KBG ont démontré que même en présence de dépendances récursives, les similarités entre objets peuvent être quantifiées. Il est nécessaire pour cela d'utiliser une fonction d'agrégation qui admet un mode de calcul récursif. Cependant, ce type de calcul est très coûteux et le prix à payer croît vite lorsque le nombre de couples d'objets à comparer augmente. Il est donc à employer avec prudence dans une base de connaissances ayant de nombreuses relations inter-objets.

Une technique réduisant le coût du calcul de similarité exploite la notion de distance entre couples d'individus, mesurée en nombre de relations. En effet, on observe le phénomène suivant : plus la chaîne d'attributs séparant deux couples d'individus est longue, plus l'influence que chacun exerce sur la similarité de l'autre est faible. En conséquence, la précision que la prise en compte de couples éloignés peut apporté à la valeur de la similarité d'un couple donné devient négligeable à partir d'un certain seuil. Afin de réduire le prix du calcul récursif, Bisson propose d'utiliser dans ce cas une fonction *terminale* qui arrête l'exploration des relations inter-individus et produit une valeur approximative sur le couple limite.

Notre propre technique permettant de comparer efficacement les couples objets fortement reliés entre eux est basée sur la combinaison des deux visions, la structurelle et l'ensembliste, dans le calcul de dissimilarité.

3.3.3 Vision ensembliste et comparaison d'objets

La vision ensembliste fait abstraction de la description des classes et des objets et ne les considère qu'en tant qu'ensembles et éléments d'ensembles respectivement. Les seules informations exploitables pour la comparaison entre objets/classes deviennent les relations d'identité, d'appartenance et d'inclusion. Ce nouveau cadre est plus pauvre que le cadre structurel et par la n'admet pas de comparaison entre objets aussi précises. Cependant, des bonnes raisons existent pour que la comparaison entre objets y soit appuyée. En effet, une classe n'existe que par le fait que ses membres partagent certains points communs, une similarité particulière exprimée par l'appartenance à la classe. Cette similarité n'est pas nécessairement corrélée avec les descriptions des objets. Les classes fournissent donc une base tout aussi légitime pour la comparaison.

En outre, la comparaison basée sur la taxonomie dans de nombreuses situations est beaucoup plus efficace que celle basée sur les descriptions car elle n'entraîne pas de décompositions récursives, tout au plus une recherche de la classe commune dans la taxonomie. Pour cette raison, elle pourra présenter un meilleur compromis entre précision et coût que la décomposition exhaustive.

Les classes d'appartenance sont donc une bonne alternative aux descriptions des objets, mais encore faut-il que ces classes existent. Autrement dit, lorsque la comparaison d'objets d'un concept doit porter sur les classes, il faut s'assurer que celles-ci sont déjà disponibles ou qu'elles pourront être construites, manuellement ou automatiquement, au préalable. Nous avons vu dans les sections précédentes que lors de la comparaison des objets sur un concept il est impossible de garantir l'existence de classes pertinentes sur tous les concepts qui lui sont liés par des attributs relationnels. C'est ici notamment la tâche d'une stratégie : elle doit d'abord délimiter parmi les concepts qui nécessitent une construction ceux qui peuvent être traités au préalable de ceux qui ne le peuvent pas, puis appliquer la construction sur les premiers de telle sorte qu'un maximum de taxonomies soient disponibles au moment de la comparaison.

3.3.4 Combiner les deux visions

Les deux visions décrites ci-dessus présentent une forte complémentarité car chacune est basée sur un aspect spécifique de la connaissance sur les individus représentés. La même complémentarité peut être constatée sur le plan calculatoire : alors que la vision structurelle permet une comparaison selon un nombre potentiellement grand de critères, la vision ensembliste ne se concentre que sur une structure unique, la taxonomie[18]. Ainsi, une mesure structurelle apportera à la comparaison une plus grande précision (par rapport aux descriptions des objets), tandis qu'une mesure ensembliste, n'impliquant pas de décompositions récursives, sera moins coûteuse en temps de calcul.

La double complémentarité plaide en faveur d'une comparaison mixte qui combine les avantages des deux visions. On pourra noter que lorsqu'il s'agit de comparer les objets d'un concept c dans le but d'en extraire des classes, seule une mesure structurelle est applicable au niveau de c. Cependant, une application stricte de la vision structurelle conduirait, en partant d'un couple d'instances de c, à un ensemble d'autres comparaisons dont une partie porte sur des couples d'objets (des objets qui se trouvent dans les deux graphes de décomposition respectifs). Parmi ces couples d'objets, un certain nombre peuvent appartenir à des concepts dont (au moins) une structure de classes est déjà disponible. Il est donc possible de substituer, sur ces couples, une mesure ensembliste à la mesure purement structurelle. Cette substitution aura pour effet la production d'un résultat immédiat qui ne fait pas référence à la description des objets, elle arrête donc l'exploration de la structure des objets.

La mesure substituée ne produira probablement pas un résultat aussi précis qu'une comparaison structurelle car la vision ensembliste ne distingue pas entre les couples d'objets partageant la même classe commune la plus spécifique. Cependant cette perte de précision doit être relativisée comme nous l'avons indiqué plus haut.

Nous passons outre le choix de la vision à appliquer sur les objets d'un concept particulier car ce choix tient à l'appréciation d'une taxonomie existante. Ainsi, le concepteur de la base, celui qui évoque la construction automatique de classes, est le mieux placé pour juger de la pertinence d'une taxonomie par rapport aux objectifs du regroupement et pour décider si la comparaison d'objets peut s'y appuyer. Notre objectif prochain est de clarifier les fondements d'une combinaison des deux visions. Dans la suite, nous nous emploierons donc à définir un cadre unificateur pour ces deux visions permettant d'en extraire les traits communs puis d'y fonder les fonctions concrètes de comparaison.

18. Bien que des variantes multi-taxonomies de la mesure peuvent être envisagées, pour l'instant nous nous limitons à la considération d'une taxonomie unique.

3.3.5 Un cadre unificateur

Pour définir ce cadre, nous nous sommes inspirés des travaux sur les *Systèmes Classificatoires* [Capponi *et al.*, 1995]. À la base de l'approche est l'analogie entre les notions de ADT et de concept TROEPS qui constituent des ensembles de valeurs membres, valeurs simples et objets respectivement, munis d'une collection de sous-ensembles significatifs, types et classes respectivement. En réalité, cette analogie peut être prolongée sur la comparaison entre membres. Ainsi, la comparaison au sein des ADT, qui est par ailleurs fondamentale pour la comparaison des objets, peut être calquée sur la généralisation, tout comme sous la vision ensembliste appliquée aux concepts. Plus particulièrement, la ressemblance entre deux valeurs simples peut être estimée en fonction du type commun le plus spécifique qu'elles admettent.

On remarquera que dans les deux cas il s'agit d'évaluer la taille d'une généralisation commune, ce qui pose les fondements du cadre recherché. En effet, la comparaison envisagée d'objets en tant que structures pourra s'y ranger facilement. Le cas de comparaison d'objets sans relations est immédiat : celle-ci est décomposée en un ensemble de comparaisons de valeurs simples dont chacune considère la généralisation la plus spécifique des deux valeurs respectives. Or, l'ensemble de ces généralisations constitue un n-uplet compatible avec le concept des objets, c'est-à-dire qu'il définit une description de classe sur ce concept, et, qui plus est, cette classe est la classe la plus spécifique que les deux objets admettent. En conséquence, la valeur produite par la mesure structurelle à partir des écarts sur les valeurs simples, et cela quelque soit la fonction exacte employée, constitue une évaluation de la classe la plus spécifique. Il est à noter que cette valeur est calculée sur la base des «projections» de la classe sur chacun des attributs du concept.

Afin d'étendre ce raisonnement aux objets ayant des attributs relationnels, il faut considérer les deux visions possibles sur un tel attribut. Si la vision employée sur les objets du co-domaine d'un attribut est ensembliste, alors notre affirmation est toujours valable car les objets en question sont comparés par rapport à leur généralisation. Si en revanche la structure des objets est importante, on est obligé de procéder par une sorte d'induction sur le nombre d'indirections entre objets qui démontre que le résultat de la comparaison des objets initiaux est toujours une caractéristique de la classe la plus spécifique que ceux-ci admettent. Bien évidemment, cette induction ne sera correcte que si les liens inter-objets sont non-circulaires. En réalité, dans le cas circulaire, la notion de classe la plus spécifique pour deux objets donnés n'est pas bien fondée. Cependant, comme nous allons le voir dans le Chapitre 7, une définition en point-fixe est possible. Nous reléguons donc la discussion sur ce problème au chapitre en question en nous contentant de conclure sur l'adéquation totale du cadre offert par la généralisation pour les fins de la comparaison des objets, et cela, quelque soit la complexité structurelle de ces objets.

3.3.6 Synthèse

Pour résumer les réflexions jusqu'à cet endroit, rappelons que la comparaison des objets ayant des descriptions complexes, c'est-à-dire faisant intervenir des valeurs simples et des constructeurs, nécessite la définition de deux sortes d'opérateurs :

- opérateurs de base sur les types de données,
- opérateurs génériques sur les constructeurs capables de «résumer» le résultat des opérateurs sur les constituants.

Puisque les applications emboîtées de constructeurs de type différent sont possibles (par exemple des collections d'objets et, surtout, des objets dont les valeurs de certains attributs sont des objets) une certaine «composabilité» entre les opérateurs de traitement est exigée en premier lieu.

Concernant les deux visions sur les objets exposées dans cette section, elles permettent d'inclure dans le processus de construction de taxonomie deux sortes de connaissances sur les objets. En même temps, elles offrent des modes de comparaison différentes. Ainsi, on pourra naviguer entre les deux visions afin d'aboutir à des résultats plus variés et avec une meilleure chance de satisfaire l'utilisateur de l'outil de construction de taxonomies. Il est à noter que les deux visions rencontrent des difficultés lorsque les objets comportent des relations circulaires.

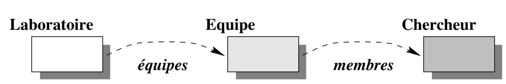
Dans la suite, nous exposons une stratégie, puis une méthode de construction de taxonomies qui permet d'aller au-delà de ces difficultés.

3.4 Stratégie de construction conjointe de classes dans une base de connaissances à objets

La notion de *stratégie* désigne ici l'ordre particulier dans lequel les différentes familles d'objets seront traitées. La stratégie est le produit d'un calcul automatique dont le point de départ est constitué par les dépendances entre familles que les attributs relationnels provoquent. Elle ne retrouve sa justification que dans la nécessité de traiter des descriptions circulaires, ce qui expliquerait le fait qu'une telle notion est complètement absente du vocabulaire des systèmes de regroupement actuels.

3.4.1 Stratégie et circularité

La première idée qui vient naturellement lorsqu'on est confronté au besoin de construire des taxonomies sur plusieurs familles d'objets est de les parcourir dans l'ordre induit par les relations entre familles d'objets. Par exemple, sur la figure ci-dessus, cette stratégie reviendrait à construire des classes d'abord sur les chercheurs, ensuite sur les équipes, puis à la fin sur les laboratoires. Une telle stratégie ne peut s'appliquer avec succès que lorsque les relations entre les différentes



familles d'objets sont sans circuits. Toutefois, dans beaucoup de domaines, cette restriction ne semble pas poser de vrais problèmes.

Par exemple, dans [Thompson et Langley, 1991] les auteurs décrivent le parcours utilisé par le système LABYRINTH, pour l'intégration d'un nouvel individu complexe comme un « parcours du graphe des parties en profondeur d'abord et post-fixe» (voir la section 2.5 du chapitre 2). Dans ce cas particulier, le graphe des familles d'objets est un arbre induit par la relation de composition qui est manifestement sans circuits[19]. La même stratégie est utilisée dans [Ketterlin *et al.*, 1995], cette fois-ci les individus peuvent avoir des relations arbitraires, à condition que celles-ci ne provoquent pas de circuits.

Quant aux formalismes structurés de RC, ceux-ci admettent souvent la circularité et donc son traitement lors de la construction des classes est au moins une question qui doit être débattue. Pour notre part, nous nous attachons à l'idée, exprimée par B. Nebel dans sa thèse [Nebel, 1990], que les descriptions circulaires ont leur fondement dans le monde réel et que cela constitue une raison suffisante pour qu'elles soient manipulées de manière appropriée, c'est-à-dire sans

19. Bien que des relations autres que la composition soient admises dans les données de LABYRINTH, celles-ci n'interviennent pas dans la stratégie, car leur impact sur la construction des classes est très réduit.

chercher à les réduire à des descriptions non-circulaires. En effet, d'après Nebel, on pourra citer de multiples contextes, dans lesquels ce genre de concepts se présente naturellement :

- une personne a deux parents qui sont des personnes,
- un homme peut être marié à une femme, ou plus généralement, une personne peut avoir pour conjoint une autre personne,
- une personne habite dans un appartement (ou maison) possédé par un propriétaire qui peut être lui-même une personne.

Trois types de concepts à circularité sont distingués dans [Nebel, 1990] :

- structures de données de taille finie définies récursivement, par exemple les arbre ou une liste chaînée,
- des relations inter-objets infinies, par exemple la relation enfant parent sur le concept `humain`,
- des relations circulaires sur les objets, par exemple la relation conjoint.

Ici, nous ne nous intéressons qu'aux cas finis, nous allons donc nous concentrer sur les cas un et trois ci-dessus. Il convient de les distinguer car dans le premier cas les circularités se limitent aux classes que l'on peut construire sur les objets, sans que des circuits entre objets se créent nécessairement. En revanche, dans le troisième cas, ce sont des circuits dans les descriptions des objets qu'il faudrait traiter. Ce dernier cas est source de plus de difficultés, car il inclut, en quelque sorte le premier, mais ne se réduit pas à celui-ci (voir chapitre 5 de [Nebel, 1990]).

Afin de concrétiser nos réflexions et de pouvoir formuler une stratégie en termes stricts, nous allons considérer les familles d'objets comme les nœuds d'un graphe orienté dont les arcs sont les relations entre familles.

3.4.2 Graphe des dépendances entre concepts

La structure graphique que nous allons considérer dans le paragraphe est définie sur les concepts d'une base de connaissances liés par leurs attributs relationnels. Soit :

- K l'ensemble des concepts,
- A l'ensemble des attributs et A^r le sous-ensemble des attributs relationnels (liens de relation et de composition).

Tout d'abord, nous proposons une expression relationnelle de la dépendance qu'un attribut relationnel crée entre le concept qui le définit, ou *source*, et le concept dont il prend ses valeurs, ou *destination*. Ainsi, nous allons noter par $\rightharpoonup \subseteq K \times K$ la relation d'*atteignabilité directe* entre concepts.

Définition 3.1. *$k \rightharpoonup k'$ ssi $k.a \subseteq k'$ ou $k.a \subseteq 2^{k'}$ ou encore $k.a \subseteq \Lambda^{k'}$ pour un attribut $a \in A^r$.*

Intuitivement, un concept k' est atteignable depuis un concept k si k' est le type de base d'un ou plusieurs attributs a de k. Soulignons que a peut être monovalué, mais aussi multivalué (les cas $k.a \subseteq 2^{k'}$ et $k.a \subseteq \Lambda^{k'}$). La relation d'*atteignabilité* entre concepts, notée $\rightharpoonup^+ \subseteq K \times K$, est la réduction transitive de $\rightharpoonup$.

Si deux concepts sont mutuellement atteignables, c'est-à-dire si chacun participe, directement ou pas, dans la définition de l'autre, nous allons dire qu'ils sont *co-définis*. La relation de co-définition entre concepts, $\rightleftharpoons \subseteq K \times K$, est la fermeture symétrique de l'atteignabilité $\rightharpoonup^+$.

Définition 3.2. *$k \rightleftharpoons k'$ ssi $k \rightharpoonup^+ k'$ et $k' \rightharpoonup^+ k$*

La relation de co-définition est manifestement une relation d'équivalence sur l'ensemble des concepts participant au moins à une relation de co-définition. La dernière restriction veut dire en particulier qu'un concept n'est pas automatiquement en co-définition avec lui-même.

Dans ce contexte, nous pouvons maintenant étudier les ensembles maximaux de concepts qui se réfèrent mutuellement dans leurs descriptions. Ces ensembles sont les classes d'équivalence pour la relation $\rightleftharpoons$. Pour en donner une définition exploitable, considérons la structure de graphe induite par $\rightharpoonup$ sur $\mathcal{K}$.

Le *graphe des dépendances entre concepts* est composé par les concepts et les attributs relationnels d'une base de connaissances. Les premiers sont les sommets et les seconds les arcs. Les arcs sont étiquetés par le nom de l'attribut sous-jacent.

Définition 3.3. *Le graphe des dépendances entre concepts $\mathcal{K}$ est un multi-graphe orienté étiqueté dont les sommets sont des concepts et les arcs sont les attributs relationnels :*

$$\mathcal{K} = \langle V_{\mathcal{K}}, E_{\mathcal{K}}, A_{\mathcal{K}} \rangle$$

où

$$V_{\mathcal{K}} = K, \qquad A_{\mathcal{K}} = A^r, \qquad E_{\mathcal{K}} \subseteq V_{\mathcal{K}} \times A_{\mathcal{K}} \times V_{\mathcal{K}}$$

et

$$\langle k, a, k' \rangle \in E_{\mathcal{K}} \textit{ ssi } \exists a \in A^r ; k.a \subseteq k' \text{ , } k.a \subseteq 2^{k'} \textit{ ou } k.a \subseteq \Lambda^{k'}$$

Deux concepts sont en relation d'atteignabilité si et seulement si un chemin orienté les lie dans le graphe. Dans notre contexte le chemin peut être défini par la succession des étiquettes de ses arcs.

Définition 3.4. *Entre deux concepts k et k' il existe un chemin $\langle a_1, ...a_n \rangle$ dans $\mathcal{K}$ ssi pour tout i de 1 à n*

$$\exists a_i \in A_{\mathcal{K}} \textit{ et } \exists k_i \in V_{\mathcal{K}}$$

tels que

$$\langle k_{i-1}, a_i, k_i \rangle \in E_{\mathcal{K}} \textit{ et } k_0 = k,\ k_n = k'.$$

Lors de la construction d'une taxonomie pour un concept k, seule une partie du graphe nous intéressera, celle qui est atteignable depuis k ou encore les concepts k' qui se trouvent liés à k par des séquences d'attributs relationnels. Celui-ci est composé par les concepts atteignables depuis k. Le sous-graphe de $\mathcal{K}$ atteignable depuis un concept k, sera noté $\mathcal{K}(k)$.

Définition 3.5. *Le graphe $\mathcal{K}(k)$ est un sous-graphe de $\mathcal{K}$ tel que :*

$$\mathcal{K}(k) = \langle V_{\mathcal{K},k}, E_{\mathcal{K},k}, A_{\mathcal{K},k} \rangle$$

où

$$k' \in V_{\mathcal{K},k},$$

ssi

$$\exists n \in \mathbb{N} \textit{ et } a_i \in A^r \textit{ ; tels que } k' \textit{ est lié à } k \textit{ dans } \mathcal{K} \textit{ par un chemin } \langle a_1, ..., a_n \rangle$$

Notons que pour étendre la notion de graphe de dépendances sur les points de vue, il faudra définir le sous-graphe atteignable depuis k par des chemins dont le premier porte une étiquette particulière, prise dans un sous-ensemble $A^P \subseteq A^r$: $\mathcal{K}(k, A^P)$. Toutefois, pour simplifier, nous allons considérer que chaque concept n'est donné que par un seul point de vue et celui-ci décrit la totalité des attributs du concept.

3.4.3 Impact de la circularité sur la construction des classes

Ayant justifié le besoin de traiter directement les liens circulaires entre les familles d'objets, il convient de s'interroger si ceci ne compromet pas notre effort de définir une stratégie, et donc un ordre entre les tâches de construction.

Dépendances entre concepts co-définis

Les circuits empêchent tout établissement d'un ordre total entre concepts : si les objets de concepts co-définis, soit k et k', doivent être organisés en classes, il est impossible de fixer un ordre de construction. Le problème est que chacune des classes recherchées sur l'un des concepts, soit k, devra faire référence à une ou plusieurs classes sur l'autre concept, k'. L'inverse est également vrai. L'établissement de ces références croisées est nécessaire à la caractérisation des classes, qui est à son tour une partie intégrale du processus de construction de classes. Il en ressort qu'afin de construire des classes complètement caractérisées sur l'un des concepts, il faut déjà avoir ces classes en main. Ce résultat paradoxal devient un obstacle surmontable si l'on se permet de transgresser certaines règles qui semblent faire l'unanimité dans le domaine du regroupement conceptuel.

Constructions conjointes de classes

D'une part, nous avons considéré jusqu'ici que les constructions de classes sur différentes familles sont des tâches indépendantes, disjointes dans le temps. En effet, les classes sur deux familles différentes étant disjointes, il est naturel lors d'une construction automatique de ne considérer qu'une famille à la fois. Toutefois, comme nous l'avons vu, les tâches de construction de classes sur des familles co-définies ne peuvent être dissociées. C'est l'impossibilité d'établir un ordre temporel entre ces tâches qui a conduit au résultat négatif ci-dessus. Or, rien n'interdit de construire les classes sur des familles co-définies de manière *simultanée*.

Séparation des étapes de regroupement et de caractérisation

D'autre part, nous avons pris comme présupposé l'indivisibilité de la tâche de la construction des classes. Cela veut dire que les étapes de regroupement et de caractérisation sont exécutées dans leur suite naturelle, sans que d'autres processus viennent s'intercaler entre les deux. En effet, c'est ainsi que le processus de regroupement conceptuel est décrit depuis l'avènement de la discipline [Fisher, 1987]. Mieux encore, la fusion des deux étapes, c'est-à-dire le regroupement guidé par la caractérisation, est parmi les principes de base de la majorité des méthodes existantes. Or, nous pouvons imaginer que le processus de construction puisse être séparé en ses constituants, le regroupement et la caractérisation, qui deviennent des étapes indépendantes. Qui plus est, ces étapes peuvent être exécutées sur plusieurs familles d'objets à la fois : regroupement sur toutes les familles, puis caractérisation.

De cette manière, nous séparons la formation effective des classes de leur description, l'étape qui nécessite obligatoirement l'existence de classes sur tous les attributs des objets. Ainsi, on peut construire des classes sur un ensemble de familles co-définies, à condition de traiter toutes ces familles simultanément.

Une stratégie qui minimise l'effort de construction

La méthode sera décrite dans la section suivante. Ici notons seulement qu'il ne sera pas très efficace de traiter systématiquement toutes les familles d'une base de connaissances pour la construction des classes. Au contraire, il convient de restreindre au maximum la taille de l'ensemble total des objets considérés par une construction, tout en gardant ensemble les familles d'objets co-définies. Il en ressort que la stratégie doit assurer le traitement à un moment donné du « bon » ensemble de familles. Cet ensemble comporte toutes les familles qui sont en co-définition avec une famille donnée et seulement celles-ci. En termes mathématiques, cet ensemble constitue une composante fortement connexe (CFC) du graphe décrit dans le paragraphe précédent.

La tâche de la stratégie devient donc la recherche des CFC du graphe, puis l'établissement d'un ordre entre celles-ci. Ensuite, le traitement consiste à appliquer la méthode de construction de classes au sein d'une CFC. Cette fois-ci, aucun circuit ne saura empêcher l'application de la méthode car le graphe des CFC est un DAG. De plus, cette manière de diviser les concepts assure une propriété intéressante qui permet l'optimisation des efforts de construction : lorsqu'un concept est traité, les concepts qui sont atteignables depuis celui-ci sont soit déjà traités, soit ils se trouvent dans la CFC en cours de traitement.

La section suivante décrit brièvement la stratégie de construction conjointe au sein d'une base de connaissances.

3.4.4 Description algorithmique

Il s'agit de parcourir le graphe de dépendances entre concepts $\mathcal{K}$ à partir du concept dont l'utilisateur a demandé l'organisation en classes, soit k. Le but du parcours est de détecter les ensembles de concepts co-définis ne pouvant pas être dissociés pour la construction de classes. Les ensembles recherchés forment des composantes fortement connexes du graphe K,

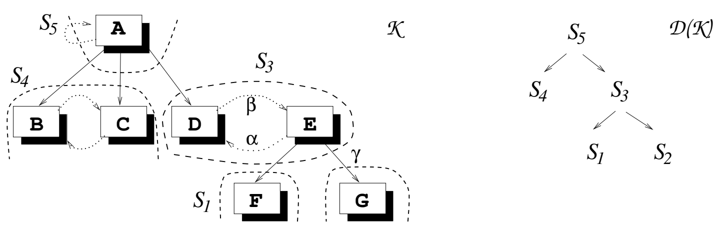


FIG. 3.6: *Exemple d'un graphe des dépendances entre concepts et de son DAG des CFC. À gauche, le graphe $\mathcal{K}$ est dessiné : les concepts sont nommés par une lettre en majuscule ; les attributs relationnels apparaissent comme des arcs ; les CFC du graphe sont délimités par des pointillés. À droite, le DAG $\mathcal{D}(\mathcal{K})$ est visualisé.*

$\mathcal{S} = \{k_1, k_2, ..., k_n\}$. Par exemple, le graphe $\mathcal{K}$ de la figure 3.6 a cinq CFC, ($\mathcal{S}_1$, $\mathcal{S}_2$,...,$\mathcal{S}_5$). À noter que $\mathcal{S}_1$, $\mathcal{S}_2$ et $\mathcal{S}_5$ sont constitués d'un concept unique, alors que $\mathcal{S}_3$ et $\mathcal{S}_4$ comportent un couple de concepts co-définis chacun. Un premier but de la stratégie est donc de définir les CFC de $\mathcal{K}$. Ensuite, l'ensemble des CFC est organisé en un graphe sans circuits, un DAG, selon l'ordre introduit par les attributs liant des concepts des différentes CFC. Un arc lie $\mathcal{S} = \{k_1, k_2, ..., k_n\}$ à $\mathcal{S}' = \{k'_1, k'_2, ..., k'_m\}$ dans $\mathcal{D}$ si et seulement si il existe un attribut relationnel a à domaine $dom(a) = k_i$ et $cod(a) = k'_j$ où le concept k_i appartient à $\mathcal{S}$ et k'_j à $\mathcal{S}'$.

Ce nouveau graphe est parcouru en profondeur d'abord avec un traitement post-fixe. Sur la figure 3.6, l'ordre post-fixe est $\mathcal{S}_4$,$\mathcal{S}_1$,$\mathcal{S}_2$,$\mathcal{S}_3$,$\mathcal{S}_5$. À chaque étape, le traitement du nœud courant de $\mathcal{D}$, c'est-à-dire la CFC consiste à appliquer la construction de classes (appel de CONSTRUCTIONMULTIPLE() dans l'algorithme 4) sur l'ensemble des concepts constituants.

CONSTRUCTIONCLASSES(k)

parcours du graphe des dépendances $\mathcal{K}$ à partir de k
décomposition du graphe en CFC, $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_p\}$
formation du DAG $\mathcal{D}$ des CFC de $\mathcal{K}$
pour tout $\mathcal{S}_i$ dans l'ordre *post-fixe* de $\mathcal{D}$ **faire**
  CONSTRUCTIONMULTIPLE($\mathcal{S}_i$) {construction de classes sur les concepts k_j dans $\mathcal{S}_i$}

Algorithme 4: Stratégie de construction de classes sur un concept k

3.4.5 Mise en œuvre

Deux algorithmes de recherche des CFC d'un graphe orienté peuvent être trouvés dans la littérature. L'algorithme classique est celui de Tarjan [Tarjan, 1988]. Il est de complexité linéaire mais a l'inconvénient de ne pas construire le DAG des CFC. Un algorithme qui construit le DAG tout en demeurant de complexité linéaire peut être trouvé dans [Aho *et al.*, 1972]. Nous avons repris cet algorithme et l'avons adapté à la structure des entités dans le modèle à objets utilisé.

3.5 Méthode de construction de classes

La méthode est le procédé algorithmique de construction effective de classes sur un ensemble d'objets. Nous décrivons dans cette section une méthode de construction compatible avec la stratégie établie préalablement.

3.5.1 Préalable

Dans le chapitre 2 nous avons montré qu'une méthode concrète de construction de classes se caractérise par un certain nombre d'éléments qu'elle met en contribution. Tout d'abord, elle fournit des langages de description pour les données et pour les résultats. Un sous-ensemble des méthodes présentées offre également des langages de description des connaissances préalables sur le domaine dont les données sont issues. Ensuite, la méthode offre des outils effectifs pour l'accomplissement de deux tâches, la constitution de groupes et la caractérisation des groupes en termes du langage de description des classes. Le processus global de construction de classes inclut également une étape d'évaluation de l'ensemble des classes. Celle-ci peut être automatisée, c'est notamment le cas lorsque le but de la construction des classes est la prédiction de valeurs d'attributs non-observées. Dans le cas général, l'évaluation des classes est faite par l'utilisateur qui est le seul à posséder l'objectif de leur construction.

Une différence majeure entre notre méthode et les méthodes existantes est dans la séparation des individus à regrouper en familles disjointes dont les classes sont organisées en taxonomies séparément. Toutefois, certaines familles sont traitées de manière simultanée, à cause des liens de co-définition entre familles. Puis, contrairement à la majorité des approches en regroupement conceptuel, notre méthode sépare les étapes de regroupement et de caractérisation.

3.5.2 Description des connaissances

Dans notre propre cas, les trois langages de description sont fixés : les données sont les objets, les résultats, c'est-à-dire les classes sont décrites à l'aide (d'un sous-ensemble) du langage de description des classes fourni par le SRCO. Quant aux connaissances du domaine, celles-ci peuvent prendre des formes variées en fonction des possibilités offertes par le SRCO. Toutefois, nous nous limitons à ce qui est communément disponible dans une base de connaissances à objets, l'ensemble des classes déjà identifiées. À noter l'identité entre les langages de description des résultats et des connaissances du domaine qui facilite, en particulier, la réutilisation des résultats d'une étape de construction lors d'une étape ultérieure. En effet, les classes identifiées à l'aide du module de regroupement peuvent par la suite servir dans la constitution et la description d'autres classes.

3.5.3 Processus de regroupement

Lors de cette étape, l'ensemble d'objets est séparé en sous-ensembles. Selon la technique utilisée, ces sous-ensembles peuvent à leur tour être séparés en des sous-ensembles plus fins et cela de manière récursive. Cependant, si l'on se concentre à un seul niveau de séparation, le but du processus est toujours de détecter des sous-ensembles vérifiant les exigences suivantes :

– les membres d'un sous-ensemble sont les plus similaires,

– deux sous-ensembles sont les moins similaires.

Chaque technique de regroupement met en œuvre un critère de qualité pour le découpage en sous-ensembles qui traduit sous une forme mesurable l'intuition ci-dessus. Il est le plus souvent défini explicitement, mais peut aussi être «encodé» de manière algorithmique, c'est-à-dire implicite, dans le corps de la technique. Dans notre cas, il s'agit d'un critère basé sur la minimisation des différences entre objets au sein d'une classe et sur la duale maximisation des différences entre objets appartenant à des classes différentes. Un second critère guide le processus de construction dans la direction de l'optimisation (le plus souvent locale) du critère de qualité. Il est constitué par une fonction de dissimilarité.

Un aspect orthogonal du processus de regroupement concerne le recouvrement entre sous-ensembles découverts. Dans notre cas, le choix de sous-ensembles disjoints est imposé par le système d'accueil, Troeps. Cependant, la disjonction étant un paramètre des techniques de regroupement (voir [Martin et Billman, 1994; Bisson, 1993]), notre méthode est suffisamment générique pour être facilement adaptée à la production d'un DAG de classes plutôt qu'un arbre.

3.5.4 Processus de caractérisation

Il s'agit d'une inférence de la description d'une classe à partir des descriptions des objets dans son extension. Nous supposons que les groupes d'objets ont été constitués et qu'une classe a été créée pour chaque groupe. Le but du processus est donc de définir :

– l'ensemble des attributs de la classe,
– une contrainte pour chaque attribut.

L'ensemble des attributs est inclus dans l'ensemble de tous les attributs communs aux objets de la classe. Quant à la contrainte sur un attribut particulier, celle-ci est produite à partir de l'ensemble des valeurs de cet attribut dans les objets.

Soit $O = \{o_1, o_2, ..., o_n\}$ l'ensemble des objets et c la classe créée. Soit encore A l'ensemble des attributs partagés par les objets dans O. Il est nécessaire de définir un opérateur de caractérisation $GEN_o()$ dont le seul argument est un ensemble d'objets et le résultat est une liste attribut-contrainte. Logiquement, $GEN_o()$ fera appel à des opérateurs de généralisation sur chaque attribut a, $gen_a()$. Ici $gen_a()$ est un opérateur générique qui s'instancie en fonction du type et de la nature de l'attribut a.

3.5.5 Description algorithmique

La méthode de construction de classes que nous proposons sépare les étapes de regroupement et de caractérisation afin de réaliser un traitement simultané de tous les concepts co-définis au sein d'une CFC. Les classes sont d'abord formées en extension par un algorithme de regroupement par dissimilarité ; ensuite, chaque classe est décrite en intention. Le processus global de construction se divise donc en trois étapes (voir l'algorithme 5 ci-dessus) :

- calcul de la fonction de proximité,
- formation de sous-ensembles (groupes) par des algorithmes de regroupement,
- transformation des groupes en classes puis description des classes.

ConstructionMultiple($\mathcal{S}$ ={$k_1, k_2, ..., k_m$})

CalculerDissim($\mathcal{S}$) {calcul simultané des dissimilarités sur les couples d'objets de chaque k_i}
pour tout concept k_i dans $\mathcal{S}$ **faire**
    Regrouper(k_i)
pour tout concept k_i dans $\mathcal{S}$ **faire**
    CaractériserClasses(k_i)

Algorithme 5: Construction de classes sur une CFC $\mathcal{C}$

L'avantage de cette démarche est que les classes sont constituées indépendamment de leur description. Par conséquent, lors de la caractérisation, toutes les classes sur les concepts co-définis existent déjà et peuvent être utilisées dans la description d'autres classes. Le processus intégral est visualisé sur la figure 3.7 qui représente deux familles d'objets à organiser en classes. La construction commence par une première étape de calcul de dissimilarité effectuée de manière simultanée sur la totalité des concepts. L'étape produit un ensemble de matrices de dissimilarité, une par concept, utilisées par la suite par des algorithmes de regroupement automatique dans la formation de groupes homogènes d'objets sur chaque concept (traitement séquentiel des concepts). Chaque groupe donne ensuite lieu à une classe dont il devient l'extension, alors que la description de la classe est inférée à partir des descriptions des objets membres. Il est à noter que les liens circulaires entre objets ne gênent plus la construction car à ce moment toutes les classes nécessaires pour la caractérisation sont déjà identifiées.

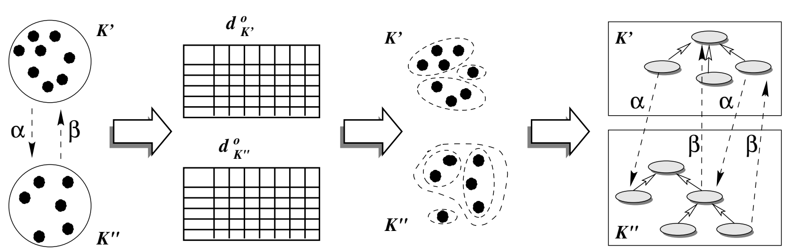


FIG. 3.7: *Méthode de construction de classes en mode multi-concepts.*

3.6 Outils pour la construction de classes

Il faut noter que les SRC sont bien fournis pour la conduite d'un raisonnement déductif : tous les mécanismes d'inférence corrects n'induisent que des connaissances qui sont déjà présentes dans la base sous une forme implicite. Les SRCO ne font pas défaut à cette règle car l'héritage a une nature essentiellement déductive. La construction de classes, quant à elle, part des objets et produit des classes qui sont des abstractions de ces objets, c'est-à-dire qu'elle est une opération inductive. Pour mener à bien un raisonnement inductif, un ensemble d'outils sont indispensable qui rendent possible le passage des connaissances individuelles vers les connaissances génériques.

3.6.1 Mesure de proximité

Afin d'assurer une meilleure cohérence entre le processus de regroupement et le but final de la construction de classes, nous utilisons une mesure de dissimilarité entre objets qui repose sur l'opération de généralisation. Celle-ci permet de mesurer, pour deux objets, la spécificité de leur classe commune la plus spécifique et cela en respectant les principes de comparaison énoncés plus haut dans ce chapitre. L'intuition est qu'en minimisant les dissimilarités au sein d'un groupe, les membres du groupe admettant des descriptions compactes deux à deux, une description compacte du groupe entier sera atteinte aussi.

Notre objectif premier dans les chapitres suivants (chapitres 5, 6 et 7) de ce mémoire sera donc de définir une mesure de proximité effective qui est en accord avec les principes de comparaison, ce qui signifie :

- la formulation d'un ensemble de fonctions de comparaison sur les valeurs d'attributs, encore appelées *mesures d'écart*,
- d'une fonction combinant les valeurs sur les attributs en une valeur, globale et unique, exprimant la proximité d'un couple d'objets selon leurs descriptions, la *mesure de dissimilarité*.

3.6.2 Algorithmes de regroupement

Un grand nombre d'algorithmes de regroupement peuvent être trouvés dans la littérature. La majorité d'entre eux ont été appliqués avec succès à un large éventail de tâches de regroupement allant de la segmentation d'un ensemble de clients dans une base de données commerciale à l'étude des modes d'écroulement du réseau d'approvisionnement en électricité.

Au sein de notre approche, tout algorithme utilisant une notion générique de proximité est potentiellement applicable. Seules exceptions sont les algorithmes qui reposent sur une interprétation géométrique de l'espace de description des individus. En effet, les notions comme le *centre de gravité* ou l'*inertie* d'un ensemble d'individus ne sont pas bien définies dans les formalismes admettant des attributs non-numériques.

Ainsi, parmi les algorithmes classiques, le *k-means* (voir section 2.3 du chapitre 2), ne peut être utilisé qu'au prix de certaines modifications. En revanche, les *nuées dynamiques* ou les algorithmes ascendants s'appliquent sans restrictions aux matrices de dissimilarité que notre mesure de dissimilarité produit.

3.6.3 Mécanismes de caractérisation

Si dans les SRC et plus particulièrement dans les SRCO il est aisé de passer de la description des classes à la description des objets membres, il n'y a pas, au moins à notre connaissance, de mécanismes dédiés à l'opération inverse, la synthèse de la description de la classe à partir des descriptions des objets. Inversement, en apprentissage automatique, la notion de généralisation est fondamentale : elle est l'opération centrale en apprentissage de concepts et joue un rôle prépondérant en regroupement conceptuel au sein des systèmes comme Cluster/2 et Kbg.

Compte tenu de l'importance de la caractérisation, il est indispensable de munir un système d'extraction de connaissances intégré à un SRC de mécanismes effectifs de caractérisation. Cette tâche s'avère relativement aisée dans les formalismes à objets, il suffit de « renverser » le test d'appartenance d'un objet à une classe, utilisé lors de la *classification* d'instances. Comme cette opération se réduit à un ensemble de tests d'appartenance d'une entité à un type, elle se base essentiellement sur les relations de *sous-typage* (sur les types de données) et de *spécialisation* (sur les concepts). Dans un SRCO à attributs typés, la *spécialisation* peut se réduire, sauf cas de circularité, à une suite de tests de *sous-typage*. La relation de *sous-typage*, quant à elle est directement définie sur les types de données et calculable.

La seule exception de cette définition est constituée par les attributs multivalués, pour lesquels il n'existe pas de langage de généralisations standard. Nous nous sommes vus obliger d'en définir un qui peut s'adapter à Troeps (voir chapitre 4).

3.7 Conclusion

Dans ce chapitre nous avons décrit le cadre que notre approche constitue. Nous avons également identifié les éléments constituant cette approche. Certains d'entre eux, qui se situent à un niveau générique, ont été également décrits. Ainsi, nous avons mis en évidence le besoin d'une stratégie de construction de classes lorsque plusieurs groupes d'objets inter-reliés sont concernés par cette tâche. La stratégie permet de séparer l'ensemble total en sous-ensembles indissociables et de ne s'occuper que d'un seul ensemble à la fois. Elle garantit également que lorsqu'un tel ensemble est traité, tous les autres ensembles dont il peut utiliser les structures de classes ont déjà été traités. Le traitement proprement dit, c'est-à-dire la construction effective des classes, est à la charge d'une méthode de construction. Celle-ci permet de construire des classes sur plusieurs familles d'objets grâce à une séparation entre les diverses tâches que l'opération comprend. Ainsi, les objets sont tout d'abord séparés en classes homogènes par l'usage d'une mesure de proximité, puis d'un algorithme standard de regroupement automatique. Ensuite, les classes constituées sont décrites de manière intentionnelle par l'application des opérateurs de généralisation.

Les outils qui assurent la réalisation de la méthode sont décrits dans les chapitres suivants : chapitre 4 pour les opérateurs de généralisation, chapitre 5 pour la définition d'un principe de comparaison entre entités individuelles, objets ou valeurs, et des fonctions de proximité sur les constituants de base, les types de données, et sur les familles d'objets ayant une taxonomie de classes. Une attention particulière est attribuée aux constructeurs de collections et la manière d'en extraire une valeur de proximité en accord avec la définition globale (chapitre 6). Finalement, le chapitre 7 décrit le traitement des circularités lors du calcul de la proximité.

4

Inférence de description de classes

Introduction

La méthode de construction de classes qui a été présentée au chapitre précédent impose une séparation entre les étapes de constitution des extensions des classes et la synthèse de description pour celles-ci. Ainsi, nous construisons d'abord des groupes d'objets homogènes, puis ces groupes sont transformés en classes et chaque classe est munie d'une description. Le but du chapitre est de donner un apperçu des principales opérations impliquées dans le calcul de description des classes. Pour la plupart, il s'agit des opérateurs de généralisation, une opération très utilisée en apprentissage de concepts. La définition d'opérateurs de généralisation sur les objets suit le principe général qui guide notre approche vers la construction des classes : deux types d'opérateurs sont spécifiés, un pour les classes et un pour les attributs. Les opérateurs que nous proposons ont une définition intuitive, nous avons choisi néanmoins de les décrire de manière formelle car cela facilite leur utilisation ultérieure dans le document. De plus, les deux opérateurs représentent une bonne illustration de l'approche globale : définition d'un traitement sur les constituants élémentaires puis sur les constructeurs de descriptions. En même temps, ils offrent un exemple de la double vision sur les objets qui est à la base de l'approche : d'une part comme des tuples de valeurs, d'autre part comme des éléments des ensembles que les classes représentent.

Au début, nous faisons quelques brefs rappels sur la nature de l'opération de généralisation, ainsi que sur les opérateurs définis dans divers types de formalismes (section 4.1). Ensuite, une description des principes de notre approche est proposée (section 4.2). Celle-ci est réalisée moyennant deux opérateurs de généralisation. Le premier est une opérateur générique sur les attributs des objets, $gen_a()$ (section 4.2). Il s'instancie différemment selon le constructeur de l'attribut : les opérateurs pour les types monovalués (section 4.3) et pour les types multivalués (section 4.4) sont présentés. Finalement, la section 4.5 est dédiée à la définition de l'opérateur $GEN_o()$ qui combine le résultat des divers opérateurs $gen_a()$ sur un ensemble d'objets en une description de classe pour cet ensemble.

4.1 L'opération de généralisation

La production d'une description de classe, générique, à partir d'un ensemble de descriptions d'objets, individuelles, dont elle est un résumé, est une forme de généralisation. La généralisation est couramment utilisée par les méthodes d'apprentissage symbolique, surtout sa branche supervisée. L'apprentissage de concepts est une forme d'apprentissage dont la généralisation d'un ensemble d'exemples en un (une description de) concept est l'opération centrale. Celle-ci a connu un succès considérable ces dernières années, ce qui explique l'abondance de la littérature sur les sujets concernant la généralisation (voir [Mitchell, 1997], chapitre 2).

4.1.1 Nature de l'opération

Intuitivement, la généralisation dans notre contexte peut être définie comme l'extraction de traits communs à partir d'un ensemble d'observations en vue de la caractérisation de l'ensemble complet. En toute généralité, la généralisation est une opération inductive qui permet de passer du plus spécifique au plus général. À noter que cette opération ne concerne qu'un seul aspect des classes dans les modèles à objets, leur description. Mis à part la description, les classes possèdent une identité et une extension (partielle). De plus, d'autres connaissances génériques s'y associent. Avant de présenter un jeu d'opérateurs de généralisation pour la production de descriptions de classes, nous allons brièvement rappeler ce qu'est la généralisation en apprentissage automatique. Pour cela, nous introduisons le cadre formel qui semble faire l'unanimité dans le domaine.

4.1.2 Définitions

Les observations sont décrites à l'aide d'un langage de description. La généralisation est une opération qui permet de passer des expressions représentant les observations vers l'expression d'une abstraction de ces observations [Michalski, 1983]. Le cadre général d'un problème de généralisation est décrit par Mitchell [Mitchell, 1990a] comme suit :

Étant donné :

- un ensemble d'individus, $\mathcal{I}$, décrits à l'aide d'un langage de description des individus, $L_{\mathcal{I}}$,
- un ensemble de généralisations potentielles, encore appelées hypothèses, Υ, les généralisations sont décrites en termes d'un langage de description des généralisations, L_{Υ},
- un prédicat, `covers()`, permettant d'établir l'appariement entre une description d'individu et une généralisation,

Rechercher : Une généralisation qui s'apparie avec tous les individus.

D'autre part, comme Mitchell le souligne, un grand nombre de généralisations possibles existe pour un ensemble $\mathcal{I}$ donné [Mitchell, 1990b]. Une relation de généralité est définie sur les couples de généralisations : une généralisation g est plus générale qu'une autre, g' si et seulement si tout individu qui s'apparie avec g' s'apparie aussi avec g. En d'autres termes, l'ordre de généralité est équivalent à l'inclusion entre ensembles d'individus *couverts* par les généralisations. Il est donc possible de parler de généralisations plus *générales* ou plus *spécifiques*.

La relation de généralité n'est pas forcément une relation d'ordre, mais peut s'y ramener en prenant comme équivalentes toutes les généralisations couvrant le même sous-ensemble d'individus. Nous allons considérer désormais, pour simplifier, que cette relation est effectivement un ordre. Ainsi, les membres de Υ sont organisés par la relation en une structure hiérarchique qui est parfois appelée *espace des généralisations*.

La tâche de la généralisation peut donc être vue comme une recherche dans cet espace d'une généralisation qui optimise un critère [Mitchell, 1990a]. En effet, une conséquence directe de la multiplicité des descriptions généralisantes est le besoin d'en choisir une. Ce choix est généralement guidé par l'objectif de l'opération : caractériser une seule ou plusieurs classes ou discriminer une ou plusieurs de ces classes par rapport au reste. L'opération est effectuée la plupart du temps en présence d'exemples et de contre-exemples pour chaque classe (voir par exemple [Michalski, 1983]). Ainsi, en apprentissage automatique supervisé on parle de généralisations caractéristiques et de généralisations discriminantes.

En apprentissage non-supervisé, on recherche le plus souvent la *généralisation la plus spécifique*. La raison en est qu'elle fournit les informations les plus précises sur l'ensemble d'individus à caractériser. Un critère supplémentaire qui peut guider le choix en regroupement conceptuel est la *disjonction* qui exige que les classes retrouvées sur un ensemble d'individus soient disjointes sur leur extension (ensembles d'individus de $\mathcal{I}$ couverts), mais aussi sur leurs descriptions. Cela revient à une disjonction sur les ensembles d'individus couverts que l'on peut décrire à l'aide de $L_{\mathcal{I}}$. Ce type de disjonction est utilisé par le système Cluster/2. Toutefois, elle s'est révélée une condition trop coûteuse à satisfaire.

Quant à la recherche effective d'une généralisation convenable, un ensemble d'opérateurs de généralisation permet de naviguer dans l'espace des généralisations. Des opérateurs dits de généralisation, de spécialisation ou de reformulation ont été définis dans [Michalski, 1983].

Dans la suite, nous nous efforcerons de faire la distinction entre les acceptions différentes du mot « généralisation » dans la littérature. Ayant fixé les données du problème, il est courant de confondre deux ou trois des notions énumérées ci-dessous qui relèvent de la généralisation :

- une **opération**, permettant le passage d'une description plus spécifique, c'est-à-dire se rapportant à un nombre plus petit d'individus, vers une description plus générale ; notre cas est un cas particulier où l'on manipule plusieurs descriptions spécifiques des individus,
- une **relation** existant entre les descriptions ; celle-ci distingue une description plus spécifique et une plus générale et en cela elle ne va pas sans rappeler la *spécialisation* entre classes dans les SRCO ; pour cerner les différences fondamentales, rappelons que la spécialisation est une relation affirmée ; cela veut dire que deux classes qui ne sont pas liées par la spécialisation peuvent avoir des descriptions, qui sont compatibles pour la relation de généralité ou la généralisation ; cependant, dans notre contexte, c'est-à-dire la construction automatique de classes, les deux relations sont équivalentes ; en particulier, si deux classes automatiquement construites sont liées par une généralisation entre descriptions, alors elles le sont aussi pour la spécialisation dans la taxonomie finale, et cela dans le même sens,
- une **description** qui est le résultat de l'opération de généralisation sur un ensemble d'autres descriptions est dite leur généralisation.

Dans ce qui suit, le terme généralisation sera employé pour désigner, sauf mention explicite, l'opération de généralisation.

4.1.3 Généralisation en formalismes structurels

Dans les formalismes structurels les individus ont des descriptions qui rassemblent la totalité des connaissances qui concernent chaque individu. Cela permet de fixer un ensemble de propriétés qui caractérisent conjointement tous les individus et un ensemble de relations entre individus. Chaque caractéristique est représentée au sein de la description d'un individu particulier par un attribut. Ainsi, une telle description peut être vue comme un tuple (type de données *enregistrement* en programmation) composé par les valeurs des différents attributs. Les langages de description de classes permettent aussi la formation de tuples dont les éléments sont des plages de valeurs admissibles et non des valeurs comme dans les objets. Le but de l'inférence de descriptions est donc de déterminer ces plages de variation pour chaque attribut partagé par les objets à généraliser.

Les langages de type attribut-valeur comme ceux utilisés par la plupart des systèmes de regroupement conceptuel classiques (COBWEB, AUTOCLASS, WITT, etc.) sont des formalismes structurels simples, c'est-à-dire sans relations. Ils ont une expressivité essentiellement égale à celle des langages de la logique des propositions (VL_{21} par exemple [Michalski et Stepp, 1983]).

La généralisation est relativement aisée sur un tel formalisme. Les descriptions des individus peuvent être assimilées à des conjonctions d'atomes propositionnels sous la forme (attribut - relation - valeur) appelée sélecteur dans [Michalski, 1983]. Les hypothèses sont décrites dans un langage légèrement étendu admettant par exemple des disjonctions internes (voir [Michalski, 1983]). La généralisation consiste, dans sa version la plus simple, en une intersection des ensembles d'atomes. Une forme plus sophistiquée permet de rendre les descriptions plus compactes en autorisant des disjonctions entre valeurs dans les atomes. Celles-ci expriment la variabilité de la propriété sous-jacente dans les individus caractérisés.

4.1.4 Généralisation en formalismes relationnels

Dans les langages de représentation à caractère relationnel (langages de prédicats de premier ordre, graphes conceptuels), les descriptions des individus sont composées d'un ensemble de noms d'individus simples liés par des relations. La généralisation cherche à isoler la structure commune entre descriptions qui se présente sous forme de sous-ensembles d'individus simples liés par les mêmes relations. La détection d'une telle sous-structure est un problème NP-difficile car il relève d'une recherche de sous-graphe isomorphe. Afin de garder la complexité polynomiale, diverses techniques de nature heuristique sont employées, dont une des plus fréquentes consiste à restreindre le langage de description des classes, L_{Υ}.

Logique des prédicats

Avec un langage de prédicats ou de puissance équivalente les individus sont décrits par des conjonctions d'atomes instanciés. Ils sont dénotés par des constantes qui interviennent en argument des divers prédicats. Les hypothèses sont, en revanche, des formules (sous forme clausale, le plus souvent) où des variables peuvent apparaître.

Le principe de la généralisation a été décrit de manière théorique par Plotkin qui a introduit la notion de généralisant le plus spécifique de deux clauses. Les résultats donnés par Haussler [Haussler, 1989] soulignent le caractère NP-difficile de la généralisation avec un langage de clauses non contraint. En effet, lors de la généralisation, un appariement doit être trouvé entre variables/constantes apparaissant dans les formules à généraliser. La recherche de tel appariement avoisine le problème d'isomorphisme de graphes. Ce problème est NP-complet, la généralisation est donc NP-difficile. Depuis, un travail pour déterminer les classes de langages (d'hypothèses) qui admettent une opération de généralisation efficace a été effectué au sein de la communauté *Programmation Logique Inductive* (voir [Lavrac et Dzeroski, 1994] pour une introduction en ILP). Parmi ceux-ci, citons :

- les langages sans symboles de fonctions [Quinlan, 1990], comme celui utilisé par le système FOIL [Quinlan et Cameron-Jones, 1993],
- les langages de clauses ij-déterminées utilisées par GOLEM [Muggleton et Feng, 1992].

Avec un langage de clauses, les techniques de généralisation les plus utilisées sont le calcul du généralisant le plus spécifique relatif[20] (par un ensemble de connaissances du domaine) et la

20. *relative least general generalisation* en anglais

résolution inverse [Muggleton et Buntine, 1988]. Les deux techniques construisent les hypothèses en introduisant des variables à la place des constantes et en excluant des termes prédicatifs.

Une forme intéressante de la généralisation dans la LPO, dite *structurelle*, est utilisée par le système KBG [Bisson, 1992]. Elle repose sur un appariement établi à l'aide de la mesure de similarité discutée au chapitre 2.

Graphes conceptuels

La généralisation au sein d'un langage de graphes conceptuels, considère deux langages qui sont quasiment identiques (à la présence d'un marqueur individuel près). La généralisation d'un ensemble de graphes représentant des individus s'apparente à la recherche du sous-graphe commun maximal. Compte tenu de la complexité prohibitive du calcul du sous-graphe maximal (en réalité, c'est un problème NP-difficile [Garey et Johnson, 1979]), diverses heuristiques sont utilisées. Une technique possible est la décomposition des graphes en leur ensemble d'arcs conceptuels (voir [Mineau, 1990] par exemple), c'est-à-dire l'aplatissement de la structure de graphe. Les arcs sont constitués par un couple de sommets conceptuels et le sommet relationnel qui les lie (limitation à des relations binaires). Chaque arc peut être généralisé en augmentant le type dans un ou plusieurs de ses sommets.

On ne travaille que sur des ensembles d'arcs conceptuels sur lesquels la généralisation est retrouvée par une intersection.

4.1.5 Synthèse

Le passage de l'individuel vers le générique est une opération de généralisation. Elle est souvent utilisée dans le cadre d'une structuration des ensembles d'individus par apprentissage, supervisé ou non. Dans le cas non-supervisé, trois éléments du problème doivent être fixés : les langages de description pour les individus et pour les classes, ainsi que le type de généralisation que l'on veut obtenir. Parmi ceux-ci, le langage des généralisations jouit d'une plus grande liberté car les deux autres éléments sont fortement liés à la définition du problème d'apprentissage.

Alors que la généralisation sur les langages de description structurels établit des plages de variabilité pour un ensemble d'attributs, dans les langages relationnels, des squelettes relationnels communs entre les descriptions des individus sont recherchés. La dernière opération étant de nature NP-difficile dans le cas général, les systèmes utilisant une généralisation ont recours à des restrictions dans le langage des généralisations.

Cette simplification peut se justifier si elle ne détériore pas sensiblement la qualité de la solution du problème traité. Toutefois, la décision de son emploi ne peut être prise qu'au cas par cas. Une telle technique ne peut être utilisée sur les objets sans nuire à l'objectif de complétude et flexibilité. En effet, plutôt qu'un outil adapté à un sous-ensemble de situations qui permettraient des restrictions raisonnables, nous cherchons à concevoir la construction de classes de manière la plus générique possible en ne l'appuyant que sur des caractéristiques propres au formalisme d'expression que sont les objets.

4.2 La généralisation dans les objets

Le but de la généralisation dans un contexte d'objets est l'inférence d'une description de classe à partir des descriptions de ses instances. La tâche se réduit sur chacun des attributs : une restriction pour les valeurs de chaque attribut dans les instances de la classe sera calculée.

C'est une instanciation du schéma générique décrit par Mitchell dans [Mitchell, 1990a] : le langage des individus est le langage de description des objets et le langage des généralisations le langage de description des classes. Le **premier langage** est composé par des listes de couples attribut-valeur. L'ensemble des attributs est celui défini par la classe commune des objets (la racine du concept dans TROEPS). Les valeurs sont de deux sortes, valeurs simples et valeurs objets. Le **second langage** est composé de listes attribut-type. Les attributs sont les mêmes que dans le cas des objets. Chaque type est un sous-type du type initial de l'attribut défini dans la classe commune aux objets.

4.2.1 Préliminaire

Intuitivement, il s'agit de mettre à profit les relations de spécialisation et de sous-typage qui sont inhérentes aux modèles à objets. En effet, dans une base de connaissances à objets, ces deux opérateurs sont utilisés comme « briques de base » pour quasiment toute manipulation (classification, filtrage, etc.). Ainsi, le co-domaine d'un attribut a est considéré comme muni d'un ensemble Υ_τ de parties du domaine $\mathcal{I}_\tau = \texttt{dom}(\tau)$ [21] du type sous-jacent, $\tau = \texttt{type}(a)$. Υ_τ est donc un sous-ensemble de 2^τ qui décrit toutes les généralisations admissibles sur τ. Il peut être spécifié explicitement, en particulier en cas d'attribut relation de co-domaine k. Ainsi, Υ_k est l'ensemble des classes sur k, soit $\mathcal{C}_k$.

L'ensemble Υ_τ peut également rester implicite s'il s'agit d'un type de données T. Dans ce cas, l'ensemble de tous les types t sur T, soit $\mathcal{T}_T$ est facilement calculable de sorte que ses membres sont engendrés au fur et à mesure que leur utilisation se rend nécessaire.

La technique de généralisation consiste à prendre, à partir d'un ensemble d'entités dans $\mathcal{I}_\tau$, un élément précis g de Υ_τ.

Pour un type de données T, l'opération à réaliser est une synthèse de type t, alors que pour un concept, il s'agit d'un simple choix d'une classe c parmi les classes dans $\mathcal{C}_k$. Le choix exact de g est défini par le type de généralisation que l'on cherche à obtenir (voir section 4.2.5).

4.2.2 Exemple introductif

Dans la suite de ce chapitre, nous allons nous servir de l'exemple ci-dessous pour illustrer l'effet des opérateurs de généralisation que nous allons définir. En outre, cet exemple présente une situation où un ensemble de classes est à caractériser en termes d'autres classes du même ensemble. Il nous permettra donc d'illustrer la seconde étape de notre méthode de construction dans le cas de la co-définition entre classes.

TAB. 4.1: *Exemple d'ensemble d'objets dans* `humain`

attribut	Pierre (o_1)	Anne (o_2)	Yves (o_3)	Léa (o_4)	Claude (o_5)	Sophie (o_6)
`âge`	26	23	27	30	32	35
`salaire`	3	2,3	2,2	2,7	2,9	3,2
`sexe`	M	F	M	F	M	F
`conjoint`	Anne	Pierre	Léa	Yves	Sophie	Claude

21. Domaine de type et domaine d'attribut n'ont pas le même sens, bien que dans les deux cas il s'agit d'un ensemble de valeurs.

L'exemple est composé de six objets, tous instances du concept `humain`. Les objets sont décrits par quatre attributs : `âge`, `salaire`, `sexe` et `conjoint` (voir table 4.1). L'attribut `salaire` indique le revenu mensuel d'une personne (en Euros), son type est réel, l'attribut `âge` est de type entier et l'attribut `sexe` de type énuméré (chaîne de caractères). L'attribut `conjoint` est de nature lien et de type `humain`. Les six objets constituent trois couples d'époux : (o_1, o_2), (o_3, o_4) et (o_5, o_6).

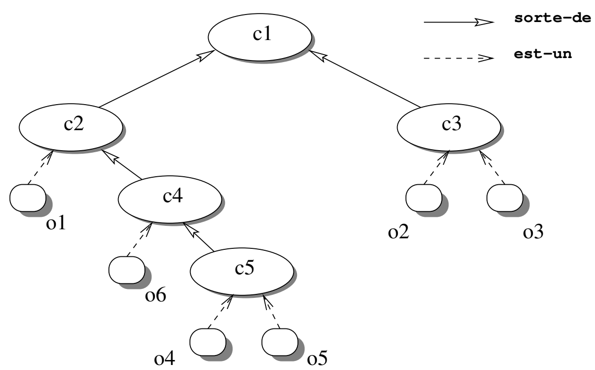


FIG. 4.1: *Exemple d'une taxonomie de classes sur l'ensemble des objets de la table 4.1. Les classes n'ont pas de description, celle-ci est à construire à partir des descriptions des objets.*

Les objets sont organisés en cinq classes qui forment une taxonomie. La taxonomie est visualisée sur la figure 4.1. Les classes n'ont pas de description, elles sont donc à caractériser.

4.2.3 Principes de la généralisation

La technique est relativement simple. Nous lui réservons néanmoins une place importante dans ce mémoire car elle offre une bonne illustration de notre approche globale qui impose le même schéma pour la définition de chacun des outils. Le schéma permet de respecter la structure des objets en répartissant la définition des outils sur les différents niveaux de la structure.

Traitement à deux niveaux

En suivant la structure des objets, nous définissons les opérateurs de généralisation :

- un traitement sur les constituants élémentaires de la structure des objets, c'est-à-dire sur les attributs monovalués,
- un traitement sur chacun des constructeurs de description, *tuple* pour les objets et *liste* ou *ensemble*, pour les collections.

Les traitements du premier groupe sont directement exécutables sur un ensemble d'entités de représentation, objets ou valeurs simples. En revanche, les traitements sur les descriptions construites se décomposent en un ensemble de traitements sur leurs constituants dont les résultats sont ensuite combinés en un résultat global.

À noter que les deux types d'opérateurs peuvent s'appliquer, pour des raisons différentes, au même ensemble d'objets O. Si la tâche de la caractérisation porte sur la classe commune c des objets dans O, l'opérateur de tuple est appliqué sur O. Si, en revanche, la classe c doit servir dans la caractérisation d'une classe c' sur un ensemble d'objets O' dont les valeurs d'un attribut

a sont dans O, alors le premier type d'opérateur est à appliquer sur O. Par exemple, si l'on cherche à caractériser la classe `c3` sur la figure 4.1, le second type d'opérateur sera appliqué sur l'ensemble des objets $\{o_2, o_3\}$, alors que si l'on cherche à décrire une classe commune à $\{o_1, o_4\}$, le premier type d'opérateur sera appliqué à $\{o_2, o_3\}$.

Cette manière de procéder correspond à une double vision sur les objets : d'une part comme des structures d'attributs, d'autre part comme membres de classes ; cette vision bi-valente est intrinsèquement présente dans les mécanismes de manipulation d'objets ; l'exemple du filtrage : un objet est typiquement considéré comme structure lorsqu'il est « filtré », en revanche, si le filtre spécifie des restrictions sur un de ses attributs relationnels, alors le ou les objets qui font partie de sa valeur dans l'objet filtré sont considérés comme des membres de leurs classes, sauf si des conditions sur leur propres attributs sont spécifiées par le filtre.

Inverser l'attachement

L'aspect descriptif d'une classe Troeps consiste en la donnée d'un ensemble d'attributs et des contraintes sur les valeurs pour chaque attribut. Inférer une description de classe revient donc à donner la liste de ses attributs et leurs restrictions respectives. Pour cela, on peut utiliser, pour un attribut donné, l'ensemble de toutes les valeurs de celui-ci qu'on rencontre dans les objets membres de la classe.

Les objets sont liés à leur classe par une relation d'*attachement*. Celle-ci exprime une contrainte forte entre les valeurs d'un attribut dans les objets et le type de cet attribut, spécifié dans la classe. Ainsi, toute valeur dans un objet doit appartenir au type spécifié par la classe.

Inversement, si l'on considère l'ensemble de toutes les valeurs d'un attribut dans les objets de la classe, celui-ci décrit un sous-ensemble des valeurs admises par le type de l'attribut dans la classe. On peut donc utiliser cette observation afin de définir le principe de l'opération de généralisation sur les objets. En effet, il suffira de prendre, pour un attribut commun, un type/classe qui inclut la totalité des valeurs/objets. En combinant les descriptions pour l'ensemble des attributs en commun, il est possible de produire une description complète de la classe commune des objets.

4.2.4 Aspect dynamique et multi-concept

La technique exige une prise en compte de l'aspect dynamique de la construction de classes. Ainsi, au moment de la caractérisation, l'existence des ensembles Υ_τ sur chaque attribut a doit être assurée. À noter qu'une alternative existe pour les cas où Υ_τ n'est pas directement accessible pour un attribut a. Celle-ci consiste à prendre un ensemble Υ_τ composé d'un seul élément, celui qui contient la totalité des valeurs de a dans l'ensemble des objets à regrouper, soit O. Toutefois, cette alternative ne conduit pas à la construction de classes pertinentes sur a car leurs descriptions ne permettront pas de distinguer les objets sur cet attribut. Sur l'exemple de la section précédente, cela revient à prendre systématiquement, c'est-à-dire dans chacune des cinq classes sur la figure 4.1, pour l'attribut `conjoint` la restriction `conjoint = c1`. Une telle caractérisation n'a, en effet, aucun pouvoir discriminant sur les objets et du coup n'apporte rien à l'interprétation du regroupement effectué.

4.2.5 Choix effectués

Langage des généralisations

Le système Troeps offre un large éventail de moyens d'expression pour les restrictions de type. Par conséquent, plusieurs langages de généralisations peuvent être envisagés. Nous nous tiendrons à une solution unique, celle qui représente le meilleur compromis entre la lisibilité (donc concision) et précision (moindre « bruit » dans la description et donc perte d'information). Le langage choisi part d'un sous-ensemble du langage offert par Troeps, dans la mesure où nous ne considérons que des descriptions « convexes » :

- sur les types de données ordonnés, un seul intervalle est pris,
- il n'y a pas d'interdiction de valeurs avec `sauf`,
- pour les relations monovaluées, une seule classe est considérée pour les relations, ce qui correspond à la prise en compte d'un seul point de vue par concept.

Ce noyau de langage est complété par des descriptions sur les attributs multivalués qui représentent une extension du langage de Troeps. Ainsi, étant donné un attribut multivalué, les éléments du langage sont des couples d'intervalles. Le premier intervalle porte sur les cardinalités des collections, donc sur les nombres entiers. Le second est un intervalle d'ordre partiel défini à partir des généralisations sur les éléments du plus bas niveau des collections. Une opération d'extraction de ces éléments est donc utilisée dans les cas où les membres des collections à généraliser sont eux-mêmes des collections.

Type de généralisation

Pour la généralisation dans Troeps, nous avons choisi systématiquement les descriptions de type les plus spécifiques. Celles-ci apportent le plus d'information sur les objets. Bien évidemment, cela n'est valable que dans le cadre fixé par le choix du langage de description des classes.

Quant à la disjonction, nous l'exigeons sur les extensions des classes, mais pas sur leurs descriptions. La raison pour la première condition est que dans Troeps, ainsi que dans d'autres SRCO, la multi-instanciation, c'est-à-dire l'appartenance d'un objet à plusieurs classes incomparables au sein de la même taxonomie est interdite. Si l'on autorise des intersections non vides entre extensions de classes incomparables, l'absence de multi-instanciation devient difficile à assurer. En revanche, il n'est pas d'usage de chercher des classes de descriptions disjointes dans un SRCO qui est destiné, entre autre, à la représentation de connaissances partielles et évolutives. Ainsi, les descriptions de classes couvrent souvent une partie plus large de l'espace des descriptions des objets que leur extension présumée. Des intersections non vides se produisent naturellement dans ce contexte.

En somme, nous allons nous tenir à des classes d'extensions disjointes. À noter que de telles classes sont naturellement produites par une méthode probabiliste ou basée sur des proximités dont le résultat est une *partition* des individus en groupes. En revanche, la notion de disjonction n'est pas inhérente à l'approche par treillis de Galois, sa préservation demande un effort supplémentaire de reformulation sur les concepts construits [Bournaud, 1996; Polaillon, 1998]. Notons au passage que cette reformulation peut conduire à une perte des propriétés de fermeture de ces concepts qui est parmi les justifications de l'application de l'approche par treillis.

4.2.6 Opérateurs génériques

Soit un ensemble d'objets $\{o_1, o_2, .., o_n\}$ appartenant au concept k définissant m attributs $\{a_1, a_2, .., a_m\}$. Nous chercherons à décrire la classe c dont les instances connues seront $\{o_1, o_2, .., o_n\}$. Ainsi, pour chaque attribut a_i, nous donnerons l'expression de l'attribut $c.a_i$. Pour cela, nous considérons les valeurs de a_i dans les objets $\{o_1.a_i, o_2.a_i, .., o_n.a_i\}$.

On définit la généralisation à deux niveaux, les opérateurs sur les attributs et sur les objets.

Description sur les attributs

La généralisation sur un attribut est un cas particulier du schéma décrit par Mitchell. En effet, les individus à considérer sont les valeurs de l'attribut concerné par la généralisation. Celles-ci peuvent être des valeurs simples ou des objets selon que l'attribut représente une propriété ou une relation. En fonction, le langage des généralisations est composé soit par des types soit par des classes. La particularité ici est que les types ne sont pas immédiatement disponibles, mais en revanche ils peuvent être facilement composés. Quant aux classes, leur ensemble est connu auparavant et elles sont directement utilisables, c'est-à-dire qu'aucune opération particulière de synthèse n'est nécessaire. Les prédicats de test d'appariement entre individus et généralisations sont aussi immédiatement disponibles car ils correspondent à des opérateurs utilisés lors de la classification d'instances dans un SRCO :

- test d'appartenance d'une valeur à un type ($\in_T$),
- test de satisfaction des contraintes définies dans une classe par un objet ($\in_k$).

Ces tests ne sont pas utilisés de manière directe, car de par la définition des opérateurs de généralisation, ils sont vérifiés.

On peut donc définir l'opérateur générique $gen_a()$ (pour niveau *attribut*). Soit un attribut a de type τ, soient encore $\mathcal{I}_\tau$ l'ensemble des entités du τ (on parlera également d'individus) et Υ_τ l'ensemble des généralisations sur $\mathcal{I}_\tau$. Cet ensemble est organisé dans une taxonomie $\mathcal{H}_\tau$ par une relation de généralité $\leq_\tau$ qui est un ordre. Dans ce contexte, l'opérateur de généralisation sur τ est défini comme suit :

Définition 4.1.

$$gen_a(\{e_1, e_2, \ldots, e_n\}) = min_{\leq_\tau}\{g \in \Upsilon_\tau | \ \forall i = 1, n; e_i \in \mathtt{membres}(g)\}.$$

L'opérateur $gen_a()$ s'instancie en fonction de la nature et du constructeur de l'attribut a. Ainsi, nous définissons un opérateur sur les propriétés, soit $gen_v()$, et un sur les relations, $gen_r()$. De plus, une attention particulière est accordée aux attributs multivalués dont la généralisation est effectuée moyennant l'opérateur $gen_m()$.

Dans la prochaine section, nous définissons l'opérateur sur les attributs monovalués. Ensuite, nous nous servons de cette définition pour donner une définition dans le cas multivalué.

Descriptions sur les objets

La définition de l'opérateur de généralisation sur les objets en tant que structures d'attributs $GEN_o()$ est fondée sur la notion de produit. Celle-ci utilise une version fortement simplifiée de la formalisation des types de tuples que l'on peut trouver dans [Abadi et Cardelli, 1996]. En particulier, nous ne nous intéressons pas à la possibilité d'accéder aux attributs d'un objet ou d'une classe par leur nom car cela n'a que peu d'importance dans le processus de généralisation.

En conséquence, on peut voir les objets comme des éléments d'un ensemble obtenu par produit à partir des domaines de leurs attributs. La notion de produit que nous utilisons est une généralisation du produit entre structures ordonnées car elle autorise les circularités : un domaine peut être produit d'un ensemble d'autres domaines dont il peut faire, directement ou non, partie. Toutefois, dans le cas de descriptions non circulaires, ce produit correspond au produit Cartésien d'ensembles.

Soit un concept k avec ensemble d'attributs $A = \{a_1, .., a_n\}$. Pour chaque a, soit son type $\tau(a)$ et soit $\Upsilon_{\tau(a)}$ l'ensemble des généralisations possibles sur le co-domaine de a. L'ensemble des individus $\mathcal{I}_k$ contient tous les objets du concept k, c'est-à-dire que $\mathcal{I}_k = objets(k)$. Celui-ci n'est généralement qu'un sous-ensemble du produit des domaines des attributs.

$$\mathcal{I}_k \subseteq \times_{a \in A} \mathcal{I}_{\tau(a)}.$$

L'ensemble des généralisations potentielles sur k, Υ_k est le produit des ensembles respectifs sur les attributs :

$$\Upsilon_k = \times_{a \in A} \Upsilon_{\tau(a)}.$$

Il est muni d'un ordre de généralité $\leq_k$ qui est le produit des ordres sur les attributs, $\leq_{\tau(a)}$

$$\leq_k = \times_{a \in A} \leq_{\tau(a)} .$$

Sous ces conditions, nous définissons l'opérateur de généralisation GEN_o sur k comme le produit des opérateurs de généralisation sur l'ensemble des attributs $a \in A_k$:

Définition 4.2.

$$GEN_o(\{o_1, o_2, .., o_n\}) = \times_{a \in A} gen_a(\{o_1.a, o_2.a, .., o_n.a\}).$$

Il faut noter que le manque de bien-fondé dans nos définitions n'est pas gênant pour l'usage que nous en faisons ici. S'il s'agit de l'ensemble des éléments du domaine $\mathcal{I}_k$, celui-ci est directement spécifié. Quant à l'ensemble des généralisations, Υ_k, il est également connu au moment de la caractérisation car notre méthode de construction de classes assure l'existence des ensembles admissibles sur chaque attribut de k *avant* l'étape de la caractérisation pour k. Finalement, les ordres $\leq_{\tau(a)}$ sont également connus car il se basent sur l'inclusion des ensembles de membres des généralisations g sur un type $\tau(a)$. En conséquence, chaque classe c de k à caractériser est également une généralisation autorisée sur k. Dans le cas de co-définition, la caractérisation revient à chercher pour c l'élément minimal de $\times_{a \in A} \Upsilon_{\tau(a)}$ qui «couvre» tous les objets dans c.

À noter que la vision ci-dessus est très proche du modèle des *systèmes classificatoires* dans leur version décrite dans [Euzenat, 1994]. Toutefois, nous n'avons pas introduit ce formalisme ici par souci de concision.

4.3 Description sur les attributs monovalués

Cette section clarifie la manière dont l'opérateur de généralisation s'instancie sur chaque type d'attribut.

4.3.1 Description sur les propriétés

Le type τ est un type de données T et l'ensemble des individus $\mathcal{I}_v$ est égal à son domaine $\mathcal{D}_T = dom(T)$, $\mathcal{I}_T = \mathcal{D}_T$. L'ensemble $\mathcal{T}_T$ de généralisations contient tous les types t qu'il est possible de décrire sur le type de données T ; il est organisé en une taxonomie $\mathcal{H}_T = \langle \mathcal{T}, \leq_T^t \rangle$ par la relation de *sous-typage* $\leq_T^t$. Le sous-typage suit l'inclusion entre ensembles de valeurs des types dans $\mathcal{T}_T$. Cet ensemble est donné par la fonction `valeurs`: $\mathcal{T} \rightarrow 2^{\mathcal{D}_T}$.

L'opérateur $gen_v()$ a l'effet inverse à la fonction `valeurs`. Il est défini sur l'ensemble des parties de $\mathcal{D}_T$ qu'elle renvoie sur l'ensemble des types $\mathcal{T}$, $gen_v : 2^{\mathcal{D}_T} \rightarrow \mathcal{T}$. Pour un ensemble de valeurs dans T, soit $\{v_1, v_2, \ldots, v_n\}$, elle calcule le type le plus spécifique dans $\mathcal{T}$ qui comprend toutes les valeurs dans l'argument.

Définition 4.3.

$$gen_v(\{v_1, v_2, \ldots, v_n\}) = min_{\leq_T}\{t \in \mathcal{T} | \; \forall i = 1, n; v_i \in \texttt{valeurs}(t)\}.$$

.

À noter qu'en Troeps, la fonction `valeurs` est calculée à l'aide des primitives offertes par le système de gestion de types [Capponi, 1995].

On montre ci-dessous l'instanciation de $gen_v()$ pour les différentes catégories de type que le système de gestion de types de Troeps admet. Chaque catégorie correspond à une échelle ou un ensemble d'*échelles* de la théorie de la mesure qui sont utilisées en analyse de données et en apprentissage automatique (voir [Anderberg, 1973] pour une explication sur les échelles utilisées en analyse de données).

Types de données ordonnés

Ils se distinguent par l'existence d'un prédicat d'ordre entre valeurs. Cette catégorie de types regroupe les échelles *numériques* (intervalles et rapports) et *ordinales* en théorie de la mesure.

En Troeps, on y catégorise les nombres *réels* et les nombres *entiers*. Les types t sur un type de données ordonné T sont des ensembles d'intervalles avec des interdictions de valeurs par le descripteur `sauf`. Par exemple, l'attribut `âge` dans la classe `c3` du concept `humain` que l'on observe sur la figure 4.1, peut être défini dans Troeps comme suit :
`c3.âge = { intervalles = { [23 27] } sauf = { [24 26]}}`.

Pour nos propres objectifs, nous avons fixé le langage des généralisations Υ_T comme étant l'ensemble de tous les intervalles sur T, $\mathcal{J}_T$.

L'opérateur de généralisation est donc défini comme suit :

Définition 4.4.

$$gen_{ord}(\{v_1, v_2, \ldots, v_n\}) = [\min(v_1, v_2, .., v_n), \max(v_1, v_2, .., v_n)].$$

Exemple La table ci-dessous illustre quelques applications de gen_{ord} sur les objets de l'exemple de la section 4.2.2. Les attributs concernés sont `âge` et `salaire`.

Ensemble de valeurs S	$gen_{ord}(S)$
`{o4.âge, o5.âge}`	`[30 32]`
`{o2.âge, o3.âge}`	`[23 27]`
`{o1.âge, o5.âge, o5.âge}`	`[26 32]`
`{o6.salaire, o4.salaire}`	`[2,7 2,9]`

Types de données énumérés

Les types de données énumérés correspondent à l'échelle *nominale* en théorie de la mesure. Pour un tel type de données T, aucune comparaison entre ses valeurs n'est possible. Les types sur T sont donc précisés en énumération. En d'autres termes, un type t est décrit sous forme d'un ensemble de valeurs.

Les contraintes sur les types énumérés dans les classes de TROEPS sont définies à l'aide du descripteur `domaine`. Celui-ci introduit un ensemble de valeurs dont l'interprétation est intuitive : elles sont les seules autorisées dans les attributs des instances de la classe. Par exemple, l'attribut `sexe` de la classe `c4` de la figure 4.1 qui est de type chaîne de caractères, et donc énuméré, est décrit par :
`c4.sexe = { domaine ={ "F" "M"}}`.

Dans le cas de types de données énumérés on prend comme langage des généralisations Υ_T l'ensemble des parties de D_T, $2^{\mathcal{D}_T}$, ce qui est également le choix dans TROEPS.

L'opérateur de généralisation gen_{enum} est donc défini comme suit :

Définition 4.5.

$$gen_{enum}(\{v_1, v_2, \ldots, v_n\}) = \{v_1, v_2, .., v_n\}.$$

4.3.2 Description sur les relations

Lorsque l'attribut a représente une relation, son co-domaine est une famille d'objets, soit le concept k. Les restrictions sur les attributs relations sont décrites en TROEPS par un ensemble de classes sur le concept co-domaine. Chacune des classes provient d'un point de vue différent. Une telle contrainte s'interprète comme l'ensemble des objets qui sont dans l'intersection des extensions des classes dans la liste (ils sont membres de chacune des classes). Les contraintes sont introduites par le descripteur `type`. Par exemple, l'attribut `conjoint` est de type `humain`. Ainsi, dans la classe `c3` sur la figure 4.1, il peut être décrit par :
`c3.conjoint = { type ={ c2 }}`.

Dans la suite, pour simplifier, nous ne considérons qu'une seule classe dans la liste introduite par `type`.

Définitions

Dans ce cas, l'ensemble des individus sur lequel on définit la généralisation, $\mathcal{I}_k$, contient tous les objets de k, $\mathcal{I}_k = objets(k)$. À noter que cet ensemble se limite aux seules instances connues de k au moment de la caractérisation. De même, l'ensemble des généralisations Υ_k est égal à l'ensemble des classes disponibles sur k à cet instant, $\mathcal{C}_k$. L'ordre de généralité est la *spécialisation*, $\leq_k^c$ qui suit l'inclusion entre les extensions des classes. Les extensions, ou plutôt leurs parties connues au SRCO sont rendues par une fonction `objets`: $\mathcal{C}_k \rightarrow 2^{\mathcal{I}_k}$. À noter qu'en TROEPS, `objets` est une primitive disponible via l'API du système. Finalement, les classes sont organisées en une taxonomie $\mathcal{H}_k = \langle \mathcal{C}_k, \leq_k^c \rangle$.

Les contraintes sur les relations en TROEPS sont décrites sous forme d'une liste de classes. Celles-ci appartiennent toutes au concept co-domaine de l'attribut relationnel a, mais à des points de vue différents. Le descripteur utilisé pour introduire la liste est `type`.

La généralisation est définie sur l'ensemble des parties de $\mathcal{I}_k$ qu'elle renvoie sur l'ensemble des classes $\mathcal{C}_k$, $gen_o \ : \ 2^{\mathcal{I}_k} \rightarrow \mathcal{C}_k$. Pour un ensemble d'objets dans k, soit $\{o_1, o_2, \ldots, o_n\}$, elle calcule la classe la plus spécifique dans $\mathcal{T}$ qui comprend tous les objets dans l'argument.

Définition 4.6.

$$gen_o(\{o_1, o_2, \ldots, o_n\}) = min_{\leq_k^c}\{c \in \mathcal{C}_k | \ \forall i = 1, n; o_i \in \texttt{objets}(c)\}.$$

.

Exemple

La table suivante présente le résultat de l'application de gen_o sur quelques ensembles d'objets de l'exemple précédent (section 4.2.2) par rapport à l'attribut `conjoint`.

Ensemble de valeurs S	$gen_o(S)$
`{o2.conjoint, o3.conjoint}`	`c2`
`{o4.conjoint, o1.conjoint}`	`c3`
`{o6.conjoint, o5.conjoint}`	`c4`

Il est à noter que la généralisation ne considère pas les classes par rapport à leurs descriptions. Cela permet de se servir de celles-ci en tant que généralisations possibles sans qu'elles en aient une. C'est notamment le cas lorsque la généralisation dans l'exemple précédent est effectuée lors du processus de caractérisation : les classes sont décrites en termes de classes qui font partie du même ensemble et sont donc en cours de caractérisation.

Ce fait important est utilisé par notre méthode de construction de classes afin de traiter la co-définition dans les structures de connaissances, concepts, classes ou objets.

Algorithme

La recherche efficace de la classe commune d'un ensemble d'objets au sein d'une taxonomie n'est pas une tâche d'une extrême difficulté. Toutefois, si elle est utilisée fréquemment, il y a intérêt à ce que cette tâche soit réalisée de la manière la plus efficace possible.

Ci-dessous, un algorithme de recherche est présenté qui est inspiré de [Ketterlin, 1995]. L'algorithme consiste à composer une liste de classes par objet dans l'ensemble à généraliser. Ensuite, les listes sont parcourues en parallèle, en partant de la classe en tête qui est la classe racine de la taxonomie. Le parcours cherche la première position sur laquelle deux listes divergent, soit j. La classe commune recherchée se trouve dans toutes les listes sur la position juste avant, $j - 1$.

L'algorithme est de complexité $O(n * m)$ où n est le nombre d'objets à généraliser et m la profondeur de la taxonomie $\mathcal{H}_k$. Dans le cas pire, c'est un traitement en n^2, mais dans les cas pratiques, la complexité se rapproche à une fonction linéaire de n (m est de l'ordre de $\log n$).

Discussion

La généralisation sur un attribut relationnel a été abordée dans le cadre de la *formation de concepts* structurés (voir section 2.5 du chapitre 2). Comme dans notre cas, les classes existantes, appelées concepts probabilistes, sont utilisées par la généralisation. La méthode de référence, celle du système LABYRINTH, préconise une généralisation plus fine que celle décrite ici, en arbitrant entre la spécificité et la complexité du calcul de celle-ci. Ainsi, la généralisation d'un ensemble d'objets est un ensemble de concepts. Le choix de cet ensemble est guidé par un critère numérique de couverture globale de l'ensemble d'individus (propre à l'approche probabiliste). La recherche de cet ensemble est exhaustive, toutes les combinaisons de concepts contenant, conjointement, la totalité des objets seront évaluées. Le choix de l'ensemble est guidé par un critère numérique

```

RECHERCHE-CLASSE({k_1, k_2, ..., k_m})

pour i de 1 à n faire
   c_i := classe(o_i)
   CL_i := sur-classes(c_i)
identique := vrai
c := tête(CL_1)
tant que identique faire
   classe := c
   pour tout concept k_i dans C faire
      identique := identique et (c = tête(CL_i))
      c := tete(CL_1)
      CL_1 := queue(CL_1)
retourner classe
```

Algorithme 6: Recherche de la classe commune la plus spécifique d'un ensemble d'objets

de couverture globale de l'ensemble d'individus (propre à l'approche probabiliste). La recherche de cet ensemble est exhaustive, toutes les combinaisons de concepts contenant, conjointement, la totalité des objets, seront évaluées. Dans [Ketterlin et Korczak, 1994], une approche identique à la nôtre est proposée, consistant à ne retenir qu'un seul concept pour la généralisation, le plus spécifique possible.

Nous avons privilégié la simplicité dans les descriptions obtenues par la généralisation , ce qui garantit également une complexité correcte pour les procédures de calcul. Le choix que nous avons fait n'est pas le seul possible, une généralisation guidée par un seuil serait possible. Toutefois, le gain en précision escompté dans les descriptions saurait difficilement compenser l'augmentation de la complexité. En effet, celle-ci est estimée comme étant une fonction exponentielle du nombre des objets à généraliser[22].

4.4 Description sur les attributs multivalués

Les attributs de nature propriété multivalués dans TROEPS ont pour valeur des collections construites à l'aide des collecteurs `ensemble` et `liste` sur un type de base. La généralisation sur des collections des valeurs est liée à certaines difficultés. Principalement, il est difficile d'imaginer un langage d'hypothèses ayant en même temps les propriétés de concision et précision. De plus, celui-ci devrait disposer d'un moyen de calcul efficace pour la relation de sous-typage entre types décrits qu'il sera nécessaire d'établir.

4.4.1 Définitions

Soit l'attribut multivalué a de type de base τ et constructeur `cons(`a`)` qui est `ensemble` ou `liste`. Dans la suite, les ensembles seront notés entre accolades, $\{\}$, et les listes entre chevrons, $\langle\rangle$.

Le type τ est soit un concept k, soit un type de données T. Soit encore Υ_τ l'ensemble des généralisations, des types ou des (ensembles de) classes admissibles sur τ.

22. des améliorations sont suggérées dans [Thompson et Langley, 1991], mais aucun résultat n'a été publié par la suite.

Nous considérons deux fonctions sur l'ensemble Υ_τ. La première est une fonction d'« aplatissement », `elements`. Elle est définie sur les ensembles ou les listes d'éléments dans le type de base, soit 2^τ et Λ^τ respectivement, ainsi que sur toute combinaison de constructeurs ayant pour type de base τ. La fonction fournit l'ensemble des membres d'une collection qui sont éléments de τ. À noter que ces éléments peuvent se trouver à une profondeur quelconque dans la structure de la collection.

Exemple

Collection z	`elements`(z)
$\{2, 7, 12, 13\}$	$\{2, 7, 12, 13\}$
$\langle c, o, l, l, a, n, t\rangle$	$\{a, c, l, n, o, t\}$
$\{\{2, 7, 12, 13\}, \{2, 11, 12, 15\}, \{7, 15\}\}$	$\{2, 7, 11, 12, 13, 15\}$
$\{\langle 2, 7, 2, 12\rangle, \langle 11, 12, 7, 13\rangle\}$	$\{2, 7, 11, 12, 13\}$

La seconde fonction, « cardinalité », notée `card`, est définie sur les ensembles ou les listes construits par une application d'un nombre quelconque de constructeurs sur le type de base. Elle renvoie le nombre des membres du niveau le plus haut.

Exemple

Collection z	`card`(z)
$\{2, 7, 12, 13\}$	4
$\langle c, o, l, l, a, n, t\rangle$	7
$\{\{2, 7, 12, 13\}, \{2, 11, 12, 15\}, \{7, 15\}\}$	6
$\{\langle 2, 7, 2, 12\rangle, \langle 11, 12, 7, 13\rangle\}$	5

Le *langage des généralisations* $\Upsilon_{\mu(\tau)}$ est composé par des paires dans : $\Upsilon_{\mu(\tau)} = \mathcal{J}_{\mathbb{N}} \times \mathcal{J}_{\Upsilon_\tau}$. Les deux éléments de chaque paire sont des intervalles : le premier est un intervalle d'entiers, alors que le second un intervalle d'ordre dans Υ_τ.

Soit un ensemble $\{z_1, z_2, .., z_n\}$ dont les éléments z_i sont des collections d'un type de base τ ($z_i = \{e^i_1, e^i_2, .., e^i_{m_i}\}$). Soient encore les nombres m_i les cardinalités des collections respectives $m_i = card(e_i)$. Le premier intervalle inclut toutes les cardinalités des éléments dans la collection, il est donc égal à :

$$[\min_{i=1,n}\{m_i\} \ \max_{i=1,n}\{m_i\}].$$

Le second intervalle est un *intervalle d'ordre partiel* dans la taxonomie des généralisations sur τ, $\mathcal{H}_\tau = \langle \Upsilon_\tau \leq_\tau\rangle$. Les bornes de l'intervalle sont obtenues comme généralisations de deux ensembles d'éléments dans τ. Le premier ensemble, $e_{\min}$ est le minimum des ensembles d'éléments résultant de l'aplatissement de chaque collection :

Définition 4.7. *L'ensemble* $e_{\min}$ *est l'intersection des ensembles d'éléments de chaque collection, soit :*

$$e_{\min}(\{z_1, z_2, .., z_n\}) = \bigcap_{i=1}^{n} elements(z_i).$$

De manière duale, $e_{\max}$ est le suprémum des ensembles d'éléments.

Définition 4.8. *L'ensemble* $e_{\max}$ *est l'union des ensembles d'éléments de chaque collection, soit :*

$$e_{\max}(\{z_1, z_2, .., z_n\}) = \bigcup_{i=1}^{n} elements(z_i).$$

Dans la suite, lorsque cela ne crée pas d'ambiguïté, nous allons omettre les arguments de $e_{\max}$ et $e_{\min}$ afin de gagner en concision.

La borne inférieure du second intervalle est la généralisation de $e_{\min}$ sur τ, et la borne supérieure la généralisation de $e_{\max}$. En somme, la généralisation de l'ensemble des collections $\{z_1, z_2, .., z_n\}$ est définie comme suit :

Définition 4.9.

$$gen_m(\{z_1, z_2, .., z_n\}) = \langle [\min_{i=1,n}\{m_i\} \;\; \max_{i=1,n}\{m_i\}], [gen_\tau(e_{\min}) \;\; gen_\tau(e_{\max})] \rangle$$

Les généralisations s'interprètent comme l'ensemble de toute collection qui contient au moins un élément inclus dans l'interprétation de $gen_\tau(z_{\min})$ et dont tous les éléments sont inclus dans l'interprétation de $gen_\tau(z_{\max})$. De plus, le nombre des ses éléments varie entre les bornes de l'intervalle en première position dans la paire.

Correction de la définition

Dans ce paragraphe nous allons montrer que notre opérateur de généralisation est correctement défini. Tout d'abord, la propriété suivante assure la consistance de la structure comme un produit d'intervalles. Le premier intervalle est manifestement correct, ses bornes sont énumérées dans le bon ordre. La correction du second intervalle est assurée par la propriété ci-dessous.

Le but est de démontrer que les deux généralisations d'un intervalle obtenu par gen_m sont liées par une relation de généralité. Autrement dit, quel que soit l'ensemble de collections généralisé, le résultat vérifie :

Proposition 4.1. *Pour un ensemble de collections $\{z_1, z_2, .., z_n\}$ sur un type de base τ, si sa généralisation est $\langle [a_l \;\, a_h], [g_l \;\, g_h] \rangle = gen_m(\{z_1, z_2, .., z_n\})$, alors $g_l \leq_\tau g_h$.*

Le sous-typage sur $\Upsilon_{\mu(\tau)}$

Nous définissons la relation de *sous-typage* sur $\Upsilon_{\mu(\tau)}$ comme l'ordre-produit des ordres sur $\mathcal{J}_{\mathbb{N}}$ et sur $\mathcal{J}_{\Upsilon_\tau}$.

Soient $\langle [a_l \;\, a_h][g_l \;\, g_h] \rangle$ et $\langle [a'_l \;\, a'_h][g'_l \;\, g'_h] \rangle$ deux descriptions dans $\Upsilon_{\mu(\tau)}$. Notons $\leq_{\mu(\tau)}$ le sous-typage. Alors $\leq_{\mu(\tau)}$ est défini comme suit :

Définition 4.10. $\langle [a_l \;\, a_h][g_l \;\, g_h] \rangle \leq_{\mu(\tau)} \langle [a'_l \;\, a'_h][g'_l \;\, g'_h] \rangle$ *ssi :*

- $a'_l \;\leq\; a_l \;\leq\; a_h \;\leq\; a'_h$
- $g'_l \;\leq_\tau\; g_l \;\leq_\tau\; g_h \;\leq_\tau\; g'_h$

La propriété suivante assure l'adéquation entre les deux définitions précédentes. En outre, elle met en évidence la correction de la définition 4.10 en tant qu'opérateur de généralisation :

Proposition 4.2. *Pour deux ensembles A et A' de collections sur le type de base τ, $A \subseteq A'$ implique $gen_m(A) \leq_{\mu(\tau)} gen_m(A')$.*

Dans la suite, nous allons décrire la manière dont les opérateurs de généralisation s'instancient dans les différents cas de types de base τ. Le constructeur `ensemble` est considéré en priorité, tout en soulignant les éventuelles différences avec la manière de traiter le constructeur `liste`.

4.4.2 Description sur les propriétés

L'effet de l'opérateur de généralisation, gen_m, est tout d'abord illustré sur un exemple d'ensembles d'entiers.

Exemple Soient les trois ensembles d'entiers suivants :

- $e_1 = \{2, 7, 12, 13\}$,
- $e_2 = \{2, 11, 12, 15\}$ et
- $e_3 = \{2, 12\}$.

Ici $e_{\min}$ est [2 12] et $e_{\max}$ [2 15]. L'intervalle de variation pour les cardinalités est [2 4]. En conséquence, la généralisation de ces ensembles est définie comme suit :

$$gen_m(\{e_1, e_2, e_3\}) = \langle [2\ 4], [[2\ 12]\ [2\ 15]]. \rangle$$

Les listes de valeurs d'un type énuméré se généralisent de manière semblable. À noter que les bornes du second intervalle restent des ensembles.

Exemple Soient les trois listes de caractères suivantes :

- $l_1 = \langle l, o, t \rangle$,
- $l_2 = \langle c, o, l, l, a, n, t \rangle$ et
- $l_3 = \langle c, o, l \rangle$.

Avec $e_{\max}$ de $\{a, c, l, n, o, t\}$ et $e_{\min}$ de $\{c, l, o\}$, ainsi que [3 6] pour l'intervalle de variation des cardinalités, la généralisation est :

$$gen_m(\{l_1, l_2, l_3\}) = \langle [3\ 6], [\{c, l, o\}\ \{a, c, l, n, o, t\}] \rangle.$$

Algorithme

L'algorithme de calcul pour la généralisation est intuitif : les collections sont tout d'abord aplaties, ensuite l'intersection et l'union des ensembles sont calculées, ainsi que l'intervalle de variation. La procédure est d'une complexité en temps quadratique en nombre total d'éléments dans toutes les collections. Toutefois, il est possible de formuler, moyennant diverses optimisations sur les opérations ensemblistes, des procédures de complexité quasi-linéaire, comme nous l'avons indiqué dans la section précédente.

4.4.3 Description sur les relations

L'opérateur de généralisation, gen_m, sur les objets est illustré sur des exemples d'ensembles et de listes d'objets. Pour cela, nous nous servons de l'ensemble d'objets décrit par la table 4.1. d'entiers.

Exemple Soient les trois ensembles d'objets suivants :

- $e_1 = \{o_4, o_5, o_6\}$,
- $e_2 = \{o_1, o_4\}$ et
- $e_3 = \{o_2, o_3, o_4, o_5\}$.

Les ensembles $e_{\min}$ et $e_{\max}$ sont $\{o_4\}$ et $\{o_1, o_2, o_3, o_4, o_5, o_6\}$ respectivement. Les cardinalités varient dans [2 4]. La généralisation des trois ensembles est donc :

$$gen_m(\{e_1, e_2, e_3\}) = \langle [2\ 4], [c5\ c1] \rangle.$$

La généralisation sur les listes d'objets est définie de façon identique.

Exemple Soient les trois listes d'objets suivantes :

- $l_1 = \langle o_4, o_5, o_6, o_5 \rangle$,
- $l_2 = \langle o_5, o_4, o_3, o_4 \rangle$ et
- $l_3 = \langle o_1, o_2, o_3, o_4, o_5 \rangle$.

Avec $e_{\max}$ de $\{o_1, o_2, o_3, o_4, o_5\}$ et $e_{\min}$ de $\{o_4, o_5\}$ et intervalle de variation pour les cardinalités de [4 5], la généralisation est :

$$gen_m(\{l_1, l_2, l_3\}) = \langle [4\ 5], [c5\ c1] \rangle.$$

Il est à noter que les ensembles des objets membres d'un ensemble de collections n'ont pas forcément une intersection non vide. Dans ce cas, la borne inférieure du second intervalle est $\varnothing$.

Exemple Les deux ensembles d'objets suivants :

- $e_1 = \{o_4, o_6\}$ et
- $e_2 = \{o_1, o_5\}$

se généralisent en :

$$gen_m(\{e_1, e_2\}) = \langle [2\ 2], [\varnothing\ c2] \rangle.$$

L'opération sur les ensembles ne nécessite pas la définition de nouveaux algorithmes car elle se réduit en des calculs qui ont été décrits préalablement.

4.4.4 Discussion

Le langage de généralisation ne permet pas d'exprimer des régularités dans les collections car une seule expression peut être retenue ; toutefois, c'est pour des raisons de simplicité des calculs de sous-typage.

4.5 Description d'une classe entière

La définition d'un éventail complet d'opérateurs de généralisation sur les attributs permet d'affirmer la complétude de notre procédure de généralisation. Quant à la correction de notre définition, elle fait objet de la proposition suivante.

Proposition 4.3. *Pour un couple d'ensembles d'objets O et O' en provenance d'un concept k, si $O \subseteq O'$, alors $GEN_o(O) \leq_k GEN_o(O')$.*

Exemples La figure 4.2 illustre le résultat de l'application de l'opérateur de généralisation $GEN_o()$ sur les classes de l'exemple présenté dans la section 4.2.2.

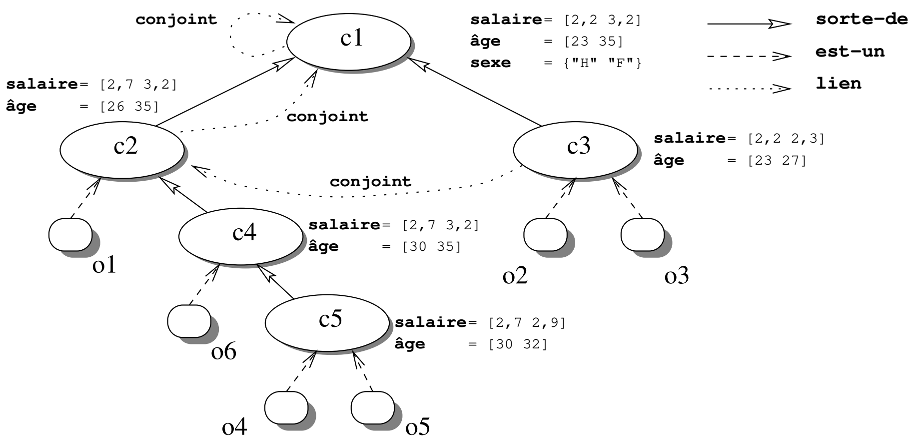


FIG. 4.2: *Le résultat de la caractérisation des classes de la figure 4.1. Les contraintes que les classes héritent ne sont pas visualisées pour des raisons de concision : les liens* `conjoint` *et la propriété* `sexe`.

4.6 Conclusion

Nous avons présenté un jeu d'opérateurs de généralisation dont le but est de produire des descriptions génériques à partir d'un ensemble de descriptions individuelles. Les généralisations sont décrites par un langage qui a quasiment le pouvoir descriptif du langage de description des connaissances de TROEPS. Les généralisations produites sont maximalement spécifiques et convexes. Un tel choix permet une double simplification : d'une part, les résultats sont plus faciles à interpréter par l'utilisateur et, d'autre part, la manipulation des descriptions est simplifiée, en particulier lorsqu'il s'agit de calculer le sous-typage qui est une opération très fréquemment utilisée dans un SRCO. Ainsi, un gain de temps de traitement est réalisé, qu'il soit automatique ou manuel.

En outre, la généralisation sur les attributs multivalués que nous avons proposée est une notion originale d'un problème encore très peu exploré. Les définitions sont en complet accord avec les objectifs posés en début de la partie II du mémoire. En particulier, nous avons assuré la **complétude** grâce à une définition de la généralisation à deux niveaux : au niveau des attributs monovalués, sur l'ensemble des catégories de types dans TROEPS (cet ensemble étant complet par rapport aux types de données utilisés en analyse de données ou en apprentissage automatique) et au niveau des constructeurs, *tuple* pour les objets pris comme des structures d'attributs et *liste* ou *ensemble* pour les collections.

La **composabilité** est également assurée car des descriptions d'objets de complexité potentiellement illimitée peuvent être traitées. En particulier, la définition de l'opérateur sur les collections, permet une application sur des collections dont les membres sont d'autres collections. En revanche, cette approche ne permet pas de préserver la homogènéité des contraintes de type en TROEPS. Cependant, une plus grande rigueur dans la définition de la généralisation conduit nécessairement à des descriptions fort complexes, ce qui contredit notre objectif d'intelligibilité.

En somme, nous avons accepté de perdre de la précision, ce qui aurait été inadmissible si le but avait été un gain de prédictivité[23].

23. Nous avons la tendance de penser que le fait de préserver des informations plus riches que l'enveloppe convexe d'un ensemble d'entités est une capacité importante pour un système de regroupement. Cela semble être à l'origine du relatif succès de systèmes comme COBWEB et AUTOCLASS en ECD.

Finalement, dans ce chapitre, nous avons mis en évidence la possibilité de caractériser les classes à l'aide d'autres classes qui appartiennent au même ensemble.

5

Extractions des ressemblances entre objets

Introduction

Après la description d'une méthode de construction pour les classes d'objets au chapitre 3 suivie par l'introduction d'un ensemble d'opérateurs de généralisation, nous nous tournons vers la définition d'une mesure de dissimilarité qui est capable de travailler sur les objets en toute généralité. Compte tenu de la complexité de la tâche qui reflète la complexité de la structure des objets, la définition s'étale sur les trois prochains chapitres.

Afin de faciliter la lecture, dans le présent chapitre nous adoptons un cadre simplifié. Celui-ci ne comporte pas d'attributs multivalués. Les aspects calculatoires sont également simplifiés, en laissant de côté les problèmes liés à la circularité dans les descriptions des objets. Ainsi, nous nous concentrons sur la définition détaillée de la mesure de dissimilarité de base, en insistant sur son adéquation vis à vis de la structure des objets.

Le principe fondateur de la mesure qui consiste à appuyer la comparaison des objets sur leur généralisation permet l'application de cette mesure sur les descriptions des objets dans leur intégralité et cela de manière uniforme.

Le chapitre commence par un bref rappel des objectifs que la mesure de dissimilarité doit satisfaire dans notre contexte (section 5.1). Il est suivi par la présentation des principes qui ont fondé la définition de la mesure (section 5.2). Les deux définitions génériques qui permettent de réaliser les principes fixés, la mesure de dissimilarité entre objets et la mesure d'écart entre attributs sont proposées dans la section 5.3. Elles s'instancient différemment sur chaque attribut en fonction de sa nature et son type. Les différentes instanciations sont décrites dans les sections qui suivent : section 5.4 pour les attributs propriétés et section 5.5 pour les attributs relations. Finalement, une extension de la mesure d'objets sur les classes est esquissée (section 5.6).

Les deux aspects de la description des objets que nous avons exclus de ce chapitre, à savoir les attributs multivalués et les relations, sont progressivement intégrés au modèle proposé dans la suite. Les problèmes, les solutions que nous avons retenues, ainsi que les questions ouvertes qui découlent de ces intégrations sont discutés dans les chapitres suivants (chapitres 6 et 7 respectivement).

5.1 Objectifs

Au début de cette partie du mémoire, nous avons identifié quelques exigences que les éléments de notre approche doivent satisfaire. Comme le poids de la réalisation de l'approche est porté essentiellement par les outils, il n'est pas étonnant que ceux-ci soient soumis à un grand nombre de contraintes. En ce qui concerne la mesure de dissimilarité qui est l'outil principal de l'approche, elle a la charge d'assurer surtout la *complétude*, la *cohérence* et la faisabilité de la construction de classes.

La satisfaction des exigences ci-dessus est détaillée dans la suite.

5.1.1 Complétude et extensibilité

Afin d'assurer la **complétude** de la construction de classes sur les descriptions des objets, il faut s'assurer tout d'abord que les mesures d'écarts définies admettent la totalité de ces descriptions (en termes de type, constructeur et nature d'un attribut). Pour cela, on a le choix entre une énumération exhaustive de tous les cas de figure et une définition générique qui s'instancie sur tous les attributs d'une manière qui leur est propre. Dans ce cas, la définition doit s'appuyer sur un principe de comparaison universel, permettant d'abstraire les différences entre attributs. Nous avons opté pour la seconde solution en articulant la définition de la mesure autour de l'opération de généralisation . Ainsi, deux entités, objets ou valeurs simples, sont comparées par rapport à leur généralisation.

Un autre aspect de la complétude concerne la structuration des descriptions. Ainsi, il faut s'assurer non seulement que les constituants de base des descriptions, c'est-à-dire les attributs mono-valués, son universellement admis, mais aussi que toutes les structures qu'il est possible de créer à partir de ces attributs – descriptions d'objets simples ou d'objets complexes, collections – peuvent être traitées de manière directe. Cela relève du choix des fonctions de dissimilarité qui combinent les fonctions d'écart au niveau de ces structures.

La complétude est en lien étroit avec l'**extensibilité** de la mesure qui veut que celle-ci soit facilement adaptable sur de nouveaux éléments de la description. L'extensibilité peut être réalisée par un choix approprié du principe de comparaison : il suffit que celui-ci soit facile à définir sur les nouveaux descripteurs.

5.1.2 Composabilité

La possibilité de mise à l'échelle de la construction de classes par rapport à la complexité de la structuration des objets que nous avons appelée la *composabilité* (des outils) est également fonction de la définition de la mesure. Plus que la complétude, elle garantit que lors d'une exploration récursive de la structure des objets, les outils, en particulier la mesure de dissimilarité, peuvent s'appliquer sur chaque niveau de cette structure, et que les résultats de cette application peuvent être combinés de manière cohérente. Cet objectif, propre aux formalismes de description structurés, est à satisfaire par une uniformisation des outils permettant, en particulier, de remplacer un outil sur les attributs par un outil sur les objets.

5.1.3 Cohérence

La *cohérence* de la méthode de construction de classes dépend de la définition dont il est attendu qu'elle établisse le pont entre les étapes de regroupement et de caractérisation des classes. La cohérence est réalisée par le principe de comparaison qui fonde la dissimilarité sur la généralisation.

5.1.4 Faisabilité

Finalement, la *faisabilité* de l'approche entière dépend de l'efficacité du calcul de dissimilarité. Le traitement des descriptions est une tâche intrinsèquement complexe. Toutefois, son coût peut être réduit en s'appuyant sur les structures de classes disponibles dans une base de connaissances. En particulier, elles peuvent être mises à profit dans l'évaluation des ressemblances entre leur objets membres.

5.2 Principes de la mesure

5.2.1 Vision sur les objets

Notre mesure suit la définition inductive des concepts dans TROEPS dont il a été question au chapitre 4. Elle permet de voir chaque concept k le résultat de l'application d'un constructeur de tuples sur un ensemble d'autres concepts et de types de données. Ces derniers deviennent les co-domaines des attributs de k. En conséquence, l'ensemble des concepts d'une base de connaissances peut être défini par l'application successive du produit sur les ADT et sur les concepts obtenus aux pas précédents. Ainsi, le concept `humain`, ayant des attributs `âge` et `salaire` est obtenu comme le résultat du produit des types de données `entier` (pour `âge`) et `réel` (pour `salaire`).

Cela n'est pas sans rappeler la vision des logiques de descriptions, et plus particulièrement, les constructeurs de termes `and` et `all`. D'une manière très approchée, on pourra affirmer que dans TROEPS le `and` qui exprime la subsomption directe s'applique exclusivement aux classes, alors que le `all` qui exprime les restrictions de valeurs peut s'appliquer aux concepts et aux classes. Notre contexte est tout de même un peu plus pauvre car toutes les combinaisons possibles ne sont pas autorisées.

En suivant [Capponi, 1997], nous distinguons deux types de constructeurs sur les objets : le constructeur de *tuples* qui sert dans la définition des concepts et les constructeurs d'*ensembles* et de *listes* qui sont utilisés dans la définition d'attributs multivalués. La différence principale est que le premier s'applique sur un ensemble hétérogène d'arguments, ADTs et concepts (donc autres tuples), alors que les constructeurs de collections s'appliquent sur un type unique.

En conséquence, notre dissimilarité doit être définie à deux niveaux : au niveau des descriptions construites à l'aide d'un constructeur et au niveau des descriptions constituantes (qui peuvent être construites à leur tour). En ce qui concerne la dissimilarité entre objets, elle doit comporter deux sortes de fonctions :

- fonctions sur les objets, d^o
- fonctions sur les attributs, propriétés et relations, δ_a

Les fonctions d^o sur un concept sont obtenues à partir des fonctions δ_a sur les attributs du concept en suivant la sémantique du constructeur de tuples.

5.2.2 Comparaison de deux entités au sein d'un domaine

Le principe de comparaison de deux entités est de se servir pour cela de leur description commune, c'est-à-dire leur généralisation.

Cela veut dire que dans la définition des fonctions de dissimilarité d^o, pour un couple d'objets o et o' nous nous appuyons sur l'opérateur $GEN_o(o, o')$. De plus, ce principe s'appliquant récursivement sur la définition de $GEN_o(o, o')$, les fonctions d'écart sont également calculées sur les généralisations. Ainsi, pour un couple de valeurs v et v' d'un attribut a, les écarts traitent $gen_a(v,v')$. Comme nous allons le voir dans la suite, le résultat de l'application de $gen_a(v,v')$ n'est pas toujours défini. C'est notamment le cas lorsque a est une relation circulaire entre les objets $o_v = v$ et $o'_v = v'$ sont en co-définition avec o et o' respectivement. Dans ce cas, $GEN_o(o_v, o'_v)$ est substitué à $gen_a(o_v, o'_v)$ (voir section 5.5).

L'idée d'appuyer la dissimilarité sur la généralisation est très ancienne, elle est à la base des distances d'*édition*. Plus récemment, elle a été utilisée dans la comparaison de descriptions relationnelles (voir par exemple [Teccuci et Kodratoff, 1989]).

5.2.3 Équité entre attributs

Les attributs des objets dans une base de connaissances expriment les caractéristiques des entités modélisées de la manière la plus fidèle possible. Quand il s'agit de comparer les objets moyennant une mesure de proximité, les écarts sur chaque attribut sont cumulés. Ces écarts ne peuvent pas être pris dans leur forme d'origine au risque de privilégier artificiellement certains attributs aux dépens d'autres.

Soient, par exemple, les deux attributs `loyer` et `étage` du concept `appartement` de type entier et de domaines $[2000, 5400]$ et $[0, 30]$ respectivement. Il est clair que s'ils sont pris tels quels dans une fonction de proximité, les écarts sur `loyer` vont avoir un impact beaucoup plus fort sur la valeur finale que les écarts sur `étage`.

Afin de remédier à ce déséquilibre artificiel, il est d'usage lors du calcul de la proximité de ramener tous les écarts dans l'intervalle $[0, 1]$. Ceci peut être fait de manière variable. En statistique, par exemple, c'est la description des individus qui est modifiée. Ainsi, une procédure, appelée *standardisation* est effectuée sur tous les attributs à échelle numérique. Elle consiste en un centrage et une réduction des valeurs de l'attribut. Le premier représente la soustraction de la moyenne de l'attribut de chaque valeur. Le résultat est ensuite réduit en le divisant par l'écart-type de l'attribut. Les données ainsi transformées représentent un nuage de points dont le centre de gravité est l'origine.

En apprentissage à base de similarité, un autre procédé est utilisé pour les attributs numériques, entiers ou réels. Ceux-ci ne sont pas transformés directement mais lors du calcul des similarités, les écarts sont divisés par l'étendue du domaine de l'attribut sous-jacent qui représente l'écart maximum.

Dans notre cas, la normalisation est intégrée à la procédure de calcul des écarts. En effet, lors de la description de chaque fonction nous avons assuré que la valeur rendue sera comprise dans $[0, 1]$. Ceci a été obtenu en divisant systématiquement la valeur absolue $\delta_a^f(v, v')$ par la valeur maximale possible sur le co-domaine de a, $\max_{cod(a)}(\delta_a^f)$.

5.2.4 Pertinence des attributs

Il est généralement admis que tous les attributs n'ont pas la même importance pour la construction de classes. Ainsi par exemple, si l'on doit regrouper des appartements en vue de retrouver des classes d'appariements qui peuvent intéresser le même type de locataires, les attributs comme `loyer` où `surface` vont avoir un degré de pertinence plus élevé que les attributs `prix` ou `propriétaire`.

Afin d'assurer la construction de classes pertinentes, les différences de pertinence doivent être reflétées par la procédure de regroupement. Dans la première partie de ce document nous avons montré quelques approches vers l'attribution d'un rôle variable aux attributs lors du regroupement. Dans les cas du regroupement par ressemblances le degré de pertinence est exprimée par le biais de `poids` numériques. Ainsi, chaque attribut reçoit un poids non-négatif qui contrôle la contribution des écarts sur celui-ci dans le calcul de la proximité totale d'un couple objets.

L'établissement de ces poids est une tâche délicate et a fait l'objet de multiples critiques adressées aux méthodes basées sur les ressemblances, essentiellement pour deux raisons. D'une part, elle oblige la présence d'un spécialiste du domaine pour fournir ces poids au système et cela possiblement tout au long du processus d'analyse. Ceci ne représente pas une restriction dans notre propre cas où l'outil est destiné à un usage de la part des spécialistes et non pas à un fonctionnement autonome. D'autre part, les valeurs des poids, difficiles à établir avec précision, sont décisives pour le résultat du calcul de la dissimilarité et par là la structure de la taxonomie finale. Ainsi, dans certains cas, de fortes fluctuations dans cette structure seraient enregistrées à la suite de changements minimaux dans les poids. Nous estimons que ce phénomène, lorsqu'il se produit est plutôt indicateur de la forte proximité entre certains individus qu'il est difficile de discerner. C'est un fait objectif, il ne dépend pas de la méthode utilisée pour le regroupement et tout autre approche aurait rencontré les mêmes difficultés pour peu qu'elle puisse faire varier ses critères de regroupement. Avoir des résultats qui varient fortement n'est pas forcément un inconvénient et est en tout cas un prix à payer pour la flexibilité dans le regroupement.

Finalement, il faut noter que même les critiques les plus sévères ne nient pas le besoin de différencier les caractéristiques des individus en fonction de leur pertinence. Du coup, dans les applications à des domaines réels, il est souvent nécessaire d'introduire des poids dans des méthodes qui ne les incluaient pas d'origine [Vasco *et al.*, 1996; Bournaud, 1996].

Calcul en mode récursif

Nous l'avons vu, il est parfois nécessaire de calculer les valeurs de la dissimilarité en tant que solutions d'un système d'équations (voir chapitre 7). Cela implique certaines contraintes sur la forme de l'opérateur de composition utilisé dans la définition de la fonction de dissimilarité. En particulier, des combinaisons non-homogènes ne semblent pas convenir car les équations obtenues sont difficiles à résoudre. De plus, les fonctions de dissimilarité de la famille des *distances Euclidiennes* utilisent une matrice de coefficients pour la combinaison des écarts (voir par exemple [Saporta, 1990]). Les éléments de cette matrice n'ont pas toujours un sens lorsqu'il s'agit de traiter des individus décrits par des attributs hétérogènes. Ainsi, alors que dans les *distances de Minkowski* une matrice diagonale d'éléments égaux à 1 est utilisée, les matrices M employées par d'autres membres de la famille Euclidienne reposent sur des calculs de variance (pour la distance de *Mahalanobis*), des fréquences marginales (pour la distance de χ_2), etc. En conséquence, nous avons éliminé *a priori* ces mesures de distance en ne retenant que des membres de la famille de *Minkowski*.

En revanche, d'autres opérateurs peuvent être envisagés grâce à leur lien avec la constitution de généralisations. La question du choix d'opérateur de combinaison est discutée dans le paragraphe suivant.

5.3 Définitions génériques

Dans cette section nous précisons les définitions des fonctions génériques sur chaque niveau, les objets et les attributs. Celles-ci sont ensuite instanciées pour chaque cas concret.

5.3.1 Opérateurs de combinaison sur les tuples

Notations

Soient k un concept et o et o' deux instances de k. Soit encore A l'ensemble des attributs de k, `cpt-attributs`(k). Dans la suite, les poids des attributs sont notés par λ et indexés par le nom de l'attribut. Afin d'assurer la normalisation, la somme des poids est fixée à 1 :

$$\sum_{a \in A} \lambda_a = 1 \tag{5.1}$$

Nous cherchons donc une fonction $d_k^o : \ k \times k \rightarrow \mathbb{R}$ qui est calculée sur l'ensemble des écarts $\delta_a^f(o.a, o'.a)$ sur les valeurs des attributs a dans A :

$$d_k^o(o, o') = \biguplus_{a \in A} (\delta_a^f(o.a, o'.a), \lambda_a) \tag{5.2}$$

où $\biguplus$ est un opérateur permettant de combiner des valeurs à l'aide de poids fixés auparavant. Dans la suite, deux instanciations possibles de cet opérateur sont considérées, produit et somme, avec leurs interprétations vis à vis de notre principe.

Produit pondéré

Une première variante de l'opérateur de combinaison est un produit pondéré. Ainsi, les écarts sont des termes multiplicatifs dont la contribution est modulée par des exposants.

$$d_k^o(o, o') = \prod_{a \in A} \delta_a^f(o.a, o'.a)^{\lambda_a} \tag{5.3}$$

La formule ci-dessus revient à prendre une moyenne géométrique généralisée. Celles-ci se traduit en langage des objets comme *le volume*, relatif et pondéré sur les dimensions, de la généralisation $GEN_o(o, o')$. En effet, $\delta_a^f()$ évalue chacun des attributs par rapport à la généralisation admise par les valeurs $o.a$ et $o'.a$, $gen_a(o.a, o'.a)$. Or, la généralisation sur les objets comparés n'est autre que le produit des généralisations sur les attributs. On en déduit donc que le résultat du calcul de $d_k^o(o, o')$ selon la formule 5.3 est exactement le volume modulo les poids des dimensions exprimés en tant qu'exposants.

Les valeurs extrêmes de la fonction d^o sont atteintes dans les cas suivants :

- 0 — lorsqu'au moins un écart $\delta_a^f(o.a, o'.a)$ est nul,
- 1 — lorsque tous les écarts sont 1.

À noter que la première propriété de cet opérateur est en total désaccord avec l'intuition sur ce que doit être une fonction de dissimilarité. En effet, une telle définition produit des fonctions qui ne satisfont pas la propriété (3), l'*identité*, définie dans section 2.3 du chapitre 2.

Moyenne pondérée

Une autre possibilité pour l'opérateur de combinaison est de prendre non pas un produit, mais la somme pondérée des écarts. Ceux-ci deviennent alors des termes additifs et les poids apparaissent en tant que coefficients dans la somme. En d'autres termes, une combinaison linéaire des écarts est prise.

$$d_k^o(o, o') = \sum_{a \in A} \lambda_a * \delta_a^f(o.a, o'.a) \tag{5.4}$$

Cette seconde formule peut être interprétée comme le *volume* moyen pondéré de chaque dimension de la généralisation $GEN_o(o, o')$. On pourra le rapprocher, de loin, du périmètre d'un polygone dans le plan.

Les valeurs extrêmes de la fonction d^o selon la formule ci-dessus sont atteintes comme suit :

- 0 — lorsque tous les écarts $\delta_a^f(o.a, o'.a)$ sont nuls,
- 1 — lorsque tous les écarts $\delta_a^f(o.a, o'.a)$ sont 1.

Cette fois-ci, la propriété d'identité est manifestement vérifiée.

Discussion

Dans [de A. T. de Carvalho, 1998], une famille de mesures de dissimilarité sur les Objets Symboliques (voir [Diday, 1998]) est présentée. Les objets symboliques traités sont comparables aux classes que nous considérons ici car ils décrivent, en général, un ensemble d'individus d'un univers. Les mesures proposées se basent sur deux opérations dyadique sur les objets symboliques, l'*union* (*join* dans le texte) et l'*intersection* (*conjunction*), ainsi que sur une fonction, monadique, le *potentiel descriptif*. Cette dernière mesure le « volume» du produit Cartésien des ensembles des valeurs admissibles pour chaque attribut de l'objet symbolique. La dissimilarité de deux objets symboliques est exprimée en fonction du volume de leur union et intersection. Une définition de ces deux opérations peut être trouvée dans [Ichino, 1994].

Les travaux décrits ci-dessus se rapprochent des nôtres par la similitude entre l'opération d'union et la généralisation la plus spécifique dans les objets, $GEN_o()$. Quant à l'intersection, elle n'intervient pas directement dans notre propre définition car dans le cas des objets de représentation, l'intersection de deux objets est soit égale à l'union (si objets identiques), soit vide (si objets distincts).

Les fonctions de comparaison entre objets symboliques obtenues représentent clairement une généralisation de notre propre fonction d^o car elle permettent de comparer des classes entières par rapport à leur descriptions. Toutefois, ces fonctions souffrent de certains défauts quant à leur application sur des descriptions hétérogènes. Il semble, par exemple, que le mélange entre attributs discrets et continus ne suit aucun principe global : des fonctions de nature différente sont combinées sans trop de précautions (*cardinalité* pour les discrets et *étendue* pour les continus). En conséquence, les valeurs des fonctions sont assignées de manière arbitraire dans chaque cas. De plus, des informations relationnelles sont difficilement intégrales dans les définitions à cause de la nature du calcul.

Pour notre propre définition, nous avons retenu l'opérateur additif (la moyenne). Ce choix a été dicté par deux raisons principales. D'une part, celui-ci assure la propriété d'identité ce qui n'est pas le cas du produit. D'autre part, il coïncide avec la formule de la distance de *Manhattan*.

En conséquence, notre mesure devient une généralisation de cette dernière (car elle admet des descriptions relationnelles). En revanche, le résultat du calcul de la moyenne est plus difficile à interpréter que le produit qui peut être assimilé à un hyper-volume.

5.3.2 Écarts sur les attributs

Définitions

Soit l'attribut monovalué a défini dans le concept k' et de type $\tau = \texttt{type(a)}$. Le type τ est soit un concept k, soit un type de données T. Soit encore Υ_τ l'ensemble des généralisations admissibles sur τ (voir section 4.2 du chapitre 4). Υ_τ contient des types ou des classes qui sont des expressions génériques d'un ensemble d'éléments du $\mathcal{I}_\tau = \texttt{dom}(\tau)$, le domaine associé. Rappelons que l'opérateur $gen_\tau()$ calcule la généralisation d'un ensemble d'éléments dans τ. Inversement, pour un g donné, la fonction `membres` calcule l'ensemble des éléments de $\mathcal{I}_\tau$ que g désigne.

L'idée intuitive qui est à la base de notre définition est de mesurer la ressemblance entre deux individus e et e', objets ou valeurs de types simples, en fonction de la « taille» de leur généralisation $gen_\tau(e, e')$. Pour cela, nous introduisons les notions de *volume* et de volume *relatif*.

La notion de volume

Nous désignons par *volume* d'une généralisation $g \in \Upsilon_\tau$ la cardinalité de l'ensemble des individus que celle-ci désigne. La fonction volume sur un type τ, notée $\mathcal{V}_\tau : \mathcal{I}_\tau \rightarrow \mathbb{N}$, est donc définie comme suit.

Définition 5.1. *Le volume d'un g est,* $\mathcal{V}_\tau(g) = \| \texttt{membres(g)} \|$.

Il est à noter que la notion de volume n'est pas exploitable si τ est un type continu.

Ce constat conduit logiquement à la définition de la notion de volume relatif qui permet d'unifier tous les cas d'attributs.

Volume relatif

Définissons la fonction $\mathcal{V}^r_\tau$ (pour volume relatif) qui permet d'évaluer la « taille» relative d'une généralisation par rapport à une autre qui peut être l'équivalent du domaine total du type sous-jacent, $\mathcal{I}_\tau$.

Dans la suite, pour alléger les notations, nous adoptons les deux hypothèses suivantes :

– le volume relatif est systématiquement pris par rapport au domaine total,

– en cas de type de données numérique τ, le domaine a toujours des bornes finies ; cette condition peut être facilement satisfaite si l'on se restreint, pour un attribut donné de type τ aux valeurs effectivement présentes dans les objets considérés qui sont toujours en nombre finis.

Sous ses hypothèses, la fonction $\mathcal{V}^r_\tau : \Upsilon_\tau \rightarrow [0\ 1]$ fait correspondre à une généralisation (type ou classe) g dans Υ_τ un nombre réel qui indique la proportion des éléments dans g (valeurs ou objets) par rapport à l'ensemble total d'éléments dans τ.

Définition 5.2.

$$\mathcal{V}^r_\tau(t) = \frac{\mathcal{V}_\tau(t)}{\mathcal{V}_\tau(D_T)} \tag{5.5}$$

Selon la nature et le type τ, la fonction $\| \|$ s'instancie différemment et de là les deux fonctions $\mathcal{V}_\tau(t)$ et $\mathcal{V}^r_\tau(t)$. Ces instanciations sont discutées dans la section suivante.

Dans la suite, l'indice τ sera omis quand aucune ambiguïté n'est possible.

La formule générique

La formule générique pour deux entités, e et e', valeurs ou objets, d'un domaine exprime les proximités entre celles-ci comme la différence entre la taille relative de la généralisation des deux et la moyenne des tailles des deux généralisations singletons :

$$\delta^f_a(e, e') = \mathcal{V}^r(gen_\tau(\{e, e'\})) - \frac{\mathcal{V}^r(gen_\tau(\{e\})) + \mathcal{V}^r(gen_\tau(\{e'\}))}{2} \tag{5.6}$$

5.4 Écarts sur les propriétés

Dans ce cas, τ est un type de données T. La fonction d'écart s'instancie différemment sur les différents types de données. Sa forme dépend de l'instanciation de $\mathcal{V}^r$ qui distingue les types énumérés des types ordonnés qui sont ensuite distingués en continus et discrets.

5.4.1 Types continus

Ici, T est sous-type de `réel` et les généralisations admissibles sont des intervalles. Par ailleurs, il est courant de prendre pour $\| \|$ la *mesure de Lebesque* d'un ensemble de réels. Sur un intervalle elle revient à une soustraction de bornes :

$$\|[a \; b]\| \;=\; b \;-\; a \tag{5.7}$$

Soit un type $t = [a \; b]$ dans Υ_T. Soit encore $\mathcal{V}_\mathcal{L}$ la mesure de Lebesque sur T, alors $\mathcal{V}^r$ est le rapport entre les différences de bornes d'une part dans t et d'autre part dans l'intervalle recouvrant D_T :

$$\mathcal{V}^r(t) = \frac{\mathcal{V}_\mathcal{L}(t)}{\mathcal{V}_\mathcal{L}(D_T)} = \frac{b \;-\; a}{\max_{D_T} - \min_{D_T}} \tag{5.8}$$

La fonction $\mathcal{V}^r$ est donc le rapport entre les mesures de Lebesgue de l'ensemble de valeurs d'un intervalle, et l'ensemble de valeurs global, correspondant au type de données complet, soit D_T =`membres`(T). La fonction d'écart δ^f_a devient alors :

$$\delta^f_a(v, v') = \frac{\mathcal{V}_\mathcal{L}([\min(\{v, v'\}), \max(\{v, v'\})])}{\mathcal{V}_\mathcal{L}(D_T)} - \frac{1}{2} * (\frac{\mathcal{V}_\mathcal{L}([v \; v])}{\mathcal{V}_\mathcal{L}(D_T)} + \frac{\mathcal{V}_\mathcal{L}([v' \; v'])}{\mathcal{V}_\mathcal{L}(D_T)}) \tag{5.9}$$

En prenant en compte le fait que la mesure de Lebesgue vaut 0 pour un point sur l'axe réel et donc pour tout intervalle de bornes égales, nous obtenons que les deux sous-termes du second terme de la somme ci-dessus sont nuls. En conséquence, la formule 5.9 se simplifie en une soustraction normalisée sur les bornes :

$$\delta^f_a(v, v') = \frac{|v - v'|}{\max_{D_T} - \min_{D_T}} \tag{5.10}$$

La fonction ainsi décrite coorespond à la soustraction normalisée sur les nombres.

5.4.2 Types ordonnés discrets

Le cas discret d'un type ordonné T est similaire au cas continu, dans la mesure où le langage Υ_T est aussi composé par des intervalles. En revanche, les ensembles `membres` ont maintenant un nombre dénombrable d'éléments. Qui plus est, nous pouvons toujours nous ramener au cas où tous les intervalles sont à bornes finies puisque nous ne travaillons qu'avec des ensembles d'objets en nombre fini. Ainsi, la fonction $\|\ \|$ est le cardinal d'un ensemble fini, elle a donc des valeurs dans $\mathbb{N}$.

Cela permet de manipuler dans la fonction $\mathcal{V}_\tau$ des nombres entiers et donc de calculer la valeur de $\mathcal{V}^r$ comme le rapport de deux entiers. Soit D_T le domaine de T et les valeurs v_l et v_h respectivement le minimum et le maximum de D_T. Pour un type t qui se décrit par l'intervalle $[v\ v']$, la valeur de $\mathcal{V}^r$ est définie comme suit :

$$\mathcal{V}^r(t) = \frac{\|[\min(v',v),\max(v',v)]\|}{\|[v_l,v_h]\|} = \frac{|v-v'|+1}{v_h-v_l+1}. \tag{5.11}$$

À noter qu'ici la taille d'un intervalle singleton n'est pas zéro, mais 1. En conséquence, la valeur de l'écart sur un couple de valeurs dans T devient :

$$\delta_a^f(v,v') = \frac{|v-v'|}{v_h-v_l+1} \tag{5.12}$$

La fonction ci-dessus correspond à la soustraction normalisée d'entiers. La seule différence est la soustraction du 1 du numérateur ; elle est essentielle car elle assure la propriété d'identité de la fonction δ_a^f.

La soustraction traduit l'intuition que la dissimilarité entre deux entités est proportionnelle à l'effort nécessaire de passer d'une entité, à l'autre. La fonction ci-dessus possède la particularité de ne jamais atteindre la valeur extrême de 1. En effet, il est impossible de satisfaire les deux conditions suivantes :

- avoir au moins un couple v et v' tel que $\delta_a^f(v,v') = 1$,
- avoir pour tout v $\delta_a^f(v,v) = 0$.

5.4.3 Types énumérés

Contrairement au cas ordonné, ici le langage Υ_τ est composé par des ensembles de valeurs donnés en énumération. De plus, la généralisation de deux valeurs est un ensemble composés de ces deux valeurs, il y donc une cardinalité constante de deux. En conséquence, la fonction $\mathcal{V}^r$ renvoie, à partir d'une généralisation de deux valeurs, une constante. La constante est propre au type de données sous-jacent :

$$\mathcal{V}^r(\{v,v'\}) = \frac{\|\{v,v'\}\|}{||D_T||} = \frac{2}{||D_T||} \tag{5.13}$$

La fonction d'écart est également constante sur l'ensemble des couples de valeurs du type :

$$\delta_a^f(v,v') = \frac{2}{||D_T||} - \frac{1}{2}\left(\frac{1}{||D_T||} + \frac{1}{||D_T||}\right) = \frac{1}{||D_T||} \tag{5.14}$$

À noter que par rapport à la fonction qui est employée par les méthodes d'apprentissage automatique, voir d'analyse de données hétérogènes, qui est le complément de l'identité, soit $\delta_a^f = 1 - id_{D_T}$, notre fonction permet de prendre en compte la taille du domaine entier. Ainsi, deux valeurs non identiques sont d'autant plus proches selon δ_a^f que le domaine D_T est large. De cette manière, les domaines plus conséquents se voient associer des fonctions plus « fines », ce qui correspond à l'intuition.

5.4.4 Exemple

Ici nous reprenons le jeu de données de la table 4.1 du chapitre 4. Les objets sont des instances du concept `humain` considérés avec les deux attributs numériques, `salaire` et `âge`. Rappelons que l'attribut `salaire` du concept `humain` est de type réel, `intervalle [1,8 3,3]`, alors que l'attribut `âge` est de type entier, avec un étendue `intervalle [21 35]`.

Pour comparer les objets o_1 et o_4 selon l'attribut `salaire` il est nécessaire d'établir le volume relatif de la généralisation de leurs valeurs respectives, $2,7$ et 3. La généralisation est l'intervalle ayant pour bornes les deux nombres :

$$gen_{ord}(\{2,7\ ,\ 3\}) = [2,7\ 3].$$

Le volume relatif est le rapport entre cet intervalle et l'intervalle du domaine englobant :

$$\mathcal{V}^r(\{2,7\ ,\ 3\}) = \frac{3\ -\ 2,7}{3,3\ -\ 1,8}.$$

En suivant la formule 5.10, la valeur de l'écart est :

$$\delta_{sal}^f(o_1.\mathtt{salaire}, o_4.\mathtt{salaire}) = |3\ -\ 2,7|/(3,3\ -\ 1,8) = 0,2.$$

Voici quelques autres résultats de l'application de l'écart δ_{sal}^f,

Couple d'objets	**valeur** de δ_{sal}^f
(o_3, o_4)	$0,33$
(o_5, o_6)	$0,2$
(o_2, o_5)	$0,4$

Les écarts delon l'attribut `âge`, sont claculés de manière très similaire la seule différence étant que le volume relatif de la généralisation d'un nombre entier est supérieur à zéro. Pour comparer les objets o_1 et o_2 selon l'attribut `âge`, le volume relatif est calculé selon la formule 5.12 :

$$\delta_{\hat{a}ge}^f(26\ ,\ 23) = \frac{26 - 23}{35 - 21 + 1} = 0,2$$

Dans la table suivante, quelques autres valeurs de l'écart sur `âge` sont présentées :

Couple d'objets	**valeur** de $\delta_{\hat{a}ge}^f$
(o_3, o_4)	$0,2$
(o_5, o_6)	$0,2$
(o_2, o_5)	$0,6$

Pour combiner les écarts sur les attributs en une dissimilarité entre objets, prenons des poids égaux pour les deux attributs dans l'équation 5.4. La table ci-dessous présente la matrice des valeurs de d_h^o pour l'ensemble d'objets donné par la table 4.1.

TAB. 5.1: *Valeurs des dissimilarités d_h^o pour les objets de table 4.1*

	o_1	o_2	o_3	o_4	o_5	o_6
o_1	0	***0,38***	*0,32*	*0,3*	*0,33*	*0,51*
o_2		0	*0,23*	*0,48*	*0,65*	*0,9*
o_3			0	***0,31***	*0,48*	*0,73*
o_4				0	*0,17*	*0,42*
o_5					0	***0,25***
o_6						0

5.5 Écarts sur les relations

Le type τ est maintenant un concept k et les entités à comparer sont des objets dans k. La fonction d'écart se base sur le volume d'une généralisation des objets. Cette généralisation peut être calculée différemment selon l'état de structuration de la base. De plus, étant donnée une généralisation des fonctions de volume, $\mathcal{V}$ variées peuvent être définies sur cette généralisation.

5.5.1 Principes

La mesure d'écart s'applique sur les objets qu'elle peut «voir» de deux manières. D'une part, les objets ont toujours une structure d'ensembles de valeurs d'attributs. La structure est disponible à tout instant et on peut appuyer une fonction d'écart sur elle. Un tel procédé revient à explorer, lors de la comparaison d'un couple d'objets, la totalité des autres objets qui leur sont liés par des attributs relationnels. Cette manière de comparer les individus, dont une variante a été adoptée par KBG, peut être très coûteuse.

Dans la majorité des cas, c'est-à-dire lorsqu'il n'y a pas de co-définition entre concepts, on peut lui substituer une autre technique, consistant à s'appuyer sur les classes existantes sur le concept. Ainsi, la fonction d'écart se définira non plus sur les objets même, mais sur leur classes d'attachement. Autrement dit, on applique une autre vision sur les objets qui n'est plus de les considérer comme des structures d'attributs, mais comme des membres d'un ensemble au sein d'une famille d'ensembles (décrits par les classes sur le concept).

Cette double vision sur les objets a déjà été illustrée dans le chapitre 4. Elle a conduit à la définition de deux opérateurs de généralisation sur les objets, gen_o et GEN_o qui calculent respectivement la classe commune la plus spécifique dans une taxonomie existante et la classe *potentielle* la plus spécifique pour un ensemble d'objets.

En suivant l'opération de généralisation, nous définissons deux fonctions d'écart sur un concept. La première s'applique aux objets sous forme d'ensembles de couples attribut-valeur. Elle est toujours disponible pour une utilisation, mais plus coûteuse. La seconde fonction compare les objets par rapport à leurs classes d'attachement. Cette fonction n'est applicable que si une taxonomie de classes existe sur le concept.

Pour des raisons qui tiennent à la nature du processus de construction de classes, la seconde fonction, lorsqu'elle est disponible, est préférée à la première. En effet, comparer les objets par rapport à leurs classes permet surtout de gagner du temps, car au lieu d'un ensemble de couples d'entités (les valeurs des attributs), un seul couple (les classes) est traité. De plus, distinguer les objets selon leur classes communes conduit à une meilleure adéquation entre les deux étapes de la construction de classes, le regroupement et la caractérisation. Cela est dû au fait que lors de ce type de comparaison les groupes d'objet homogènes retrouvés auront la tendance de rassembler des objets dont les relations admettent une caractérisation compacte.

5.5.2 Écarts sur les descriptions des objets

La vision sur les objets en tant que structures d'attributs est la seule possible lorsque les objets comparés ne sont pas organisés en classes. Nous avons montre que ce cas se réalise en présence d'une co-définition entre concepts. Dans de telles conditions, l'opérateur $gen_o()$ n'est pas défini et donc la formule 5.6 n'est plus applicable. En effet, pour que $gen_o()$ puisse être appliqué, une taxonomie de classes doit être fourni. Or, il n'y a pas de telles classes, à l'exception de la classe commune de tous les objets, dans notre cas la racine du concept sous-jacent.

Dans ce cas, on substitue à la fonction δ_a^f non plus une fonction d'écart simple, mais une dissimilarité d'objets complète. En d'autres termes, on utilise pour la comparaison une fonction d^o calculée selon la formule 5.2. Cela revient à substituer un opérateur de généralisation sur les objet pris comme des structures, $GEN_o()$ à la place de la généralisation simple, $gen_o()$.

Une telle substitution est conforme aux principes de notre définition globale car elle remplace une fonction par une autre fonction tout en préservant les mêmes orientations dans leur définitions, à savoir l'évaluation du volume relatif d'une généralisation plus spécifique. Entre les deux cas, c'est l'ensemble des généralisations admissibles qui change.

Nous ne nous arrêtons pas sur la définition de la fonction d'écart car elle est identique à la fonction de dissimilarité d'objets pour le concept sous-jacent k, soit d_k^o. Son calcul est décomposé, de manière identique en un calcul de fonctions d'écart qui peuvent à leur tour être des dissimilarités d'objets. Il y subsiste naturellement le danger de retomber, après un certain nombre de développements à l'aide de dissimilarités d'objets sur la dissimilarité des objets initiaux. Cette dépendance récursive dans les descriptions des objets, ou la co-définition, peut donc provoquer une dépendance récursive entre valeurs de d^o. Une manière pour résoudre les problèmes liés à la circularité et de calculer une valeur cohérente pour chaque couple d'objets en co-définition est décrite au chapitre 7.

5.5.3 Écarts sur les classes des objets

Principes

Le type de connaissances véhiculées par la classe est complémentaire aux descriptions de ses instances : une classe décrit la structure commune et des contraintes sur les attributs caractérisant les objets. Tous les objets sont égaux en tant qu'instances de leur classe d'attachement. La classe exprime donc la similarité globale, par groupe, de toutes ses instances. Cette similarité peut être qualifiée de *conceptuelle* et c'est elle que l'on cherche à approximer par des mesures de ressemblances sur les descriptions des objets.

La similarité *conceptuelle* entre les instances d'un concept k à extension O à taxonomie $\mathcal{H}_k = \langle \mathcal{C}_k \leq_k^c \rangle$ peut être exprimée de manière directe par l'opérateur de généralisation $gen_o()$. En effet, celui-ci définit une fonction de $SIM_k : O \times O \rightarrow \mathcal{H}_k$ avec $SIM_k(o, o') = gen_o(o, o')$ qui à chaque couple d'objets fait correspondre leur classe commune la plus spécifique.

Le co-domaine de SIM_k est partiellement ordonné et donc les similarités de deux couples d'objets ne sont pas forcement comparables.

Notre objectif ici est de donner une expression quantitative, δ^{cl} à la ressemblance induite par $\mathcal{H}_k$ sur O. Cette fonction doit respecter l'ordre induit par la spécialisation. En d'autres termes, pour tous o_1, o_2 et o_3, si $gen_o(o_1, o_2) \leq_k gen_o(o_3, o_2)$ alors $\delta^{cl}(o_1, o_2) \leq \delta^{cl}(o_3, o_2)$ (voir les emplacements des trois objets sur la figure 5.1). Notons au passage, que la fonction recherchée n'est pas forcement « équivalente » à une fonction d_k^o prélevée sur les attributs. La forme d'équivalence

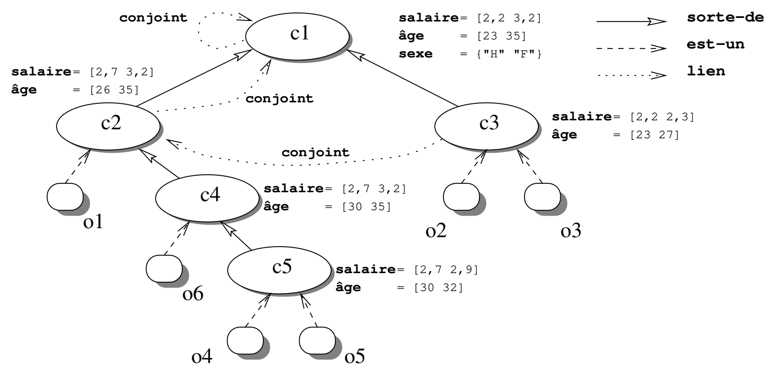


FIG. 5.1: *Un exemple de taxonomie de classes sur les objets de la table 4.1 du chapitre 4. Les classes sont données avec leurs descriptions, les objets sont liés à leurs classes d'attachement. Une proximité conceptuelle entre les objets se matérialise à travers la structure de la taxonomie : les objets dont les classes communes sont plus spécifiques sont plus « proches » que les autres.*

que nous entendons ici est une forme faible, elle n'exige pas l'égalité des valeurs des deux fonctions, mais juste l'égalité des ordres (l'isotonie des deux structures d'ordre total) que ces valeurs induisent sur les couples d'objets dans O^2.

Mesure sur les extensions des classes

La fonction $\mathcal{V}_r$ est mesurée sur les extension des classes. Elle reflète la cardinalité de cette extension par rapport à la cardinalité de la classe commune de tous les objets de la même famille.

Pour cela, nous définissons une fonction `cl-objets()` qui renvoie l'ensemble des instances de chaque classe passée en argument. `cl-objets` : $\mathcal{C}_k \rightarrow 2^{O_k}$ où $\mathcal{C}_k = conc - classes(k)$ et $O_k = conc - objets(k)$. De plus, la fonction `racine()` renvoi la classe racine du concept passé en argument. Sa signature est `racine` : $K \rightarrow C$.

$$\mathcal{V}_k^r(c) = \frac{||cl - objets(c)||}{||cl - objets(racine(k))||} \tag{5.15}$$

Nous adoptons la convention que la généralisation d'un objet est l'objet lui-même. Ainsi,

$$gen_o(\{o\}) = o$$

De plus, la fonction $\mathcal{V}_k^r$, lors qu'elle est appliquée à un objet unique, renvoie un résultat constant qui est,

$$\mathcal{V}^r k(o) = \frac{1}{||cl - objets(racine(k))||}$$

Compte tenu des remarques ci-dessus, la fonction d'écart se définit comme suit :

(5.16)

$$\delta_a^{cl}(o, o') = \frac{||cl - objets(gen_o(o, o'))||}{||cl - objets(racine(k))||} - \frac{1}{2} * (\frac{1}{||cl - objets(racine(k))||} + \frac{1}{||cl - objets(racine(k))||})$$

Avec une simplification des constantes, on obtient la formule suivante :

$$\delta_a^{cl}(o,o') = \frac{||cl - objets(gen_o(o,o'))|| - 1}{||cl - objets(racine(k))||} \tag{5.17}$$

Les valeurs de la fonction δ_k^{cl} sur l'ensemble des objets sont données dans la table 5.2.

TAB. 5.2: *L'écart δ_k^{cl} sur les objets de la figure 5.1*

	o_1	o_2	o_3	o_4	o_5	o_6
o_1	0					
o_2	1	0				
o_3	1	0,33	0			
o_4	0,67	1	1			
o_5	0,67	1	1	0,33	0	
o_6	0,67	1	1	0,5	0,5	0

Si la taxonomie $\mathcal{H}_k$ est un arbre, comme c'est le cas dans TROEPS, la fonction δ^{cl} est une *ultramétrique* (voir section 2.3 du chapitre 2).

Mesure sur les descriptions des classes

La composition de deux classes par rapport à leurs descriptions revient à définir une mesure de dissimilarité entre classes. Celle-ci est une généralisation de la mesure d^o : elle se decompose de la même manière en un ensemble d'écarts sur les attributs des classes dont les résultats sont à combiner d'une manière semblable.

La définition de la mesure d^c est décrite par la section 5.6.

5.5.4 Intégration des points de vue multiples

Cette manière de comparer les objets permet d'intégrer au modèle de la dissimilarité les multiples points de vue sur un ensemble d'objet. En effet, nous avons passés sous silence cet aspect : pour simplifier, un seul point de vue a été supposé donné. Toutefois, avec la définition de l'écart sur les classes δ^{cl}, on peur facilement imaginer comment des taxonomies multiples de classes peuvent être substituées à la taxonomie unique $\mathcal{H}_k$. Dans ce cas, un nouveau constructeur est introduit, le constructeur de *plurinomies*. Une plurinomie est une structure composée de plusieurs taxonomies alternatives sur le même ensemble d'objets. Les classes des différentes taxonomies peuvent être liées par des contraintes d'inclusion entre extensions, à l'instar des passerelles dans TROEPS.

Afin d'adapter notre mesure de dissimilarité au mode multi points de vue, il suffit de garder la fonction δ^{cl} pour l'extraction des ressemblances d'objets à partir d'une taxonomie unique et lui superposer une mesure construite, soit δ^{mpv} [Valtchev, 1997].

L'opérateur d'aggrégation des fonctions δ^{cl} sur chacune des taxonomies en une valeur unique pour δ^{mpv} est typiquement une combinaison linéaire normalisée (identique à celle de d^o) :

$$\delta_k^{mpv}(o,o') = \sum_{\mathcal{H}\in\mathcal{PV}_k} \lambda_{\mathcal{H}} \delta_{\mathcal{H}}^{cl}(o,o') \tag{5.18}$$

Ici, $\mathcal{PV}_k$ est l'ensemble des points de vue du concept k ($\mathcal{PV}_k$ = `cpt-points-de-vue` (k)) et $\lambda_{\mathcal{H}}$ sont les poids de chaque point de vue (éventuellement fournis par l'utilisateur). Cette extension sans un effort démesuré est un indice de la flexibilité du modèle.

5.5.5 Discussion

Alors que l'utilisation d'une fonction d'écart d'attributs relationnels basée sur les descriptions des objets référencés semble naturelle, l'introduction d'un écart que l'on mesure sur les classes d'attachement de ces objets semble moins intuitive. Nous réservons cette section à la discussion des effets d'une telle définition et de ses lacunes les plus importantes.

Lien entre écart de classes et dissimilarité d'objets

En toute généralité, il est possible d'avoir des objets o_1, o_2 et o_3, tels que $gen_o(o_1, o_2) \leq_k gen_o(o_2, o_3)$, mais pour chaque attribut a dans k $\delta_a^f(o_2, o_3) \leq \delta_a^f(o_1, o_2)$. Dans ce cas, aucune pondération des attributs dans une expression globale de la proximité d_k^o ne peut arriver à inverser l'ordre, c'est-à-dire, à ce que $d_k^o(o_1, o_2) \leq d_k^o(o_2, o_3)$. Un exemple d'une telle situation est présenté sur la figure 5.2 ci-dessous.

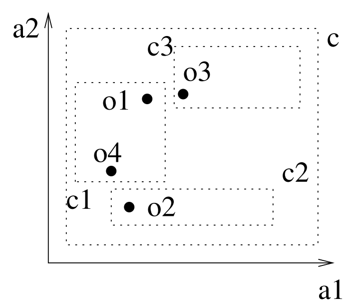


FIG. 5.2: *Exemple d'inversion entre d^o et δ^{cl} sur deux couples d'objets : la dissimilarité d'objets entre o_1 et o_3 est inférieure de celle de o_1 et o_4 ; en revanche, l'écart de classes entre o_1 et o_3 est supérieur à l'écart entre o_1 et o_4 car ces derniers partagent une classe, c_1.*

Les liens entre entre les structures partiellement ordonnées et les mesures qui y sont prélevées constituent un domaine qui a été exploré sous différents angles. Une étude très complète sur le sujet peut être trouvée dans [Montjardet, 1981].

Alternatives

La fonction δ^{cl} est une ultramétrique, elle offre donc une évaluation relativement grossière de la position mutuelle des objets dans la taxonomie. En particulier, elle ne reflète pas les étapes intermédiaires de la généralisation d'un couple d'objets o et o' vers $gen_o(\{o, o'\})$. Autrement dit, la valeur de δ^{cl} ne dépend pas du nombre des sous-classes (imbriquées) de $\mathcal{H}_k$ qui ne contiennent qu'un seul des deux objets.

> *Exemple* : Les valeurs de la fonction δ^{cl} sont identiques pour les couples (o_1, o_2) et (o_4, o_2) (le maximum de 1). Pourtant, entre les objets du premier couple, un seul pas de généralisation pour chaque objet suffit pour atteindre la classe commune c_1. En revanche, trois pas de généralisation sont nécessaires à partir de o_4.

Une alternative à δ^{cl} permettant de prendre en compte ce facteur est la mesure *topologique* δ^t [Valtchev et Euzenat, 1996]. Elle évalue la longueur des « branches » liant chacun des deux objets à la classe commune. De manière formelle δ^t est définie comme la longueur en nombre d'arrêts du plus court chemin liant les deux objets dans (la réduction transitive de) $\mathcal{H}_k$. Pour un couple o_1, o_2, avec $c = gen_o(o_1, o_2)$ elle est calculée comme suit :

$$\delta^t(o_1, o_2) = dist(o_1, c) + dist(o_2, c) \tag{5.19}$$

Ici $dist(o, c)$ est le nombre de classes qui contiennent l'objet o et sont sous-classes non strictes de la classe c. Dans le cas d'une dégénérescence, c'est-à-dire $o_1 = o_2$, nous considérons que $\delta^t(o_1, o_1) = 0$.

La fonction normalisée, $\overline{\delta}^t$, est calculée de manière triviale en divisant par la valeur maximale :

$$\delta^t_{max}(O) = \max_{o,o' \in O} \delta^t(o, o') \tag{5.20}$$

Les valeurs de la fonction δ^t_k pour l'ensemble des objets de la figure 5.1 sont données dans la table 5.3. Parmi ces valeurs, les valeurs qui distinguent la mesure δ^t_k de δ^{cl}_k sont mises en italique.

TAB. 5.3: *L'écart δ^t_k pour les objets de la figure 5.1*

	o_1	o_2	o_3	o_4	o_5	o_6
o_1	0					
o_2	1	0				
o_3	*0,67*	0,33	0			
o_4	0,67	1	1			
o_5	0,67	1	1	0,33	0	
o_6	*0,5*	*0,87*	*0,87*	0,5	0,5	0

À noter que la fonction δ^t n'est pas à l'abri de certaines inconsistences non plus, par exemple elle met au même niveau les couples (o_1, o_2) et (o_1, o_4) ce qui semble contre-intuitif car

$$gen_o(o_1, o_4) <^c_k gen_o(o_1, o_2).$$

En somme, aucune des deux fonctions ne rend parfaitement compte des ressemblances entre objets par rapport à leur position mutuelle dans la taxonomie $\mathcal{H}_k$.

Un compromis intéressant entre les deux semble être une fonction de distance de graphes δ^{t+} qui ne se limite pas à compter les nombres d'arcs parcourus ce que la distance *topologique* fait essentiellement. La mesure est définie sur un graphe aux arcs pondérés et effectue un cumul des poids au long d'un plus court chemin entre deux objets comparés. Finalement, pour réconcilier les deux fonctions δ^{cl} et δ^t, le poids d'un arc peut être fixé comme étant la cardinalité, de sa classe d'arrivée.

Cette approche a ses limites, car la similitude entre deux individus qui dépend fortement, nous l'avons montré, du contexte de leur comparaison n'est pas toujours liée à une catégorie commune reconnue. Un exemple de telle mesure, est la distance de Dayhoff [Dayhoff *et al.*, 1978] qui est utilisée pour la comparaison de séquences d'acides aminés[24].

24. voir également à ce propos section 6.6.2 du chapitre 6.

5.6 Dissimilarité entre classes

A l'instar de la mesure sur les objets d^o, une dissimilarité entre classes, soit d^{cl}, peut être définie. Celle-ci a sa place au sein d'une méthode de regroupement ascendant (voir chapitre 2). En effet, d^{cl} pourra remplacer la mesure sur les objets, à condition que les étapes de regroupement et de caractérisation puissent être alternés. Un tel remplacement permettra, lorsque cela est souhaitable, d'appuyer la fusuion de deux groupes d'objets davantage sur la similitude des descriptions que ces groupes admettent.

5.6.1 Fonction englobante

Les classes, tout comme les objets, peuvent être vues comme des tuples dont les éléments sont des types, c'est-à-dire des ensembles de valeurs. Ainsi, une manière directe de définir d^{cl} consiste à adapter l'expression de la fonction sur les objets aux descriptions de classes. Cela revient à spécifier un ensemble de fonctions d'écart sur les attributs, δ_a^f, ainsi qu'une fonction d^{cl} dont le rôle est de combiner les valeurs des écarts dans une quantité unique. Dans le cas d'une dissimilarité de type *Manhattan* pondéré, cette dernière fonction s'exprime comme suit :

$$d_k^{cl}(c, c') = \sum_{a \in A} \lambda_a * \delta_a^f(c.a, c'.a) \tag{5.21}$$

Ici A est l'ensemble des attributs du concept k. Afin d'aboutir à une mesure effective, les définitions des fonctions d'écart données dans la section précédente doivent être reformulées de telle sorte qu'elle puissent comparer des ensembles de valeurs et non des valeurs isolées.

5.6.2 Écart sur les attributs des classes

Avant d'aborder la question des écarts entre attributs de classses, nous donnons quelques définitions permmetant de fixer un cadre pour la comparaison de restrictions de valeurs.

Définitions supplémentaires

Afin d'étendre δ_a^f sur les expressions génériques, types et classes, nous avons besoin de deux nouveaux opérateurs sur ces expressions. Étant donné un type τ et sa taxonomie $\mathcal{H}_\tau = \langle \Upsilon_\tau \leq_\tau \rangle$ ces opérateurs permettent de retrouver les bornes inférieures et supérieures de deux éléments g et g' dans Υ_τ. Pour cela nous faisons l'hypothèse que $\mathcal{H}_\tau$ a la structure de *treillis* (voir [Davey et Priestley, 1992] pour une introduction en théorie des ordres et des treillis). Cette hypothèse n'est pas trop restrictive, elle est vérifiée sur plupart des types de données conventionnels (nombres, chaînes de caractères, etc.). En ce qui concerne les concepts, dans le cas d'*héritage simple*, la taxonomie a la forme d'un arbre qui est un *sup demi-treillis*. L'arbre est facilement complété en lui ajoutant un élément minimal qui est la classe vide, sous-classe de toute autre classe dans $\mathcal{H}_\tau$. Lorsque $\mathcal{H}_\tau$ admet un héritage multiple les propriétés de sup demi-treillis ne sont plus automatiquement garanties. Toutefois, l'utilité d'une structure de treillis pour la taxonomie des classes dans un langage à objets a été mise en évidence [Caseau, 1993]. Par ailleurs, toute une littérature est consacrée aux problèmes de la transformation d'une taxonomie de classes $\mathcal{H}_k$ quelconque en un treillis $\mathcal{TC}_k$ ([Missikoff et Scholl, 1989; Carpinetto et Romano, 1996; Godin *et al.*, 1995b; Dicky *et al.*, 1996; Valtchev, 1999]). Cela nous permet d'ignorer les cas où $\mathcal{H}_\tau$ n'est pas un treillis et ne peut pas être transformé en treillis de manière simple.

Nous définissons donc les deux opérateurs ci-dessous comme les bornes inf et sup dans $\mathcal{H}_\tau$:

- $\vee_\tau : \Upsilon_\tau \times \Upsilon_\tau \rightarrow \Upsilon_\tau$ associe à deux éléments dans Υ_τ leur *majorant* le plus spécifique
- $\wedge_\tau : \Upsilon_\tau \times \Upsilon_\tau \rightarrow \Upsilon_\tau$ associe à deux éléments dans Υ_τ leur *minorant* le plus général

En TROEPS, ces deux opérateurs sont disponibles via les primitives *INF* et *SUP* qui font partie des définitions des types de données (voir [Capponi, 1995], chapitre 3).

Écart sur les propriétés

La définition de la fonction δ_a^f est tirée de la formule 5.6 en substituant les restrictions de valeur $c.a$ aux valeurs $o.a$, ainsi qu'en remplaçant l'opérateur de généralisation de valeurs gen_τ par celui de borne sup dans le treillis des types :

$$\delta_a^f(c.a, c'.a) = \mathcal{V}_r(c.a \vee_\tau c'.a) - \frac{\mathcal{V}^r(c.a) + \mathcal{V}^r(c'.a)}{2} \tag{5.22}$$

Écart sur les relations

Pour continuer le parallèle avec les objets, nous considérons deux principes de comparaison pour les attributs relationnels. Les restrictions de valeur sur ces attributs sont des classes, soit $c_1 = c.a$ et $c_2 = c'.a$. Par conséquent, elle peuvent être comparées selon leur aspect intentionnel, mais aussi selon leur aspect extensionnel. Cela revient à appliquer à la comparaison de c_1 et c_2 une des deux visions évoquées au début de ce chapitre.

Par exemple, lorsque les classes sont vues comme des simples ensembles d'objets, la fonction de comparaison s'apparente à l'écart défini sur les propriétés. En outre, cette fonction reprend la définition de la fonction δ_a^{cl} de la section 5.5.3 (voir l'équation 5.16) avec des adaptations semblables à celles spécifées précédemment, :

(5.23)
$$\delta_a^f(c, c') = \frac{||cl-objets(c \vee_k c')||}{||cl-objets(racine(k))||} - \frac{1}{2} * (\frac{cl-objets(racine(c))}{||cl-objets(racine(k))||} + \frac{cl-objets(racine(c'))}{||cl-objets(racine(k))||})$$

Les classes peuvent égallement être prises comme des régions de l'espace de de description du concept sous-jacent. Dans ce cas, elle sont comparées sur la base de leur description, c'est-à-dire à l'aide d'une fonction de type d^{cl}. Ce principe entraîne une application récursive de la mesure d^{cl}, défini ci-dessus, au couple c_1, c_2. Le calcul de la mesure d^{cl} est identique au calcul de d^o. En particulier, les procédés de calcul récursif qui ont été évoqués précédemment et dont il sera question au chapitre 7 s'y appliquent pleinement.

5.7 Conclusion

Ce chapitre a présenté la définition d'une mesure de dissimilarité entre objets, d^o. Celle-ci s'appuie sur les opérateurs de généralisation définis au chapitre 4. La mesure est une généralisation des mesures numériques en provenance de l'analyse de données car elle admet des descriptions structurelles, mélangeant des informations sur les propriétés et sur les relations des individus. Lorsque la comparaison est limitée à un ensemble d'attributs monovalués, sa puissance est comparable à celle de la mesure de similarité utilisée dans KBG. Cependant, les fondements de notre mesure lui procurent des qualités appreciables que cette dernière n'a pas. En effet, la définition d'un principe universel de comparaison entre valeurs, basé sur une opération fondamentale qu'est la généralisation, permet une amalgame sensée entre écarts sur attributs de nature fort différente. De plus, l'*uniformité* des fonctions de comparaison assure une extensibilité sur des types de données nouveaux qui est bien utile dans le contexte d'un système de gestion de types ouvert. En outre, les choix de fonctions de comparaison permettent à l'utilisateur d'un module de regroupement de guider la construction de classes en introduisant des paramètres, par exemple les poids des attributs. Finalement, le principe de comparaison entre objets offre la possibilité d'inclure une partie de la connaissance disponible dans la base, les taxonomies de classes, dans le processus de construction de nouvelles classes.

Afin d'aboutir à une mesure universellement applicable sur un ensemble d'objets, il est nécessaire de considérer deux cas particuliers qui sortent du cadre restreint défini dans ce chapitre : la comparaison des collections et le calcul de la mesure d^o dans le cas de descriptions circulaires d'objets. C'est l'objet des deux chapitres suivants.

6

Dissimilarité entre collections

Le chapitre précédent traitait de la dissimilarité entre objets et nous avons vu que celle-ci se réduit naturellement à une dissimilarité entre valeurs d'attributs.

Nous avions fait l'hypothèse que tous les attributs, qu'ils modélisent des propriétés ou des relations, ont une nature fonctionnelle. Autrement dit, ceux-ci associent à un objet possesseur de la caractéristique modélisée au plus un élément du co-domaine de l'attribut. Dans ce chapitre, nous montrons comment une mesure de la même nature, et donc compatible, peut être définie pour les collections, aussi bien d'objets que de valeurs. Ainsi, nous étendons le champ d'application de notre mesure sur les attributs multivalués que nous appellerons également construits.

En début de chapitre, section 6.1, nous définissons le cadre de cette étude : la nature des collections et les difficultés que leur comparaison représente. Cette présentation est suivie d'un bref survol des méthodes existantes pour la comparaison des collections, section 6.2. Ensuite, section 6.3, nous décrivons une mesure de dissimilarité générique sur les collections, ainsi que ses instanciations sur les différents constructeurs (section 6.4). Celle-ci est examinée dans la section 6.5 pour en dégager quelques propriétés intéressantes. Finalement, les aspects algorithmiques du problème sont discutés dans la section 6.6.

6.1 Les collections en tant que données

Dans beaucoup de domaines, les relations entre éléments n'ont pas une nature fonctionnelle, c'est-à-dire qu'elles associent à un élément non pas un seul élément, mais un ensemble, ou plus généralement une collection, d'autres éléments. Par exemple, la relation de descendance entre une personne et ses enfants représente une telle association multiple.

6.1.1 Associations multiples entre entités

Dans le langage des bases de données relationnelles, ces associations sont définies comme étant de type $1 : N$ ou $M : N$ [Delobel *et al.*, 1991]. En termes d'objets, les associations multiples entre entités d'un domaine sont représentées moyennant des attributs dont la valeur est une collection. Ceci s'applique autant aux valeurs des types qu'aux objets.

Ainsi, dans le cas d'une modélisation de la synthèse organique, les éléments chimiques peuvent être représentés sous forme d'objets. Les valences possibles d'un élément pouvant varier, l'attribut sous-jacent aura comme valeur un ensemble d'entiers. Un autre exemple vient de la génétique, où un gène consiste en une séquence de nucléotides. Lors d'une modélisation par objets, cette séquence sera traduite par un attribut dont les valeurs sont des séquences des lettres qui représentent les nucléotides : A, C, G et T.

Les collections peuvent avoir une structure plus complexe, par exemple de liste ou de multi-ensemble. Dans les langages de programmation à objets, des structures de collections encore plus complexes peuvent apparaître : dictionnaires, tables, etc. Une référence en la matière est la bibliothèque standard `Collections` de Smalltalk [Goldberg et Robson, 1989] (voir [Cook, 1992] pour une taxonomie de ces structures). Toutefois, nous ne nous intéressons qu'aux collections les plus souvent utilisées, à savoir les ensembles, les listes et, dans une moindre mesure, les multi-ensembles.

6.1.2 Définition des collections

Un type de collection, τ est défini à partir d'un type de base τ_b à l'aide d'un constructeur de collections, `cons`. On dira que τ est construit sur τ_b. Le type de base peut, en toute généralité, être lui-même construit.

Les constructeurs définissent la structure de la collection. On en distinguera trois :

- le constructeur d'ensembles, `ensemble`, est le plus trivial, il n'admet pas d'ordre entre les membres, ni de répétition,
- le constructeur de listes, `liste`, est plus complexe, car il établit une collection totalement ordonnée avec des répétitions possibles,
- le constructeur de multi-ensembles, `m-ensemble`, admet des répétitions, mais pas d'ordre.

D'un point de vue mathématique, d'autres constructeurs valables peuvent aider à obtenir des graphes étiquetés, en particulier des DAG ou des arbres. En ce sens, les listes et les ensembles sont des cas extrêmes, les premières étant des ordres totaux et les seconds des ordres triviaux.

6.1.3 Quel intérêt de traiter les collections

On pourra s'interroger sur l'utilité d'une comparaison directe des collections. Dans un premier temps, il faut souligner l'abondance de domaines où des associations multiples se présentent naturellement : liens de filiation et de descendance, représentation de la structure des organisations, etc. Qui plus est, les attributs multivalués peuvent représenter la partie essentielle des caractéristiques des individus : par exemple une équipe sportive est composée par un ensemble de sportifs membres de l'équipe.

Par ailleurs, rien n'oblige un traitement direct des collections, on pourra facilement imaginer un passage vers des structures plus simples. Par exemple, dans [Cohen, 1996] un « codage » des collections de valeurs simples est suggéré qui transforme une collection en un ensemble d'attributs monovalués. Chaque attribut correspond à un élément du domaine de base, $dom(\tau_b)$ et indique sa présence/absence dans la collection encodée. Une technique similaire est utilisée pour traduire les collections d'individus dans [Faid *et al.*, 1999] au sein d'une approche par treillis de Galois. Cette décomposition des collections en leur constituants entraîne une perte d'information non-négligeable : les membres des collections ne sont plus considérés dans leur globalité, mais isolément. De cette manière, l'information de leurs éventuelles similitudes sont perdues. Quant à l'espace mémoire nécessaire, nous laissons au lecteur le soin de se persuader de l'inéfficacité de ces techniques (par exemple dans le cas de listes de nombres entiers).

Dans la suite de ce chapitre nous chercherons à démontrer qu'un traitement direct des collections à coût raisonnable est possible.

6.1.4 Problèmes liés à la comparaison de collections

Nous avons déjà montré quelles difficultés les collections posent pour la généralisation. En ce qui concerne l'établissement d'une dissimilarité, les difficultés sont liées à la nature des collections, mais aussi aux objectifs que nous avons fixés à la conception des outils de regroupement : la complétude, la cohérence et la composabilité.

Problèmes généraux avec les collections

Les collections sont composées par un ensemble d'éléments du type de base qui peuvent avoir une cardinalité variable. La comparaison de deux collections nécessite donc un arbitrage entre les différences sur ces deux aspects, composition et cardinalité :

1. **Différences de composition** Lorsque l'on compare des ensembles généraux, on peut utiliser la distance de *Hamming*. La distance de Hamming entre deux ensembles A et B d'un ensemble D est la cardinalité de leur différence symétrique. La valeur de la distance peut être facilement normalisée par la cardinalité de l'union ensembliste des deux arguments. Le résultat s'interprète comme le degré de communauté entre les deux ensembles.

   $$D_{hg}(A,B) = \frac{|A \triangle B|}{|A \cup B|}$$

   Cela revient à appliquer une dissimilarité égale à la fonction identité. Toutefois, dans beaucoup de cas, une telle comparaison n'est pas suffisamment précise car elle met tous les membres au même niveau, alors que certains peuvent être plus proches que d'autres. En particulier, il est très probable que deux collections d'objets aient une intersection vide. Ce n'est pas pour autant que leur dissimilarité est systématiquement zéro. On en déduit que les membres d'une collection doivent être comparés à l'aide d'une fonction plus fine que l'identité. Pour cela, nous supposons qu'une dissimilarité sur le domaine du type de base $D = dom(\tau)$ est fournie, soit δ_D. La définition de la mesure sur les collections va s'ancrer au-dessus de cette fonction tout en suivant la sémantique du constructeur sous-jacent.

2. **Taille variable** Un autre problème se pose lorsque l'on compare deux collections de tailles différentes. Ceci concerne plus particulièrement les listes, car lorsqu'elles sont comparées, une idée intuitive est d'établir une correspondance entre les éléments du même rang. Malheureusement, cela n'est pas une tâche triviale dans le cas où les listes ont un nombre de membres différent. La comparaison globale et directe entre collections est donc impossible.

Problèmes avec les collections dans les objets

Dans les formalismes logiques, les multiples associations peuvent être traitées comme des caractéristiques à part, c'est-à-dire en prenant chaque association comme un trait particulier. Cette technique est utilisée dans KBG, et plus récemment dans le contexte des treillis de Galois [Faid *et al.*, 1999].

Dans les objets, l'ensemble des associations est désigné par un seul attribut. Ainsi, lorsqu'on compare un couple d'objets par rapport à un attribut multivalué, les collections sous-jacentes doivent donner lieu à une valeur unique. Cette contrainte est imposée par le besoin d'assurer simultanément :

– l'équité entre attributs pour le calcul d'une fonction de dissimilarité,

– l'équité entre les objets comparés vis-à-vis d'un attribut multivalué.

Ce constat conduit naturellement à l'introduction d'une normalisation dans le calcul de la dissimilarité entre collections. En effet, le fait de ramener les valeurs de la dissimilarité entre collections dans [0 1], assure la même échelle pour les écarts monovalués et multivalués, leur mélange dans la dissimilarité d'objets d^o reste donc cohérent. La normalisation permet également une comparaison cohérente entre plusieurs objets par rapport à un attribut multivalué. Sans elle, la taille variable des collections serait le facteur prédominant dans une comparaison de collections. Finalement, la normalisation est de plus nécessaire dans TROEPS afin de préserver la composabilité sur l'ensemble des diverses fonctions de dissimilarité d'attributs.

6.2 Panorama de l'existant

Nous décrivons ci-dessous quelques techniques pour la comparaison de collections. La liste ne contient certainement pas une énumération exhaustive des méthodes existantes, elle se veut plutôt une synthèse des approches les plus représentatives.

6.2.1 Passage à des attributs monovalués

Une (non-)réponse aux problèmes liés aux collections est l'utilisation des techniques d'encodage évoquées plus haut. Ainsi, des attributs Booléens sont utilisés pour encoder la présence/absence d'un membre particulier du domaine de τ_b dans la collection [Cohen, 1996]. Cet aplatissement peut être effectué même en cas de collections d'objets. Une telle technique est décrite dans [Stepp et Michalski, 1986] et constitue le cœur de l'approche de l'algorithme CLUSTER/S pour le regroupement conceptuel sur des données structurées.

6.2.2 Résumés par des histogrammes

Une autre technique s'est développée dans le cadre de l'analyse de données symboliques. Bien que l'approche proposée (voir [Diday, 1998]) considère les collections comme des disjonction d'éléments et non comme des conjonctions, elle pourra facilement être généralisée pour ce dernier cas.

L'approche consiste à transformer une collection (les auteurs semblent se limiter ur des ensembles) en un histogramme. Pour cela, chaque objet symbolique de la collection est décomposé en ses couples attribut-valeur constituants. Le histogramme est ensuite composé en associant à chaque couple sa fréquence d'apparition sur l'ensemble des objets de la collection. Fondamentalement, un histogramme n'est autre que la donnée d'un ensemble de distributions de probabilités sur les valeurs, une par attribut rencontré dans les objets de la collection. il représente une vue compacte du ce qui est le « contenu » d'une collection et peut être utilisé à des fins de comparaison.

L'originalité de l'approche est dans l'utilisation à des fins de comparaison d'une structure qui en même temps fourni une caractérisation (un histogramme peut être défini sur plusieurs collections à la fois). Toutefois, cette caractérisation ne rend compte des ressemblances entre valeurs d'attributs au sein des collections que de manière globale, sans prêter attention aux combinaisons particulières dans chaque objet membre. Pour se convaincre, il suffit de regarder l'exemple ci-dessous.

Exemple Soient Trois collections chacune composée de deux individus, qui sont décrit par deux attributs, a et b. Les deux premières collections sont identiques et différentes de la troisième. Cependant, les techniques de d'histogrammes ne pourront distinguer c_3 des deux autres collections : l'histogramme produit sera identique dans tous les cas.

$c_1 = ((a : x, b : y), (a : t, b : z))$
$c_2 = ((a : x, b : y), (a : t, b : z))$
$c_3 = ((a : x, b : z), (a : t, b : y))$

Un autre inconvénient des histogrammes est l'absence d'un facteur qui reflète la différence de cardinalité entre collections. Finalement, ce type de comparaison ne peut être généralisé sur les autres collections, en particulier sur les listes, car elle ne tire aucun parti de l'ordre qui peut exister au sein des collections.

6.2.3 Collections et similarité

Nous sommes plus particulièrement intéressés par les méthodes de comparaison basées sur des mesures de dissimilarité. Dans ce type de méthodes, une mesure de dissimilarité entre membres des collections, c'est-à-dire éléments du domaine D de τ_b est fournie. Nous allons la noter δ_D. La dissimilarité entre collections est établie à partir des valeurs de celle-ci pour un ensemble de couples de membres.

Les mesures de lien

Nous avons vu au chapitre 2 que des indices d'agrégation peuvent être utilisés pour mesurer la distance entre deux groupes d'individus. Les groupes étant des ensembles, on pourra envisager employer une technique semblable, en la complétant éventuellement, pour nos propres objectifs. Bien évidemment, cette hypothèse ne concerne que les indices «simples», ne faisant pas apparaître des notions de poids ou de barycentre. Toutefois, on s'aperçoit facilement que les indices d'agrégation n'offrent pas les fonctionnalités que l'on désire. En particulier, les trois les plus fréquents : le lien simple, le lien complet et le lien moyen, ne vérifient pas la propriété d'identité.

La table ci-dessous fournit pour chacun de ces indices une indication des anomalies qu'il représente par un exemple de cette anomalie. Les collections utilisées sont des ensembles d'entiers dans l'intervalle [0 10] et la dissimilarité est la soustraction normalisée.

lien simple	$D_s(\{1,2\},\{1\}) = 0$
lien complet	$D_c(\{1,2\},\{1,2\}) \neq 0$
lien moyen	$D_m(\{1,2\},\{1,2\}) \neq 0$

Les exemples ci-dessus viennent montrer que la comparaison raisonnée de deux collections ne peut être basée sur la comparaison d'un seul couple de membres, ni sur la totalité de ces couples. Il faut donc faire une sélection plus astucieuse parmi les couples potentiels.

La distance de Hausdorff

Une autre méthode consisterait à utiliser la *distance de Hausdorff* [Hausdorff, 1962]. Cette métrique semble bien adaptée à la comparaison de nuages de points dans l'espace $\mathbb{R}^n$ [Huttenlocher *et al.*, 1993], elle permet par exemple de regrouper ensemble les zones d'images différentes représentant le même objet et, ce qui est important, avec des précisions variables.

Définition 6.1. *Étant donné un type T de domaine D muni d'une fonction de distance δ_D, la distance de Hausdorff entre deux ensembles A et B d'éléments de D est ε si pour tout élément de A il existe un élément dans B tel que la dissimilarité entre les deux éléments est au plus ε et vice versa :*

$$D_h(A, B) = \max_{a \in A}\{\min_{b \in B}\{\delta_D(a, b)\}\}.$$

Bien qu'elle donne une meilleure évaluation de la ressemblance globale entre deux ensembles, la distance de Hausdorff ne peut pas éviter certaines situations extrêmes dans lesquelles ses valeurs sont contre-intuitives. Ainsi, suite à la définition, la différence de cardinalité entre les deux ensembles ne joue aucun ou presqu'aucun rôle. Par exemple, avec un type de base entier, d'étendue $[0, 10]$ et une dissimilarité égale à la soustraction normalisée, les trois ensembles :

S_1	$\{1, 4, 7\}$
S_2	$\{2, 7\}$
S_3	$\{1, 3, 4, 5, 6, 7, 8, 9\}$

donnent les distances suivantes :

$d_h(S_1, S_2)$	$0,2$
$d_h(S_1, S_3)$	$0,2$
$d_h(S_2, S_3)$	$0,2$

Intuitivement, on s'attend à ce que la cardinalité des deux ensembles ait un plus fort impact sur la valeur de leur dissimilarité. En effet, l'ensemble S_1 est un sous-ensemble de S_3 qui le dépasse en cardinalité presque trois fois. Cependant, les deux ensembles ont la même dissimilarité par rapport à S_2 avec qui ils partagent le même membre unique, le 7. En d'autres termes, l'ajout d'un grand nombre d'éléments à S_1 n'a aucune incidence sur la dissimilarité. Cela est encore plus gênant quand le type de base est un type continu car il devient possible d'ajouter à un ensemble un nombre potentiellement infini d'éléments sans que sa dissimilarité avec un autre ensemble fixe n'en soit modifiée. Nous retenons cette anomalie que nous allons chercher à éviter dans la définition de notre propre mesure.

La similarité entre occurrences multiples dans Kbg

L'exemple ci-dessus vient pour montrer la nécessité de considérer les couples de membres des deux collections d'une manière *globale*. Il se dessine donc la notion d'appariement ou couplage. Une technique basée sur l'appariement est décrite par Bisson dans [Bisson, 1993].

Elle consiste à établir un appariement exclusif M_{app} entre les deux collections $z = \{e_1, e_2, \ldots, e_p\}$ et $z' = \{e'_1, e'_2, \ldots, e'_q\}$:

$$M_{app}(\{e_1, e_2, \ldots, e_r\}, \{e'_1, e'_2, \ldots, e'_q\}).$$

Le choix de l'appariement est fait en fonction de la similarité totale entre couples appartenant à cet appariement. En d'autres termes, on cherche à établir un appariement optimal pour la similarité. Cela permet de calculer la similarité comme la somme des similarités des couples dans l'appariement qui est normalisée de manière appropriée dans Kbg.

$$d(z, z') = \sum_{(e_i, e'_j) \in M_{app}} \delta_D(e_i, e'_j).$$

L'algorithme de calcul de l'appariement est heuristique, il applique une stratégie gloutonne pour la construction de l'appariement qui est très proche de l'algorithme de Kruskal pour la construction d'un arbre recouvrant de poids minimal dans un graphe. Ainsi, il consiste à choisir à chaque pas le couple de similarité optimale dont aucun des membres n'a encore été sélectionné. Une version simplifiée de cet algorithme est présentée ci-dessous.

L'algorithme est d'une complexité $O(n^3)$ où n est le nombre des objets dans le plus petit ensemble. En effet, l'enlèvement d'un couple (o_i, o'_j) d'une liste triée selon un critère indépendant

CALCUL-APPARIEMENT-SUBOPT(z,z',δ^m)
– données : $z = \{e_1, e_2, \ldots, e_p\}$, $z' = \{e'_1, e'_2, \ldots, e'_q\}$
– résultat : M_{app}
– variables : $Base$ - un ensemble de couples d'objets

$Base :=\{(o_i, o'_j)\}$ l'ensemble de tous les couples dans $z \times z'$
Trier $Base$ selon les valeurs de $\delta_D(o_i, o'_j)$.
tant que $Base$ non vide **faire**
  Choisir le couple (o, o') qui minimise δ_D
  Ajouter (o_i, o'_j) à l'appariement courant M_{app}
  Enlever de $Base$ tous les couples contenant o_i ou o'_j
`retourner` M_{app}

Algorithme 7: Calcul d'appariement sub-optimal entre ensembles

est en $O(n^2)$. Cette opération est effectuée à chaque pas de la complétion de M_{app}, dont la cardinalité est égale à celle du plus petit ensemble.

Le résultat obtenu est sub-optimal et il peut être arbitrairement loin de l'optimum M_{opt}. Cela n'est pas gênant outre mesure lorsqu'il s'agit d'un simple calcul pour la comparaison de deux collections. En revanche, dans le contexte d'un calcul recursif de dissimilarité comme celui de Kbg, l'absence de garantie d'optimalité peut conduire à des anomalies (voir section 7.4)

6.2.4 Comparaison par généralisation

Une approche originale vers le traitement direct des collections (de type ensemble et séquence ou liste) est proposée dans [Ketterlin, 1995]. Celle-ci n'est pas basée sur le calcul d'une dissimilarité, nous la présentons néanmoins ici pour sa proximité avec le principe global de comparaison entre attributs, à savoir la prise en compte de la généralisation.

L'auteur définit une stratégie de comparaison entre collections d'objets, basée sur l'existence d'une arborescence de classes sur le domaine dont les membres des collections sont issus (le type de base). Le but est d'obtenir la (une en cas de listes) généralisation la plus spécifique de deux collections eut égard à la hiérarchie de classes. La généralisation se présente sous forme d'une collection de classes (concepts dans le langage du domaine) qui vérifie une triple condition d'optimalité :

1. chaque membre de chaque collection appartient à au moins une classe de la généralisation,

2. chaque classe inclut au moins un membre de chaque collection,

3. les classes de l'ensemble sont minimales pour les deux premières conditions.

La notion de degré global de spécificité d'une collection de classes est utilisée dans le choix des généralisations. Ce degré est calculé à partir des indices de spécificité des classes. Il indique, de manière indirecte, le niveau de ressemblance entre les collections généralisées.

L'approche par généralisation a l'avantage de l'élégance et de l'efficacité. Mais elle recelle des défauts inadmissibles pour nos propres objectifs :

– l'approche est difficilement applicable sur des collections de valeurs simples ; en effet, la richesse de la structure classificatoire sur un domaine d'entiers ou réels rend son application très coûteuse ;

– puisque la généralisation recherchée n'est pas nécessairement la plus spécifique possible, nous ne voyons pas comment celle-ci pourra se transformer en une valeur de dissimilarité de telle sorte que la propriété d'identité soit vérifiée par la mesure sous-jacente ;

– finalement, en tant qu'indice des ressemblances entre collections, la généralisation telle qu'elle est utilisée ne rend pas compte des différences de taille entre les collections ; par exemple il est imaginable que la comparaison de 2 ensembles, A de taille 2 et B de taille 3 rend le même résultat que celle de A avec C de taille 5.

La dernière remarque établit le lien de l'approche par généralisation et la mesure de Hausdorff dont elle partage la satisfaction d'une double condition min/max.

En somme, l'exemple de ce paragraphe nous permet d'affirmer que contrairement au cas des attributs monovalués, la généralisation n'est pas un bon candidat pour la dissimilarité entre collections.

6.2.5 Synthèse

Les techniques examinées ci-dessus proviennent de diverses disciplines : l'analyse de données, l'apprentissage automatique, la reconnaissance des formes, l'extraction de connaissances à partir de données. Elles sont toutes adaptées à une sorte particulière de collections, le plus souvent aux ensembles, où à un type de base particulier (nombres, objets structurés). De plus, le compromis entre les deux aspects des collections, à savoir le contenu et la cardinalité n'est pas géré d'une manière satisfaisante : en général, un des aspects est privilégié. Cela fait d'elles de mauvaises candidates pour la comparaison des collections dans un contexte d'objets. Seule la technique de KBG se distingue par un souci de généricité. Aucune des mesures ne peut être généralisée sur des collections de second ordre (c'est-à-dire où les membres sont des collections) et ordres supérieurs.

Notre but pour la suite est de donner la définition d'une mesure de dissimilarité entre collections δ^m qui est parfaitement générique sur les types de base des collections et sur leurs constructeurs. En même temps, nous cherchons la normalisation dans la définition de la mesure car la nature des comparaisons entre objets ne permet pas une normalisation *a posteriori*, c'est-à-dire par rapport à une valeur maximale sur la totalité des collections. Un objectif de première importance est le bon équilibre entre contenu et cardinalité dans la valeur de la mesure.

6.3 Une mesure générique de dissimilarité entre collections

Notre but ici est de définir une méthode générique pour mesurer les ressemblances entre collections. Cela veut dire que les ressemblances vont dépendre du constructeur et être paramétrables par le type de base.

6.3.1 Principes

L'idée principale de la mesure de ressemblance entre deux collections est d'évaluer l'effort nécessaire pour passer d'une collection à l'autre. Ceci implique une comparaison deux à deux entre les membres des collections, c'est-à-dire un appariement entre les collections. Le principe de l'effort minimal est semblable à celui appliqué dans le cas d'attributs monovalués à la différence près que nous ne cherchons pas la généralisation, mais le passage direct.

6.3.2 Appariement

Une comparaison directe entre collections n'est pas possible à cause de leur taille variable. Il faut donc comparer les membres des collections deux à deux et établir une valeur unique et globale.

Appariements variés

La manière immédiate de comparer deux collections est de composer tous les couples possibles entre membres. La proximité pour les collections δ^m serait une composition des valeurs de δ_D pour chaque couple, ce qui revient à utiliser une mesure de lien moyen entre collections. Toutefois, comme nous l'avons vu plus haut, le lien moyen ne produit pas une fonction intéressante car celle-ci viole la propriété d'identité. Il faut donc se limiter à un sous-ensemble, de couples dans le calcul, ce qui signifie prendre un appariement entre collections.

Appariement non-exclusif La distance de Hausdorff est basée sur le calcul, implicite, d'un appariement non-exclusif. La différence des cardinalités entre collections est quasiment ignorée.

Appariement non-minimal L'appariement proposé par Bisson est un exemple d'appariement exclusif non-optimal[25]. La valeur peut être très loin de la valeur optimale.

Appariement exclusif et minimal

Afin d'assurer la minimalité, il faudra pouvoir choisir certains couples parmi tous les possibles. Dans le cas précédent, c'est-à-dire avec deux collections identiques, il est facile à voir que l'ensemble des couples pris en compte doit se limiter aux seuls couples de membres identiques. Ainsi, avec l'exemple précédent, $\delta^m(m_1, m_1)$ dépendra seulement de $\delta_D(2,2)$ et de $\delta_D(5,5)$.

Cette procédure de sélection s'exprime par deux conditions qui seront généralisables à des collections non-identiques.

1. D'abord, chaque membre d'une collection participe exactement **une seule fois** dans un couple.
2. Ensuite, **tous** les éléments participent à un couple.

Dans le cas général, les deux collections étant de taille différente, la première condition peut toujours être respectée. En revanche, la seconde n'est plus applicable. Toutefois, on pourra exiger que **tous** les éléments de **la plus petite** collection participent à un couple. En d'autres termes, un nombre maximal de couples va être retenu, tout en respectant la première condition, ce qui signifie que le nombre de couples sera égal à la cardinalité de la plus petite collection.

Cette manière de procéder construit un appariement M entre les collections vues comme des structures de graphes dont les membres sont des sommets. Signalons que pour l'instant nous ne spécifions pas le constructeur sous-jacent et donc les arêtes des deux graphes ne sont pas prises en compte.

D'après ce qui a été dit, cet appariement est *exclusif* (première condition) et *maximal* en cardinalité (deuxième condition). Il existe un grand nombre d'appariements M qui satisfont les deux conditions. Pour lier l'appariement à la proximité, nous allons de plus exiger que M soit optimal pour la valeur de δ^m. On cherche donc un appariement M_{opt}. Dans le cas d'une

25. maximale dans son cas car la fonction recherchée est une similarité.

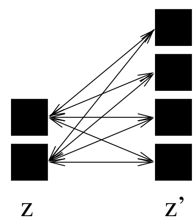


FIG. 6.1: *Appariement entre deux collections vues comme les deux composantes d'un graphe biparti. Les dissimilarités entre membres sont les poids des arcs dans le graphe.*

dissimilarité, ceci revient à exiger une minimalité, c'est-à-dire que la valeur de δ^m calculée sur les couples dans M_{opt} doit être minimale.

En poussant plus loin la modélisation des deux collections en termes de graphes, nous pouvons traduire le problème de la recherche de M_{opt} dans un cadre de graphe biparti complet $G = (X, Y, E)$. Les deux parties du graphe représentent les deux collections et chaque sommet x de X est lié à tous les sommets y de Y par une arête dont le poids est égal à $\delta_D(x, y)$. Ainsi, M_{opt} correspond à un couplage maximal de poids minimal dans G. De plus, M_{opt} répond à des contraintes liées au constructeur des collections, `cons`.

Éléments fictifs

Soient z et z' deux collections sur un type τ muni d'une fonction de dissimilarité δ_D. Soient encore les tailles de z et z', p et q respectivement et soit $q \leq p$.

Les éléments non appariés par M_{opt} dans z peuvent tout de même s'intégrer dans le calcul de la dissimilarité entre collections. À noter que les éléments non appariés, s'ils existent, appartiennent exclusivement à la collection de taille supérieure. On pourra donc les considérer comme appariés à des éléments fictifs que l'on rajoute à la collection de taille inférieure. La dissimilarité est calculée sur un appariement *total* entre la collection z et une nouvelle collection $\bar{z} = \{e'_1, e'_2, ..., e'_q, f'_1, f'_2, ..., f'_{p-q}\}$ (voir la figure 6.2).

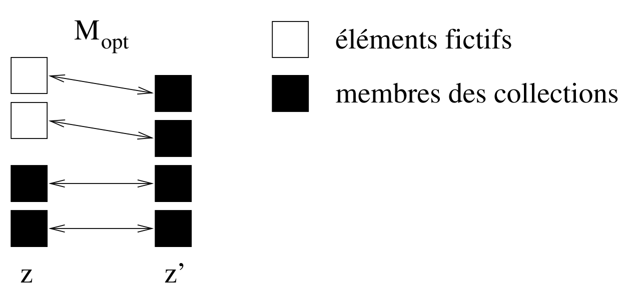


FIG. 6.2: *Complétion d'une collection par des éléments fictifs*

Afin de prendre en compte le statut particulier de ces éléments « en trop », on fixe la valeur de la fonction δ_D sur un couple comportant au moins un élément fictif à 1. En particulier, pour tout élément e_i de z si f_j est un élément fictif, alors $\delta_D(e_i, f_j) = 1$. En conséquence, la dissimilarité entre collections peut s'exprimer en tant que la moyenne des valeurs de δ_D sur les couples d'un appariement total.

Cependant, le calcul peut se limiter aux couples ne comportant pas d'éléments fictifs car la contribution des autres couples peut être calculée auparavant, elle est égale à leur nombre. Les principes du calcul effectif sont présentés dans la suite.

6.3.3 Formule générique

Soit $M_{opt}(z, z') = \{(e_i, e'_j)\}$ un appariement optimal de z et z'. La dissimilarité des deux collections par rapport à $M_{opt}(z, z')$ est calculée sur la base de la **somme** des dissimilarités des couples dans M_{opt}. La somme est **rectifiée** par la somme des dissimilarités sur les couples ayant un élément fictif. Cette contribution est exactement le nombre des membres non appariés dans la plus grande des deux collections. Finalement, la somme totale des dissimilarités est **normalisée** en divisant par la cardinalité de la plus grande collection.

Si au moins une des deux collections comparées est non-vide, alors leur dissimilarité est établie comme le rapport entre la somme des dissimilarités sur un appariement exclusif, de cardinalité maximale et de dissimilarité totale minimum d'une part et la cardinalité de la plus grande collection, d'autre part.

$$\delta^m(z, z', \delta_D) = \frac{\sum_{(e_i, e'_j) \in M_{opt}(z,z')} \delta_D(e_i, e'_j) + |p - q|}{\max(p, q)} \tag{6.1}$$

Si les deux collections sont vides, alors leur dissimilarité est nulle :

$$\delta^m(\varnothing, \varnothing, \delta_D) = 0 \tag{6.2}$$

Dans la suite, les formes concrètes que la fonction δ^m prend sur chaque constructeur sont examinées.

6.4 Instanciations selon les constructeurs

Ayant défini une fonction générique, nous nous intéressons maintenant à ses instanciations en fonction des deux paramètres, le constructeur et le type de base qui détermine la fonction δ_D. Dans la suite de cette section, la préférence est accordée aux constructeurs car les diverses possibilités pour les fonctions sur les membres ont été examinées au chapitre précédent.

6.4.1 Exemple de base

Avant de passer à la suite, nous présentons une fonction δ_D sur un ensemble de six objets. Les valeurs de la fonction ont été calculées sur les descriptions des objets présentés au chapitre 2 (voir table 2.1).

6.4.2 Ensembles

Les ensembles de valeurs ou d'objets sont des collections non-ordonnées de valeurs ou d'objets sans répétition. Soient $s = \{e_1, e_2, ...e_p\}$ et $s' = \{e'_1, e'_2, ..., e'_q\}$ deux ensembles définis sur un type de base D, ADT ou concept. Soit aussi δ_D une proximité normalisée sur le type D. Ainsi, $\delta_D : D \times D \to [0, 1]$.

TAB. 6.1: *Valeurs des dissimilarités δ_D pour les objets de la table 2.1*

	o_1	o_2	o_3	o_4	o_5	o_6
o_1	0					
o_2	0,38	0				
o_3	0,36	0,26	0			
o_4	0,28	0,43	0,31	0		
o_5	0,5	0,61	0,46	0,33	0	
o_6	0,56	0,74	0,57	0,44	0,25	0

Appariement

Le cas des ensembles est le plus simple car le constructeur `ensemble` n'introduit pas de contraintes sur la forme du couplage à calculer. Ainsi, trouver M_{opt} revient à construire un couplage de poids minimal dans le graphe G comme décrit ci-dessus. Le nombre de toutes les solutions valides du problème du couplage est de V_p^q, le nombre des variances sans répétitions de p éléments pris parmi q.

La recherche de couplage de poids minimal dans un graphe biparti est un problème connu en Recherche Opérationnelle sous le nom d'*affectation linéaire* [26] (voir [Hillier et Lieberman, 1995]). Les variantes de l'algorithme de calcul d'un appariement sont discutées dans section 6.6.1.

Exemples

Prenons l'exemple de deux ensembles $s = \{2, 7, 4, 3, 9\}$ et $s' = \{6, 4, 8, 1\}$ sur un type de base l'intervalle d'entiers $D = [0, 10]$. Soit aussi δ_D la fonction de soustraction normalisée sur D, c'est-à-dire que

$$\delta_D(a, b) = \frac{abs(a - b)}{10}.$$

L'appariement optimal entre s et s' est $M_{opt}(s, s') = \{(2, 1), (7, 6), (4, 4), (9, 8)\}$. Tout couple dans M_{opt} a une dissimilarité de $0,1$ à l'exception de $(4, 4)$ qui vaut 0. Par conséquent, la dissimilarité globale devient :

$$\delta^m(s, s', \delta_D) = \frac{(\delta_D(2, 1) + \delta_D(7, 6) + \delta_D(4, 4) + \delta_D(9, 8)) + 1}{5} = (0,3 + 1)/5 = 0,26.$$

Soient les deux ensembles d'objets, $s = \{o_1, o_6\}$ et $s' = \{o_2, o_4, o_3\}$ avec, comme fonction de dissimilarité de base, la fonction définie au début de cette section (table 6.1).

L'appariement optimal entre les deux ensembles est $M_{lopt}(s, s') = \{(o_1, o_3), (o_6, o_4)\}$. Le « poids » de l'appariement M_{lopt} est $0,8$, alors que la dissimilarité entre ensembles est $\delta^m(s, s', \delta_D) = (0,8 + 1)/3 = 0,6$.

6.4.3 Listes

Les listes se caractérisent par la présence d'un ordre entre leurs éléments qui est indépendant de la nature de ces éléments. De plus, contrairement aux ensembles, des répétitions sont autorisées dans les listes. La condition de respect de l'ordre dans les listes lors d'un appariement peut être formulée comme suit :

$$\forall (e_i, e'_j), (e_m, e'_n) \in M_{opt}(l, l') \ : \ i \leq m \ \Rightarrow \ j' \leq n'.$$

26. *Square assignment* en anglais.

Contrairement au cas d'ensembles où tous les couples sont potentiellement candidats à l'appariement, la condition ci-dessus invalide certains couples d'office. Par exemple, si $q > 1$, alors $(e_1, e'_q) \notin M_{opt}$. D'autres couples, bien que *a priori* valables, ne peuvent être simultanément dans M_{opt}. Tel est le cas de (e_1, e'_2) et (e_2, e'_1) dont la présence de l'un invalide celle de l'autre.

Le problème de la recherche de l'appariement M_{opt} sur deux listes quelconques l et l' peut être caractérisé comme suit. D'abord, supposons que l' est la liste la plus courte, c'est-à-dire que $q < p$. En effet, le cas $q = p$ est trivial. Il convient de représenter les deux listes l et l' par deux graphes orientés G et G', respectivement. Chaque graphe contient comme sommets les membres de la liste sous-jacente et les arêtes lient deux sommets représentant des éléments consécutifs dans la liste. Ainsi, par exemple, le premier graphe est défini comme suit :

$$G = (l, E = \{\langle e_i, e_{i+1} \rangle\}_{i=1,p-1}).$$

Les deux graphes G et G' sont des chaînes et peuvent être considérés comme les réductions transitives de deux ordres totaux. Les appariements que nous cherchons peuvent être vus comme des applications de l dans l' qui sont injectives et préservent l'ordre dans les listes. Par conséquent, l'espace des solutions pour le problème est composé par tous les homomorphismes d'ordre injectifs de G sur G'. Un calcul simple permet d'établir le nombre de ces solutions à C_q^p, c'est-à-dire le nombre des combinaisons de q éléments pris parmi p. Ce nombre, dans le cas pire, est une fonction exponentielle de p (et donc de p aussi).

Alignement de séquences

Tel qu'il a été formulé, notre problème représente un cas particulier d'un problème connu de la bioinformatique, *l'alignement de séquences* [27] [Gusfield, 1997].

Un alignement de séquences est une correspondance entre leurs membres qui préserve l'ordre et autorise des « trous », c'est-à-dire des éléments non-appariés dans chacune des séquences. Le but est, globalement, de maximiser le nombre de correspondances entre éléments identiques ou proches. Les « trous » sont nécessaires afin d'assurer de meilleures correspondances entre membres se situant à des places d'éloignement variable. Chaque appariement de deux membres a un coût qui exprime la proximité des éléments appariés. Les trous ont aussi un coût qui est en général supérieur au coût d'un appariement quelconque. L'alignement entre séquences de qualité maximale est celui qui minimise le coût global. Il est supposé refléter les processus d'évolution (mutations telles les substitutions, les insertions et les omissions) qui ont déterminé la divergence de deux protéines liées à un ancêtre commun.

Différences

La différence entre les deux problèmes est qu'en alignement de séquences il n'y a pas d'obligation de cardinalité maximale de l'appariement. En d'autres termes, le coût minimal de la transformation d'une séquence vers l'autre prime sur la « saturation » d'au moins l'une d'elles par l'appariement. Dans notre contexte, pour prendre l'exemple des deux listes l et l', cela voudrait dire qu'il pourrait y avoir des éléments non-appariés même dans la liste la plus courte, l. La présence de la contrainte supplémentaire de « saturation » que nous imposons ne change rien à la nature du problème. Il peut encore être résolu par *programmation dynamique* en modifiant l'algorithme de base pour l'adapter à la situation particulière. Nous discutons des différents algorithmes de calcul de l'appariement dans Section 6.6.

27. *Sequence Alignment* en angl.

Exemples

Soient $l = \langle 5,3,6 \rangle$ et $l' = \langle 2,4,6,2,7 \rangle$ deux listes sur un type d'entiers à domaine $D = [2,12]$. Le domaine D est muni de la même fonction δ_D, la soustraction normalisée. L'appariement optimal entre l et l' est $M_{opt}(l,l') = \{(5,6),(3,2),(6,7)\}$ avec une dissimilarité totale de $0,3$. La valeur définitive pour les deux listes est donc $\delta^m(l,l') = (3/10+2)/5 = 0,46$.

Soient les deux listes d'objets, $l = \langle o_1, o_6 \rangle$ et $l' = \langle o_2, o_4, o_3 \rangle$ où les objets sont ceux donnés par la table 6.1. L'appariement optimal entre l et l' compte tenu des valeurs de δ_D est $M_{opt}(l,l') = \{(o_1,o_2),(o_6,o_4)\}$ dont la dissimilarité totale est $0,82$. Alors que la dissimilarité des listes est $\delta^m(l,l') = (0,82+1)/3 = 0,606$.

Extension de la formule générique

On pourra s'interroger sur la nécessité d'avoir un appariement total entre listes. En effet, dans certains cas, notamment dans le domaine de la biologie moléculaire avec l'alignement de séquences, des appariements partiels sont non seulement possibles, mais exigés.

Il est facile de généraliser la fonction δ^m en une fonction sur tout appariement, partiel ou total. Pour cela, il suffit de remplacer le numérateur dans l'expression 6.1 par la somme totale des dissimilarités sur les couples appariés plus le nombre total des éléments non-appariés dans chacune des listes. Afin d'avoir une fonction normalisée, le dénominateur sera à remplacer par le nombre total des termes dans le numérateur, c'est-à-dire la somme des longueurs des deux listes moins la cardinalité de l'appariement.

La nouvelle fonction devient :

$$\delta^m_{part}(l,l',\delta_D) = \frac{\sum_{(e_i,e'_j)\in M^p_{opt}(l,l')} \delta_D(e_i,e'_j) + p + q - 2 * |M^p_{opt}(l,l')|}{p+q-|M^p_{opt}(l,l')|}.$$

Cette seconde fonction est plus appropriée lorsque l'on veut appuyer la comparaison sur l'existence d'une sous-liste commune. Toutefois, deux arguments parlent contre son utilisation. Tout d'abord, notre but est d'évaluer les ressemblances globales entre deux listes, l'établissement d'une sous-liste appariée n'est qu'un effet secondaire. De plus, la mesure δ^m_{part} rend le calcul itératif comme celui décrit dans le chapitre 7 extrêmement complexe. Du fait de la variabilité du dénominateur en fonction de l'appariement, elle ne permet pas une séparation préalable de la dissimilarité entre deux objets co-définis (voir chapitre 7 section 7.4). Pour ces raisons, on ne la cite ici qu'à titre de comparaison.

Saturation et optimalité

Il faut remarquer que le fait d'exiger une saturation de la plus courte liste n'est pas anodin. En effet, contrairement au cas des ensembles, la saturation n'est pas intrinsèquement liée à l'optimalité. En d'autres termes, et même si cela peut sembler paradoxal, l'abandon de cette contrainte pourrait résulter en un appariement de plus faible coût global. Ci-dessous un exemple de situation où l'on gagne en acceptant un appariement sub-optimal en cardinalité.

> *Exemple* Soient deux listes d'entiers $l = \langle 1,10,2,8 \rangle$ et $l' = \langle 5,1,10,2 \rangle$ sur le type de base $D = [0,10]$ muni comme précédemment de la même fonction de proximité δ_D. Avec un appariement optimal et de cardinalité maximale, $M_{opt} = \{(1,5),(10,1),(2,10),(8,2)\}$, la valeur de la proximité totale est de $\delta^m(l,l') \approx 0,6$, alors que si on se limite à un appariement de coût minimal, c'est-à-dire sub-optimal, $M_{sub-opt} = \{(1,1),(10,10),(2,2)\}$, on obtiendra $\delta^m(l,l',\delta_D) = 0,4$.

6.5 Quelques propriétés de la mesure

Dans cette section nous allons nous intéresser au comportement de la nouvelle mesure car elle ne saurait être utilisée sans qu'un minimum de «bonnes» propriétés ne soit assuré. Ainsi, nous démontrons que δ^m représente un indice valide de dissimilarité ou semi-métrique. Ensuite, un examen du comportement de la mesure sur les collections lorsque celles-ci varient faiblement est proposé (propriété de *monotonie*). Finalement, les conditions sous lesquelles la mesure est une métrique sont étudiées.

6.5.1 Semi-métrique

Pour que la mesure admette une utilisation adéquate en regroupement, les propriétés de semi-métrique sont exigées. Rappelons que δ^m est une fonction composée sur une fonction de base, δ_D. Pour cette raison, dans l'examen des propriétés de notre mesure, nous sommes obligés de contraindre δ_D.

Proposition 6.1. *Si la fonction δ_D est une semi-métrique, alors δ^m l'est aussi.*

Cependant, les propriétés de semi-métrique ne représentent pas de contraintes très fortes. Elles sont communément vérifiées par les écarts standard utilisés en analyse de données et en apprentissage automatique. De plus, elles sont transmissibles à travers toutes les constructions que nous définissons dans ce mémoire.

6.5.2 Monotonie

La propriété de *monotonie* peut être vue comme la généralisation de la même propriété sur les stuctures ordonnées. Rappelons qu'une fonction f liant deux structures ordonnées $\mathcal{U} = \langle X_\mathcal{U} \leq_\mathcal{U} \rangle$ et $\mathcal{V} = \langle X_\mathcal{V} \leq_\mathcal{V} \rangle$:

$$f : \mathcal{U} \to \mathcal{V}$$

est monotone si elle préserve l'ordre entre éléments de son domaine lorsqu'ils sont plongés dans le co-domaine :

$$u \leq_\mathcal{U} u' \ ssi \ f(u) \leq_\mathcal{V} f(u').$$

Dans notre cas, la définition d'un ordre sur la structure du domaine de la fonction, qui est soit l'ensemble de tous les ensembles sur un type de base τ, 2^τ soit l'ensemble des listes, Λ^τ, est déjà un problème en soi. *A priori*, les deux ensembles sont ordonnés par les relations d'inclusion d'ensembles ou de listes (sous-liste-de, notée $\Subset$), une définition de $\leq_\mathcal{U}$ ne devrait donc pas poser de problème. Cependant, notre mesure cherche un équilibre entre les similitudes des membres des collections et les cardinalités des collections, alors que les deux relations d'ordre privilégient la cardinalité. Ce biais se constitue indirectement, en ignorant complètement les éventuelles ressemblances entre membres. En effet, le calcul d'une relation d'inclusion peut être comparé au calcul d'une dissimilarité de type δ^m à l'aide d'une fonction triviale, $\delta_D = 1 - 1_D$ (l'inverse de l'identité sur D).

On peut s'attendre donc à un désaccord entre la monotonie «standard» et notre mesure dans certains cas limites. Malgré ce désaccord, nous avons décidé de tester le comportement de δ^m vis-à-vis de $\subseteq$ et $\Subset$ car les résultats de l'étude peuvent suggérer une nouvelle notion d'ordre sur 2^τ ou Λ^τ qui est mieux adaptée à nos objectifs.

Dans la suite, les opérations émentaires à la base des deux relations sont examinées : ajout d'un élément à une des deux collections. L'examen du comportement de δ^m dans ces cas est suffisant pour déduire une image complète sur la totalité de chaque relation. Nous en distinguons deux cas : lorsque le nouvel élément est ajouté à la collection de plus petite taille et lorsqu'il est ajouté à la plus grande des deux. Dans un contexte où la cardinalité est le seul critère, on peut s'attendre à ce que la dissimilarité diminue (n'augmente pas) dans le premier cas et qu'elle augmente (ne diminue pas) dans le second. Nous allons voir que cela n'est pas généralement vérifié, la constitution des collections joue un rôle important. Finalement, rappelons que les cas d'enlèvement d'un élément sont symétriques et ne seront pas traités ici.

Ajout d'un élément à la plus petite collection

Ensembles Lorsque l'on augmente la plus petite des deux collections, la dissimilarité entre la plus longue collection et la nouvelle collection obtenue devrait intuitivement décroître. Cela n'est vérifié que pour les ensembles.

Proposition 6.2. *La valeur de la fonction δ^m décroît lorsque l'on ajoute un élément au plus petit ensemble.*

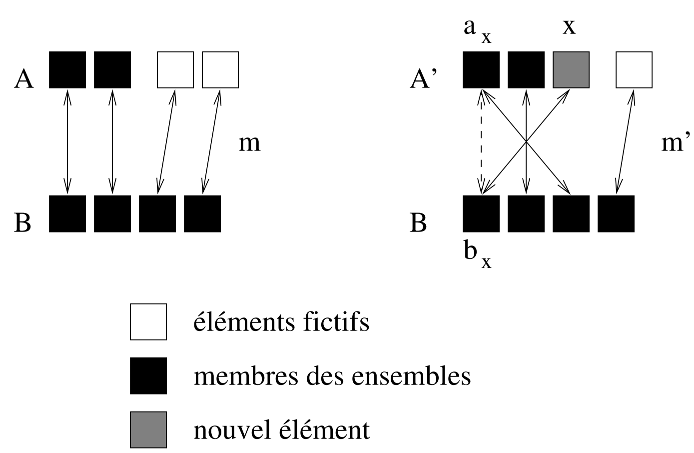


FIG. 6.3: *Ajout d'un élément à la plus petite collection*

Listes Lorsque l'on augmente la plus courte des deux listes, la dissimilarité entre la liste la plus longue et la nouvelle liste obtenue ne décroît pas nécessairement. Ainsi, l'élément 10 est inséré dans la liste L_2.

L_1	$\langle 5, 0, 1 \rangle$
L_2	$\langle 0, 1 \rangle$
L'	$\langle 0, 10, 1 \rangle$

Les trois dissimilarités à considérer sont données dans le tableau ci-dessous. La valeur de $\delta^m_{part}(L_1, L_2, -_D)$ n'est pas donnée car elle coïncide avec $\delta^m(L_1, L_2, -_D)$ (calculée sur le même appariement $\{(0,0),(1,1)\}$).

$\delta^m(L_1, L_2, -_D)$	$0,33$
$\delta^m(L_1, L', -_D)$	$0,5$
$\delta^m_{part}(L_1, L', -_D)$	$0,5$

Signalons que le meilleur appariement total entre L_1 et L' est le seul appariement total possible :

$$\{(5,0),(0,10),(1,1)\}.$$

En revanche, le meilleur appariement partiel entre L_1 et L' est :

$$\{(0,0),(1,1)\}.$$

On observe donc que l'ajout de 10 a conduit à une augmentation de la dissimilarité :

$$\delta^m(L_1, L_2, \delta_D) \;<\; \delta^m(L_1, L', \delta_D) \;=\; \delta^m_{part}(L_1, L', \delta_D).$$

Cela interdit d'affirmer la *monotonie* des fonctions δ^m et δ^m_{part} dans ce premier cas d'ajout.

Ajout d'un élément à la plus grande collection

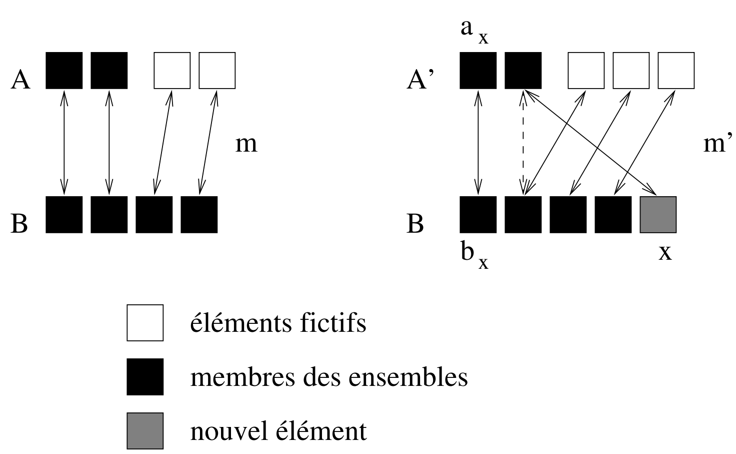


FIG. 6.4: *Ajout d'un élément à la plus grande collection*

Ensembles La fonction δ^m ne vérifie pas cette deuxième variante de la *monotonie*. Pour s'en persuader, un contre-exemple est fourni ci-dessous. Trois ensembles, S_1, S_2 et S' sont constitués sur un type de base entier composé par les entiers entre 0 et 10, c'est-à-dire $D = [0, 10]$. La fonction de dissimilarité associée, δ_D, est égale à la soustraction normalisée.

S_1	$\{10, 9, 8\}$
S_2	$\{1, 2\}$
S'	$\{10, 9, 8, 3\}$

Lorsque l'on augmente le plus grand des deux ensembles, ici S_1, la dissimilarité entre le couple d'ensembles résultant ne croît pas, mais diminue : de $0,8$ elle descend à $0,75$.

$\delta^m(S_1, S_2, -_D)$	$0,8$
$\delta^m(S', S_2, -_D)$	$0,75$

Listes Lorsque l'on augmente la plus longue des deux listes, la dissimilarité entre la liste la plus courte et la nouvelle liste obtenue n'augmente pas nécessairement. Par exemple, observons l'effet de l'ajout de l'élément 10 à la liste L_1.

L_1	$\langle 5, 0, 0 \rangle$
L_2	$\langle 0, 10 \rangle$
L'	$\langle 5, 0, 0, 10 \rangle$

Les trois dissimilarités à considérer sont données dans le tableau ci-dessous. Les valeurs de δ^m_{part} sur les deux couples ne sont pas données, elles coïncident avec les valeurs respectives de δ^m (car calculées sur les mêmes appariements).

$\delta^m(L_1, L_2, -_D)$	$0,66$
$\delta^m(L_1, L', -_D)$	$0,5$

On observe donc que l'ajout de 10 a conduit à une diminution de la dissimilarité :

$$\delta^m(L_1, L_2, \delta_D) \ > \ \delta^m(L_1, L', \delta_D).$$

De manière identique au cas précédent, cela ne permet pas d'affirmer la *monotonie* des fonctions δ^m et δ^m_{part} dans ce second cas d'ajout.

6.5.3 Métricité

Le fait qu'une mesure est une métrique ou ne l'est pas a son importance en analyse de données, en particulier pour la construction de groupes d'individus de forte similitude. Comme nous l'avons souligné dans la section 2.3 du chapitre 2, une métrique assure une certaine compacité entre les valeurs au sein d'un ensemble d'individus : les valeurs très élevées et très faibles ne peuvent cohabiter sur les mêmes objets que sous certaines contraintes fortes. De cette manière une métrique, lorsqu'elle est utilisée dans un algorithme de regroupement contribue à la construction de groupes qui sont plus homogènes[28].

Pour cette raison, lorsqu'une nouvelle mesure est définie, elle est soumise à l'examen de ses propriétés formelles et plus particulièrement de sa métricité.

Dans cette section nous présentons une brève étude de la question concernant la mesure δ^m.

Tout d'abord, signalons que la mesure δ^m est une mesure « construite » à partir d'une autre fonction δ_D. En conséquence, les propriétés de δ^m dépendent de celles de δ_D. Malheureusement, le mode de calcul que nous avons fixé pour δ^m afin d'assurer des propriétés intéressantes, ne permet pas de transposer les propriétés de mesure de base sur la mesure construite.

Ensembles

Dans le cas des ensembles, la métricité est directement transmissible de δ_D vers δ^m.

Proposition 6.3. *Soit un domaine D muni d'une mesure de dissimilarité δ_D. Si δ_D est une métrique, alors la fonction sur les ensembles d'éléments dans D, δ^m, est également une métrique.*

La nouvelle mesure vérifie les propriétés d'une métrique lorsque la dissimilarité de base, δ_D les vérifie. Ici il faut noter que les métriques sont très fréquentes dans notre contexte. Ainsi, la totalité des écarts définis dans le chapitre 5 sont des métriques. De plus, la propriété est transmissible par le biais de l'opérateur que nous avons retenu pour d^o, la combinaison linéaire.

28. sans qu'il soit nécessaire de prendre en compte la totalité des dissimilarités au sein d'un groupe.

Listes

La métricité n'est pas vérifiée en cas général par la nouvelle mesure. En d'autres termes, il est possible d'exhiber un exemple de trois listes L_1, L_2 et L_3 et la mesure correspondante, δ_D, telle que :

$$\delta^m(L_1, L_2, \delta_D) \ > \ \delta^m(L_1, L_3, \delta_D) \ + \ \delta^m(L_2, L_3, \delta_D).$$

Pour cela, il suffit de prendre le domaine binaire, $D = \{0, 1\}$, et de considérer toutes les séquences de bits. La dissimilarité δ_D peut être prise en étant la fonction `XOR`. Dans ces circonstances, pour les trois listes ci-dessous :

L_1	10
L_2	01
L_3	010

l'inégalité triangulaire n'est pas vérifiée. En effet, les trois valeurs de la dissimilarité de listes sont respectivement :

$\delta^m(L_1, L_2, \mathbf{XOR})$	1
$\delta^m(L_1, L_3, \mathbf{XOR})$	$0,33$
$\delta^m(L_2, L_3, \mathbf{XOR})$	$0,33$

À noter que cet exemple n'est plus un contre-exemple si l'on considère des appariements non maximaux en cardinalité. Sans que cela soit tenu pour preuve, nos expérimentations ont montré que la métricité de cette seconde fonction est beaucoup plus difficile à mettre en défaut. En effet, si l'on considère la fonction $\delta^m_{part}(L_1, L_2, \mathbf{XOR})$

$\delta^m_{part}(L_1, L_2, \mathbf{XOR})$	$0,6(6)$
$\delta^m_{part}(L_1, L_3, \mathbf{XOR})$	$0,3(3)$
$\delta^m_{part}(L_2, L_3, \mathbf{XOR})$	$0,3(3)$

Dans ce cas, l'égalité seulement est atteinte :

$$\delta^m_{part}(L_1, L_2, \delta_D) \ = \ \delta^m_{part}(L_1, L_3, \delta_D) \ + \ \delta^m_{part}(L_2, L_3, \delta_D).$$

Cela ne prouve pas l'absence de metricité.

6.5.4 Discussion

Nous venons de donner une liste de résultats négatifs concernant le comportement de la nouvelle mesure sur les collections de type liste. Rappelons que pour cela, nous avons fixé a priori un ensemble de propriétés dont certaines de nature intuitive, notamment la monotonie. Il s'avère que ces propriétés sont vérifiées sur les collections de type ensemble, mais ne le sont pas, pour la plupart, sur les listes. Cela ne conduit en aucun cas à un rejet de notre mesure. D'une part, les efforts nécessaires pour mettre en défaut certaines propriétés ont mis en évidence l'influence de l'ordre dans l'ajout des nouveaux éléments. En d'autres termes, les résultats ont suggéré l'existence de «bonnes» et «mauvaises» places pour l'ajout, c'est-à-dire celles qui, lorsque l'élément y est placé, ne conduisent pas à un renversement dans l'ordre intuitif des dissimilarités, et celles qui, au contraire, résultent en des dissimilarités contre-intuitives.

D'autre part, dans la définition de notre mesure nous avons incorporé des principes qui assurent des «bonnes» caractéristiques (dépendance de la cardinalité et des différences inter-membre) que d'autres méthodes de comparaison des listes n'ont pas.

Nous pensons, en conséquence, qu'une étude plus approfondie est nécessaire afin d'établir le ou les comportements désirables d'une mesure de dissimilarité de listes. Celle-ci doit très probablement prendre en compte des particularités des traitements des listes qui varient selon le domaine d'application. Soulignons que cela n'était pas l'objectif dans le cadre du travail présenté ici.

6.6 Mise en œuvre

Dans la suite, nous présentons deux techniques algorithmiques pour le calcul des appariements optimaux recherchés, un pour chaque constructeur.

6.6.1 Ensembles

L'affectation linéaire

Pour résoudre le problème de l'affectation linéaire [Papadimitriou et Steiglitz, 1998] des algorithmes polynômiaux existent basés sur la *méthode hongroise* [Kuhn, 1955]. Un algorithme de complexité d'ordre de $O(n^3)$, n étant la taille de la collection, a été décrit dans [Ahuja *et al.*, 1993]. Des solutions alternatives sont apparues depuis, dont une récente (voir [Ji *et al.*, 1997]), de complexité en temps plus faible, mais d'une complexité d'espace de l'ordre de $O(n^2)$.

Calcul d'un appariement

La description détaillée de la méthode n'est pas notre but ici. Cependant, nous pouvons donner les grands traits de l'algorithme (voir algorithme 8). Il effectue un traitement en deux étapes. La première consiste en une recherche d'une solution sub-optimale (par un appel à CALCUL-APPARIEMENT-SUBOPT()). Pendant la seconde étape, cette solution est améliorée de manière itérative. L'algorithme s'arrête lorsqu'aucune amélioration n'est possible. Nous passons sous silence le calcul d'amélioration qui tient d'un raisonnement graphique (recherche des zéros d'une table) et invitons le lecteur à consulter [Gondran et Minoux, 1984] pour une bonne présentation.

CALCUL-APPARIEMENT-OPT(s,s',δ^m)
– données : $s = \{e_1, e_2, \dots, e_p\}$, $s' = \{e'_1, e'_2, \dots, e'_q\}$ ensembles
– résultat : M_{opt} un ensemble de couples
– variable : M_{app} un ensemble de couples

$M_{opt} := \varnothing$
$M_{app} :=$ CALCUL-APPARIEMENT-SUBOPT(z,z',δ^m)
tant que $M_{opt} \neq M_{app}$ **faire**
    $M_{opt} := M_{app}$
    $M_{app} :=$ AMELIORER-APPARIEMENT(s,s',δ^m,M_{app})
`retourner` M_{opt}

Algorithme 8: Calcul d'appariement entre ensembles

Reprenons l'exemple de la section 6.4.2. Les données sont deux ensembles d'objets, $S = \{o_1, o_6\}$ et $S' = \{o_2, o_4, o_3\}$. La fonction δ_D est donnée par la table 6.1.

Avec une procédure de calcul approximative à la KBG, on obtient $M_{app}(s, s') = \{(o_1, o_4), (o_6, o_3)\}$, avec une dissimilarité totale de $0,85$. Quant à l'algorithme exact, il donne un appariement optimal $M_{lopt}(s,s') = \{(o_1,o_3),(o_6,o_4)\}$ dont la dissimilarité totale est $0,8$. Alors que la dissimilarité des listes est $\delta^m(s,s',\delta_D) = (0,8+1)/3 = 0,6$.

6.6.2 Listes

L'alignement de séquences

Le calcul de couplage de poids minimal entre deux listes s'apparente au problème connu comme l'alignement de séquences. L'alignement est une technique utilisée en bioinformatique pour la comparaison de séquences (de nucléotides ou d'acides aminés).

La méthode de calcul d'un alignement de score maximal entre deux séquences $l = \langle e_1, e_2, ...e_q\rangle$ et $l' = \langle e'_1, e'_2, ...e'_p\rangle$ introduite par Needleman et Wunsch [Needleman et Wunsch, 1970] consiste à construire une matrice A de dimension $q \times p$. Ici, $a_{i,j}$ représente le score maximal pour un alignement (non nécessairement de cardinalité maximale) entre les sous-listes $l[1,i] = \langle e_1, e_2, ...e_i\rangle$ et $l'[1,j] = \langle e_1, e_2, ...e_j\rangle$. La valeur de $A_{i,j}$ est obtenue en considérant quelques autres éléments dans la matrice dont les indices sont proches de i ou j. En effet, il suffit de remarquer qu'un alignement optimal finissant par e_i et e'_j vérifie exactement une des conditions suivantes :

1. il inclut (e_i, e'_j) et donc son coût dépend du coût de l'alignement finissant en $i-1$ et $j-1$ qui est dans $a_{i-1,j-1}$, $a_{i,j} = a_{i-1,j} + cout - appariement(e_i, e'_j)$
2. il n'apparie pas e_i (celui-ci est « effacé ») et donc $a_{i,j} = a_{i-1,j} + penalite(e_i)$
3. il n'apparie pas e_j et donc $a_{i,j} = a_{i,j-1} + penalite(e'_j)$

La fonction $cout - app$ est une évaluation de la proximité entre les deux éléments appariés. Dans le cas d'alignement de séquences d'acides aminés c'est une mesure de similarité donnée par la matrice de Dayhoff [Dayhoff *et al.*, 1978]. La fonction *penalite* a une valeur négative qui n'est pas fixée auparavant, mais dépend de la place de son argument. La valeur de $a_{i,j}$ est donc calculée comme le maximum de ces trois possibilités $a_{i,j} = max(a_{i-1,j} + cout - app(e_i, e'_j), a_{i,j-1} + penalite(e'_j), a_{i-1,j} + penalite(e_i))$.

Finalement, l'algorithme calcule le score de l'alignement plutôt que l'alignement lui-même. Dans notre cas, ce n'est pas gênant car c'est le poids de l'appariement qui nous intéresse plus que l'appariement.

Calcul d'appariement

Dans notre cas, le problème devra être reformulé pour s'adapter au contexte spécifique de comparaison de listes dans les objets. Les principales modifications sont :

1. Calcul d'un appariement de **coût minimal** plutôt que maximal. Pour cela, on substitue δ_D à $cout - app$ et on fixe la pénalité d'un élément non-apparié à 1. De plus, lors d'une sélection entre plusieurs possibilités, on choisit systématiquement celle qui conduit à un coût minimal.
2. Calcul d'un appariement de cardinalité maximale. Il n'est pas possible de satisfaire cet objectif sans modifications dans la structure du contrôle de l'algorithme. Ainsi, la saturation de la liste la plus courte est assurée en s'interdisant d'effacer ces éléments. Par conséquent, il ne reste que deux cas à examiner lors du traitement d'un couple (e_i, e'_j) : *ajout* à l'appariement ou *omission* de e'_j.

CALCUL-APPARIEMENT-OPT-L(l,l',δ^m)
– données : $l = \rangle e_1, e_2, \ldots, e_q\rangle$, $l' = \langle e'_1, e'_2, \ldots, e'_p\rangle$ ensembles
– résultat : $cout$ un réel
– variable : A une matrice carrée de réels

{============= *positionnement de la 1ère colonne dans A* =============}
INIT(A)

{============= *parcours de A* =============}
pour $i := 1$ à q **faire**
  pour $j := i$ à p **faire**
    $A[i,j] := \min(A[i-1, j-1] + \delta_D(e_i, e'_j), A[i, j-1] + 1)$
`retourner` $A[p,q]$

Algorithme 9: Calcul d'appariement optimal entre listes

Par ailleurs, des algorithmes sub-optimaux, utilisant le même principe glouton que l'algorithme 7, peuvent être définis sur les listes aussi. La différence en est que pour assurer un appariement de taille maximale, certains couplages entre membres doivent être interdits.

Exemple Soient $l = \langle 5, 3, 6\rangle$ et $l' = \langle 2, 4, 6, 2, 7\rangle$ deux listes sur un type à domaine entier $D = [2, 12]$. L'algorithme heuristique retournera un appariement $M = \{(5, 2), (3, 4), (6, 6)\}$ dont la dissimilarité totale est $0,4$. L'appariement optimal, trouvé par l'algorithme d'alignement modifié est $M_{opt} = \{(5, 6), (3, 2), (6, 7)\}$ avec une dissimilarité de $0,3$. Soient encore les deux listes d'objets, $l = \langle o_1, o_6\rangle$ et $l' = \langle o_2, o_4, o_3\rangle$ où les objets sont ceux donnés par la table 6.1. Avec une procédure de calcul approximative à la KBG, on obtient $M_{app}(l, l') = \{(o_1, o_4), (o_6, o_3)\}$, avec une dissimilarité totale de $0,85$. Quant à l'algorithme exact, il donne un appariement optimal $M_{opt}(l, l') = \{(o_1, o_2), (o_6, o_4)\}$ dont la dissimilarité totale est $0,82$. Alors que la dissimilarité des listes est $\delta^m(l, l') = (0,82 + 1)/3 = 0,606$.

6.6.3 Complexité

Soit n la taille de la collection la plus grande.

Ensembles La complexité dans le cas pire de la méthode Hongroise est de $O(n^3)$.

Listes La complexité de l'algorithme d'alignement modifié est de $O(n^2)$.

6.6.4 Discussion

Dans les divers cas d'appariement entre collections nous avons présenté des algorithmes exacts et des algorithmes heuristiques pour le calcul d'un appariement minimal en dissimilarité. Les premiers ont l'avantage de la précision, sans pour autant rajouter un coût supplémentaire au traitement.

La grossièreté des solutions produites par les algorithmes est, en partie, contrebalancée par leur côté intuitif. En effet, au moins dans le cas des ensembles, il est plus facile d'expliquer à un utilisateur comment la valeur de la fonction δ^m est obtenue avec un tel algorithme qu'avec la procédure exacte.

6.7 Conclusion

L'obtention d'une mesure de dissimilarité valide sur les collections nécessite un appariement entre les membres des collections. Pour cela, on introduit une fonction calculant l'appariement de dissimilarité totale minimale. La fonction est déterminée par le constructeur des collections comparées. Dans le cas général, il est exigé que cet appariement soit de taille maximale, c'est-à-dire que la collection la plus petite soit saturée. Alors qu'avec les ensembles cela semble naturel, avec les listes il est possible que certains appariements non-maximaux pour la cardinalité donnent des résultats meilleurs. La mesure de dissimilarité est générique sur les constructeurs (liste, ensemble, etc.) et sur le type de base qui peut être un type construit à son tour. Les mesures de cette famille ont un comportement intuitif, elles sont des semi-métriques dans la majorité des cas et la fonction sur les ensembles est une métrique sous certaines conditions. Elles sont toutes calculables par des algorithmes bien connus et/ou simples.

Compte tenu du contexte plus large de la construction de classes sur les objets, nous pouvons rapporter la satisfaction de plusieurs objectifs fixés au chapitre 3 :

– **Complétude** : la mesure est complète sur les types de données et sur les constructeurs, son application dépend de la donnée d'un opérateur d'appariement et d'une mesure de dissimilarité sur les membres,

– **Extensibilité** : la mesure peut être facilement étendue sur de nouveaux types d'éléments et, ce qui est plus intéressant, sur de nouveaux types de constructeurs,

– **Cohérente** : le résultat d'une comparaison arbitre entre les différences sur les constituants et sur la cardinalité entre deux collections.

7

Dissimilarité et circularité dans les objets

Introduction

Dans le chapitre 3 nous avons vu que les concepts peuvent se trouver en co-définition ainsi que les difficultés que ces co-définitions engendrent pour la construction des classes. Le chapitre a également présenté une stratégie et une méthode qui permettent de construire des classes sur des concepts co-définis. Les deux outils de la méthode, les opérations de généralisation et la fonction de dissimilarité, ont été introduits dans les chapitres 4 et 5 respectivement.

Dans ce chapitre nous nous attaquons à un problème difficile qui est le traitement direct des co-définitions entre objets. En effet, nous avons vu que le calcul de la dissimilarité d^o se base sur une décomposition récursive d'un couple d'objets en ces attributs. Dans les cas de co-définition, cette décomposition peut ne pas aboutir à un ensemble de calculs d'écarts entre valeurs simples, mais «revenir» sur le couple de départ. En d'autres termes, la valeur de la dissimilarité sur un couple dépend récursivement d'elle-même et un calcul direct n'est pas envisageable.

Dans la suite, une technique de calcul pour la dissimilarité sur des objets co-définis est présentée. Elle s'inspire d'une part des travaux sur la sémantique des descriptions circulaires (appelées *circuits terminilogiques*) menés au sein de la communauté des logiques de description [Nebel, 1990; Nebel, 1991; De Giacomo et Lenzerini, 1997] et d'autre part de la méthode de Kbg de comparaison de descriptions relationnelles.

Au début du chapitre nous discutons de l'impact de la circularité sur la construction de classes section 7.1.1 avec un exemple de situation où une prise en compte de la circularité peut être bénéfique. La section suivante (section 7.2) propose un cadre formel pour la co-définition entre objets ainsi que la définition de quelques structures dans la base de connaissances utiles pour le traitement de la co-définition. Ces structures sont mises en œuvre lors du calcul effectif de la mesure à l'aide d'un système d'équations dont la composition et le principe de la résolution sont exposés dans la section 7.3. L'existence d'une solution du système (section 7.3.2) et les propriétés des fonctions calculées (section 7.3.4) sont abordées. Les dificultés particulières liées à la co-définition par des attributs multivalués sont étudiés dans la section 7.4. Un exemple complet de calcul et d'utilisation de la mesure est donné dans la section 7.5 qui met en évidence les avantages de la prise en compte des relations, y compris les relations circulaires, lors d'un regroupement. Le chapitre se termine par la présentation d'un algorithme qui réalise la technique décrite auparavant (section 7.6).

7.1 Impact de la circularité sur la construction de classes

Cette première partie du chapitre est réservée à la discussion des effets de la circularité sur le processus de construction automatique de classes. Elle aborde également la question de l'utilité d'un traitement direct des descriptions circulaires.

Intuitivement, la circularité dans les descriptions des objets est un cas particulier de la circularité entre concepts. Toutefois, la circularité entre objets est source de problèmes qui sont plus délicats à traiter.

Dans la suite nous allons employer les deux termes circularité et co-définition dans des circonstances très semblables. Toutefois, par «circularité» nous allons désigner un phénomène qui n'est pas propre aux formalismes à objets, alors que« co-définition » aura un sens beaucoup plus précis, celui d'une relation entre unités de représentation : concepts (voir chapitre 3), classes et objets.

7.1.1 Difficultés du traitement circulaire

Dans le chapitre 3 nous avons vu que la co-définition entre concepts ne permet pas une construction aisée de classes par des méthodes de regroupement conceptuel classiques. Le problème de taille avec les descriptions circulaires est qu'elles rendent difficile la caractérisation des classes.

En effet, l'opérateur de généralisation sur un ensemble d'objets $GEN_o()$ défini dans le chapitre 4, se décompose naturellement sur les attributs des objets. Pour chaque attribut, la caractérisation consiste à prendre un élément parmi un ensemble d'éléments autorisés, Υ_τ, ensemble qui est propre à l'attribut. En conséquence, les ensembles Υ_τ des généralisations autorisées doivent être connus au moment de la caractérisation.

Avec des relations circulaires entre concepts, il n'est pas possible de fonder une construction de classes sur des critères qui nécessitent les descriptions de ces classes. Cela reviendrait à exiger l'existence de l'ensemble des classes $\mathcal{C}_k$ sur un concept k avant que le processus de leur construction soit lancé, ce qui est absurde.

Pour remédier à ce problème, nous avons adopté une méthode de construction de classes qui consiste à former les classes en extension par un processus de regroupement selon proximités et puis de les caractériser. Cette séparation est devenue possible grâce à l'utilisation des outils de construction, dont l'application n'est pas perturbée par l'existence des liens croisés entre concepts, en particulier une mesure de dissimilarité entre objets d^o.

On pourrait présenter cette approche ainsi : l'utilisation d'une mesure de dissimilarité permet de contourner les problèmes de la co-définition entre familles d'objets et classes en limitant le «raisonnement» lié à la comparaison sur les couples d'objets et non plus sur des ensembles entiers de tels objets comme le font, entre autre, les critères de qualité basés sur les descriptions. Ce changement de focale du plus large au plus restreint, entraîne une prépondérance des informations locales qui n'est que partiellement compensée par l'application de méthodes de regroupement selon des critères globaux. Toutefois, la perte d'une vision globale sur l'ensemble des objets et de là, une détérioration possible des résultats du regroupement, sont largement compensées par la possibilité de fournir un résultat dans des cas où les méthodes «globales» échouent.

Cependant, si d^o arrive à facilement contourner les problèmes de la co-définition entre concepts en limitant la comparaison aux couples d'objets, il n'en va pas de même pour la co-définition entre objets. La co-définition d'objets est très semblable à la co-définition entre concepts, elle est introduite par des suites d'attributs. Une définition formelle de cette relation est proposée dans la section 7.2, ici nous allons remarquer seulement que la co-définition suppose l'existence d'un circuit de relations liant un, deux ou plusieurs objets. Par exemple, deux objets représentant un couple d'époux sont en co-définition par le biais de l'attribut `conjoint` qui forme un circuit de longueur deux.

Des tels circuits peuvent empêcher le calcul direct de la dissimilarité entre objets. En effet, d'une manière semblable à la généralisation, la dissimilarité se décompose sur les attributs des objets comparés (un couple). En l'absence de circularité entre objets, cette décomposition conduit tôt ou tard à un ensemble de comparaisons d'entités, objets ou valeurs de types de données, à l'aide de fonctions d'écart qui sont directement calculables. Le résultat de la comparaison des objets de départ est donc progressivement recomposé à partir de ces résultats partiels. Toutefois, si des circuits existent dans la description des objets, il est posible, au bout d'un certain nombre de décompositions, de revenir au couple de départ.

En clair, cela veut dire que la valeur de la fonction d^o sur ce couple dépend récursivement d'elle-même et que, par conséquent, un calcul direct de cette valeur est inconcevable. Cela se produit lorsqu'on compare deux personnes par rapport à un ensemble d'attributs dont le `conjoint`.

Un exemple plus conséquent d'ensemble d'objets comportant des relations circulaires est donné dans le paragraphe suivant.

7.1.2 Exemple d'une base de connaissances commerciale

Afin d'illustrer la notion de circularité dans les descriptions des objets, ainsi que les problèmes qu'elle peut engendrer pour une procédure de construction automatique de classes, nous utilisons l'exemple d'une base de connaissances issue du domaine commercial. L'exemple est inspiré d'une application réelle des techniques de fouille de données rapportée dans [Kosters *et al.*, 1999]. Nous avons conservé la structure de l'application, en modifiant les objectifs du processus global d'extraction de connaissances. En cela, nous nous sommes basés sur des discussions avec les auteurs de l'article qui ont mis en évidence l'intérêt de notre manière de poser le problème et l'utilité potentielle des connaissances qu'il sera possible d'extraire des données.

La modélisation du domaine

Dans une base de connaissances qui constitue un modèle du domaine représenté, généralement plusieurs familles d'entités sont identifiées. Par exemple, dans une base de connaissances modélisant une entreprise commerciale, les types d'entités sont les clients, les produits, les fournisseurs, etc. Les diverses familles sont organisées en des ensembles de classes séparés (même si une taxonomie unique est constituée).

Des liens entre objets de familles variées existent : un produit est commandé par un client ou fourni par un fournisseur. Ces liens se répercutent au niveau des classes, les classes sur une famille particulière sont connectées aux ensembles, correspondant aux familles liées. Par exemple, la classe des clients étrangers est liée à la classe des produits exportés.

L'organisation de l'ensemble total d'objets en classes est le but final de la conception, toutefois le processus est incrémental : il peut commencer avec une structure plate sur chaque famille, c'est-à-dire sans classe spécifique, et avancer par des ajouts progressifs des (morceaux) de taxonomies jusqu'à ce que le résultat soit jugé globalement satisfaisant. À un moment précis de la construction de la base de connaissances il peut donc s'avérer nécessaire de construire des taxonomies sur plusieurs, voire toutes les familles d'objets.

L'identification des classes

Prenons donc l'exemple des taxonomies sur les produits et sur les clients. Chaque classe d'une des taxonomies est décrite par une classe ou plusieurs classes de l'autre famille.

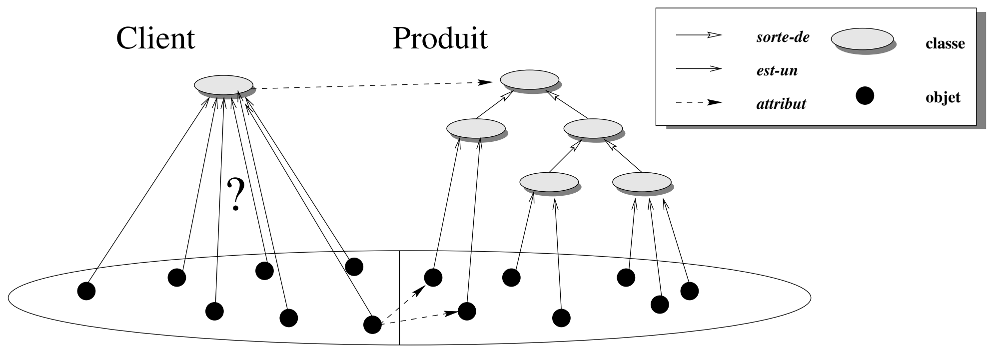


FIG. 7.1: *Construction d'une hiérarchie de classes en s'appuyant sur une hiérarchie déjà existante : les classes sur le concept* `client` *sont construites en référence aux classes sur* `produit` *qui ont été préalablement définies.*

Plusieurs scénarios sont possibles lors de la construction de la base. Dans un premier scénario, une taxonomie satisfaisante sur les produits existe déjà dans la base. Celle-ci traduit, par exemple, une nomenclature propre à l'entreprise. En revanche, il n'y a pas de classes sur les clients qui reflètent de manière adéquate leurs profils de consommateurs selon les catégories de produits existantes (voir figure 7.1). Le but est donc de construire de telles classes et la construction doit s'appuyer sur la structure des classes de produits. Les classes construites sur les clients peuvent être utilisées par la suite à des fins de marketing.

À l'inverse, des classes de clients peuvent exister qui reflètent leurs profils en tant qu'acheteurs et le but sera de répartir les produits en fonction de leur succès parmi les différentes catégories de clients. Les classes construites sur les produits peuvent suggérer des modifications dans la gamme des produits (par exemple elle peut révéler la demande potentielle d'un nouveau type de produit qui combine les caractéristiques de plusieurs autres produits).

Dans les deux cas, une taxonomie de classes préalablement établie est utilisée dans la construction d'une autre taxonomie. Une des caractéristiques de l'outil d'aide est donc de pouvoir s'appuyer sur les classes existantes sur une famille pour construire des classes sur une famille qui en dépend.

Mais, dans un second scénario, plus général, il est possible de commencer un processus de construction automatique en l'absence de toute structure de classes. Qui plus est, certaines familles d'objets peuvent être reliées de telle manière que les classes sur ces familles se réfèrent mutuellement.

Sur l'exemple commercial, cette situation (voir figure 7.2) se produit lorsque le concepteur de la base désire identifier des classes de produits qui, en plus d'une ressemblance autre, se consomment par des clients ayant un profil semblable. En d'autres termes, des classes spécifiques (de faible cardinalité) sur les produits correspondront à des classes spécifiques sur les clients et *vice versa*.

Signalons que de telles classes ne se produisent pas naturellement lorsqu'une des taxonomies est donnée au départ et l'autre est construite par référence (premier scénario). En effet, si les classes de produits sont données et celles sur les clients sont recherchées, les clients consommant

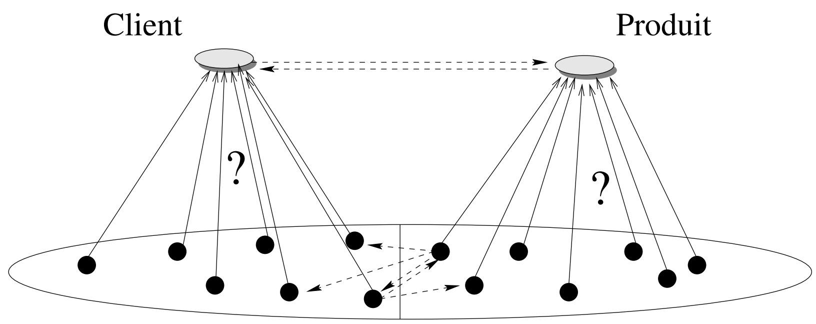


Fig. 7.2: *Construction de hiérarchies de classes sans connaissance taxonomique préalable : les classes sur les concepts* `client` *et* `produit` *doivent être construites en même temps et en faisant référence les unes aux autres.*

le même type de produit auront tendance à se retrouver au sein d'une classe spécifique. En revanche, les produits consommés par le même type de clients peuvent être situés dans des classes très disparates.

Le fait de pouvoir ajuster les deux structures de classes l'une par rapport à l'autre peut apporter des solutions à plusieurs problèmes. Par exemple, la détection d'un ensemble de produits qui est fortement corrélé avec les consommateurs ayant un certain profil, disons leur localisation géographique proche, peut avoir, en plus de tout autre bénéfice, un impact sur la manière dont les produits sont distribués, voire sur le choix des fournisseurs.

7.1.3 Pourquoi tenir compte de la circularité ?

Il faut noter que la comparaison entre individus n'est pas la seule opération dans une base de connaissances que la circularité rend plus difficile. D'autres opérations, encore plus fréquentes, telle que la classification ou le filtrage, rencontrent des difficultés en présence de circularités. Les problèmes provoqués par les descriptions circulaires au sein d'une base de connaissances ont fait objet des études approfondies dans le domaine des logiques de description [Nebel, 1990; Nebel, 1991]. Elles ont conduit à la distinction de plusieurs cas de co-définition principalement entre concepts terminologiques qui sont comparables aux concepts et aux classes en Troeps (les classes d'un modèle à objets). Nous les citons ici à titre d'étape préliminaire pour une future adaptation des techniques décrites dans la suite à un formalisme terminologique.

Parmi les divers cas de circularité que Nebel distingue dans sa thèse [Nebel, 1990], nous avons retenus les deux suivants :

- concepts simplement circulaires moyennant restrictions (*simple restriction circular concepts*),
- concepts sémantiquement circulaires (*semanticly circular concepts*).

Le premier type de circularités correspond à des concepts co-définitions dans Troeps sans qu'il y ait une co-définition entre objets. Le second type, bien qu'il porte encore sur les concepts, inclut la co-définition entre objets.

Il est à noter que la préoccupation principale dans le champ des logiques de description est l'établissement d'une sémantique dénotationnelle des expressions du langage de description des connaissances, les termes structurés. C'est pourquoi tous les problèmes sont traités sous cet angle et les considérations se rapportent majoritairement sur les concepts. Les résultats principaux en la matière sont l'établissement de plusieurs sortes de sémantiques pour les circuits terminologiques.

La description des divers types de sémantique sort du cadre de notre étude (le lecteur intéressé pourra consulter [De Giacomo et Lenzerini, 1997]), nous allons simplement noter le principe commun de ces sémantiques qui est le calcul d'un *point-fixe*. Ainsi, les ensembles d'individus qui constituent l'interprétation d'un concept circulaire, sont calculés en tant que points-fixes d'une fonction tirée des définitions des concepts :

$$C \doteq F(C).$$

Ici F dénote un terme structuré. La partie droite doit être lue comme «C est un sous-terme de F».

Notons au passage que notre vision centrée-instances est, dans un certain sens, complémentaire à la vision des logiques de description. Par ailleurs, nous ne nous arrêterons pas à la sémantique de nos propres classes : elles en possèdent nécessairement une car elles sont obtenues par un processus de nature inductive à partir d'un «modèle» que l'ensemble des objets constitue.

À part l'intérêt purement théorique de la maîtrise de la circularité (des exemples des structures intrinsèquement récursives comme les arbres, les listes, les *streams*, etc.), les travaux sur les circuits terminologiques ont également cherché à motiver l'utilisation et donc le traitement de ces circuits. Un des arguments que Nebel avance en faveur des circuits de second type (et donc des circuits dans les objets) est le pouvoir inférentiel intéressant que peuvent avoir les concepts qui en découlent. Plus précisément, les liens qu'un objet entretient avec un autre peuvent être déterminants vis à vis de son appartenance à une classe (voir exemple des voitures et moteurs de voitures dans [Nebel, 1990], chapitre 5).

Cette remarque est très importante pour notre propre étude qui cherche à *reconstituer* des classes de ce genre à partir de leurs objets membres. En outre, elle confirme notre conviction que les informations détenues par les relations qu'un individu entretient avec d'autres individus sont de première importance lors de la constitution des classes, et par conséquent pour la comparaison des individus.

Nous terminons cette discussion par le rappel de l'approche de Kbg vers le traitement des circularités (voir section 2.3.5 du chapitre 2). Sans que ce terme soit explicitement présent dans les publications sur le système, la circularité occupe une place centrale en Kbg. En effet, la sémantique accordée aux relations inter-individu dans le système est une sémantique réflexive dans le sens où un littéral

$$P(X_1, X_2, \dots, X_n)$$

faisant apparaître n individus s'interprète comme un ensemble complet de liens doublement orientés entre couples d'individus (X_i, X_j). Ces liens ne sont pas exprimés de manière explicite, mais ils sont pris en compte lors du calcul des similarités entre individus. Ainsi, un couple de littéraux

$$P(X_1, X_2, \dots, X_n) \quad et\ P(Y_1, Y_2, \dots, Y_n)$$

donne lieu à une clique orientée où les sommets sont des similarités entre individus du même rang, $sim_O(X_i, Y_i)$ et les arcs expriment des dépendances entre les valeurs des fonctions. Lors de la comparaison d'entités complexes, les cliques associées à chaque couple de littéraux de même symbole prédicatif s'enchaînent en un graphe global dont les chemins orientés expriment les dépendances entre similarités de couples d'individus liés. Il reste à noter que tous ces chemins s'interprètent à double sens, les dépendances entre couples sont donc toujours en double sens. Afin de prendre en compte les influences mutuelles entre couples d'individus, les similarités sont calculées à l'aide d'un système d'équations linéaires.

Fondamentalement, on retrouve la même approche dans Kbg que dans les logiques de description : les valeurs récursivement définies sont calculées en point-fixe d'une fonction. Dans le premier cas il s'agit des valeurs d'une fonction numérique, le calcul est donc fait à l'aide d'équations à coefficients numériques. Il n'en reste pas moins que les solutions du système soient des points-fixes de la fonction caractéristique du système. Dans le second cas, la fonction en question est une fonction ensembliste.

En simplifiant sensiblement, on pourra remarquer une certaine dualité entre ces deux approches : alors que dans KBG l'accent est mis sur les éléments individuels qui sont traités à l'aide des techniques numériques, dans l'approche terminologique on cherche à établir une sémantique sur les descriptions génériques et on emploie des techniques de la théorie des modèles, donc essentiellement discretes.

Pour clôre ce panorama, nous pouvons remarquer que le phénomène de la circularité reste encore méconnu. Toutefois, des progrès sont faits dans sa maîtrise. En ce qui concerne le regroupement sur des individus ayant des relations circulaires, on ne peut pas soutenir que ces relations sont systématiquement pertinentes pour leur regroupement, comme c'est le cas dans KBG. Cependant, dans un sous-ensemble des cas, les relations semblent effectivement transmettre une influence entre individus qui est essentielle pour leur catégorisation et de là pour leur comparaison. À part l'exemple du paragraphe précédent, nous pouvons citer deux autres applications où les relations circulaires semblent de première importance.

L'une des applications provient du domaine chimique organique, elle concerne l'appariement des structures de molécules. Ici, la position mutuelle des atomes au sein d'une molécule et leur liens, directs ou non, avec d'autres atomes jouent un rôle prépondérant dans l'évaluation d'une similarité entre deux atomes de deux molécules.

L'autre application est de nouveau du domaine commercial [Morik et Brockhausen, 1997]. Cette fois-ci il s'agit de départager les clients répertoriés dans une base de données selon leurs qualités de bons ou moins bons acheteurs de produits offerts par la société. Avec l'ensemble d'autres caractéristiques pertinentes (âge, salaire, etc.) des liens conjugaux sont représentés dans la base. Ceux-ci sont importants car ils indiquent des influences potentielles entre époux (en d'autres termes, la qualité de bon acheteur serait *transmissible*).

7.1.4 Quels outils pour les descriptions circulaires?

Face à une circularité dans les descriptions des individus, il est possible d'adopter plusieurs attitudes.

La plus simple consiste à « casser» les circuits, en enlevant certaines relations de l'ensemble des attributs considérés. Sans les attributs circulaires, les individus peuvent être traités avec succès par quasiment toutes les méthodes de regroupement. Cette approche a l'avantage de la simplicité et du moindre coût algorithmique. Elle conduit, néanmoins, à une perte d'informations qui peuvent être potentiellement intéressantes. De plus, l'approche n'est pas toujours praticable car il peut s'avérer impossible d'enlever « seulement certains» attributs. Par exemple, dans le cas des clients-conjoints, il n'est pas possible d'omettre le lien conjugal d'un sous-ensemble de personnes sans créer un déséquilibre et donc biaiser les résultats d'une éventuelle analyse. L'omission générale des liens conjugaux conduit à son tour à une perte totale des informations relationnelles, l'ensemble des personnes obtenu perd beaucoup de son intérêt.

Une seconde possibilité est l'application d'une technique de point-fixe directement sur les classes recherchées, c'est-à-dire dans la veine des logiques de description. En effet, on peut imaginer sur l'exemple de la distribution, une construction itérative qui permet de ne travailler que sur un seul ensemble d'objets à la fois. Ainsi, on commencerait avec les produits, en les regroupant sans prendre en compte leurs relations avec les clients. L'étape suivante serait le regroupement des clients en fonction des classes sur les produits. Ensuite, les produits seront de nouveau regroupés, cette fois-ci en prenant en compte les classes sur les clients, préalablement construites.

Un tel processus s'arrêtera lorsque les taxonomies construites par une étape ne se distinguent que très peu des taxonomies de l'étape précédente.

Cette approche itérative, si on la résume de manière plus formelle, consiste à calculer le plus petit point fixe d'une fonction qui exprime explicitement les liens de dépendance qui existent entre les classes des deux taxonomies. Outre la difficulté de sa définition, bien qu'elle puisse sembler assez naturelle, son calcul est de loin trop complexe pour une réalisation. De plus, à cause de l'impossibilité d'en donner une formulation analytique, nous ne saurons pas garantir sa convergence par une preuve.

Pour toutes ces raisons, nous lui préférons un calcul de point fixe non plus d'une dépendance entre classes, mais d'une fonction de dissimilarité qui permet de prendre en compte les circularités dans les objets. Cette fonction se définit de manière identique à celle présentée dans le chapitre 5. Cela constitue notamment la puissance de notre approche qui permet de définir une fonction unique capable de traiter des cas aussi variés que les descriptions d'objets simples, c'est-à-dire sans relations, et les descriptions d'objets circulaires. La seule différence dans ce cas est dans le calcul de la fonction d^o, qui est fait à l'aide d'un système d'équations. En d'autres termes, les valeurs de dissimilarité sont des points fixes de la fonction sous-jacente, calculés en tant que solutions du système.

Afin de pouvoir définir une procédure de calcul, c'est-à-dire la manière de composer et résoudre le système d'équations, nous allons tout d'abord fournir quelques définitions utiles.

7.2 Définitions

Tout d'abord, nous définissons la relation de co-définition entre objets. Ensuite, la structure permettant de détecter les co-définitions entre objets est caractérisée. Finalement, on décrit les couples d'objets en co-définition qu'il est nécessaire de considérer pour le calcul de la dissimilarité.

7.2.1 La co-définition entre objets

Intuitivement, deux objets o et o' sont en co-définition s'il existe deux séquences d'attributs tels que o est «lié» par l'une à o' et *vice versa*.

Avec plus de précision, la co-définition dans les objets est définie comme dans le cas des concepts, en deux étapes. Tout d'abord, nous dirons qu'un objet o *dépend directement* d'un autre objet o', s'il existe un attribut a du concept de o, soit $k = concept(o)$, tel que o' est (appartient à) la valeur de a pour o. Ainsi, la relation de dépendance, (notée $\rightharpoonup$) est définie sur O, l'ensemble de tous les objets d'une base de connaissances.

Définition 7.1. *Pour un couple d'objets o et o' $o \rightharpoonup o'$ ssi $\exists a \in attributs(concept(o))$ tel que :*

$$o' = o.a \text{ ou } o' \in o.a.$$

De même, la fermeture transitive de la relation $\rightharpoonup$, notée $\rightharpoonup^+$, sur les objets est définie. Intuitivement, un objet o dépend d'un autre objet o' s'il existe une séquence d'attributs $\{a_i\}_{i=1,k}$, et une liste d'objets $\{o_i\}_{i=0,k}$ tels que :

- $o_0 = o$ et $o_k = o'$,
- pour tout i de 1 à k , $o_{i-1}.a_i = o_i$.

D'une manière plus formelle, la dépendance se définit ainsi :

Définition 7.2. *Pour un couple d'objets o et o' $o \rightharpoonup^+ o'$ ssi l'une des deux conditions est vérifiée :*

- $o \rightharpoonup o'$,
- $\exists \bar{o}$ *tel que* $o \rightharpoonup \bar{o}$ *et* $\bar{o} \rightharpoonup^+ o'$.

Dans ce contexte, deux objets o et o' sont *co-définis* si chacun dépend de l'autre. En d'autres termes, il doit exister un circuit formé par des attributs qui passe par chacun des deux objets.

La relation de *co-définition* sera notée par $\leftrightharpoons$.

Définition 7.3. *Pour un couple d'objets o et o' $o \leftrightharpoons o'$ ssi $o \rightharpoonup^+ o'$ et $o' \rightharpoonup^+ o$.*

Exemple

Prenons l'ensemble des objets dessinés sur la figure 7.3. Les six objets sont des instances du concept `humain` qui est également dessiné. Il définit l'attribut `conjoint` qui est de nature lien et de type `humain`.

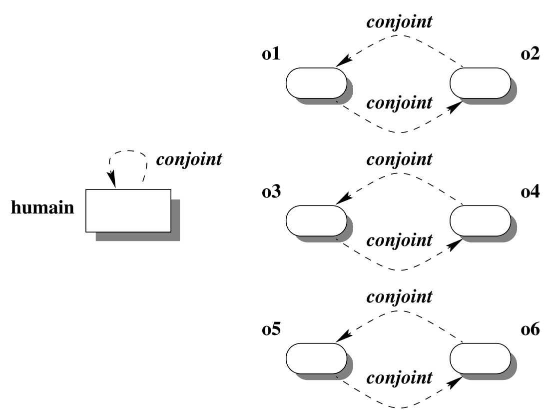


FIG. 7.3: *Ensemble de six objets, instances du concept* `humain`*. Les objets représentent trois couples d'époux, ils sont donc co-définis via l'attribut* `conjoint`*.*

L'attribut `conjoint` introduit un circuit dans le graphe des dépendances, le concept `humain` est donc en co-définition avec lui-même (`humain` $\leftrightharpoons$ `humain`). Ceci se traduit par une co-définition mutuelle dans les objets, les six objets constituent trois couples d'époux : (o_1, o_2), (o_3, o_4) et (o_5, o_6).

Dans la suite, nous donnons un cadre formel qui nous permet de raisonner sur les dissimilarités dans le cas de la co-définition.

7.2.2 Réseaux d'objets

Les descriptions des objets, tels qu'ils ont été présentés dans le chapitre 1, lient des caractéristiques propres aux entités représentées avec des relations. Ainsi, un objet n'est jamais isolé au sein d'une base de connaissances, il est lié par ses attributs à un ensemble d'autres objets. C'est le cas par exemple d'un objet composite dont les attributs de nature composant indiquent les parties. À leur tour, les objets référencés peuvent en référencer d'autres. Il en résulte une dépendance d'un objet envers tous les objets atteinables à partir de celui-ci par un chemin.

Il est donc important de fixer les limites de la partie d'une base de connaissances qui sera prise en compte pour comparer un couple d'objets. Pour cela, nous introduisons la notion de réseaux d'objets qui est comparable à la fermeture transitive d'une relation hypothétique « être référencé par ».

Notons $R(o)$ la structure induite par les attributs relationnels à partir d'un objet o. Cette structure sera appelée *le réseau d'objets* correspondant à o ou tout simplement le *réseau* de o. Les réseaux sont l'équivalent du graphe des dépendances pour les objets.

Le réseau $R(o)$ est un multi-graphe orienté étiqueté $R(o) = \langle X_o, E_o, A_o \rangle$ où l'ensemble des sommets X est composé par les objets de la base atteinables depuis o par une séquence d'attributs relationnels et l'ensemble des arcs E correspond aux attributs relationnels qui lient ses objets.

Définition 7.4. *Le réseau $R(o)$ associé à l'objet o est un multi-graphe orienté dont les sommets sont des objets et les arcs sont les attributs relationnels qui les lient :*

$$R(o) = \langle X_o, E_o, A_o \rangle$$

ou

$$X_o = \{o' | o' \in O, o \rightharpoonup^{+} o'\}, \qquad A_o = \textstyle\bigcup_{o' \in X_o} A^r(o'), \qquad E_o \subseteq X_o \times A_o \times X_o$$

et

$$\langle o, a, o' \rangle \in E_o \text{ ssi } \exists a \in A_o \,;\, o' = o.a \text{ ou } o' \in o.a.$$

Ci-dessus, la fonction $A^r : O \rightarrow 2^A$ rend l'ensemble des attributs relationnels qui sont renseignés dans un objet de la base.

Pour l'instant, nous nous limitons à des attributs monovalués. À la fin les extensions nécessaires pour le traitement des collections seront précisées.

Proposition 7.1. *Il existe un homomorphisme h entre le réseau d'un objet o, $R(o)$ et le graphe de dépendances entre concepts $\mathcal{K}$ tel que :*

- $\forall \bar{o} \in R(o), h(\bar{o}) = concept(\bar{o})$,
- *h préserve les étiquettes sur les arcs, c'est-à-dire les noms des attributs.*

L'application du réseau d'objets sur le graphe des dépendances n'est pas nécessairement injective. Par exemple, à un réseau formé par deux objets du concept `humain` mutuellement liés par leurs attributs `conjoint` correspondra un sous-graphe de $\mathcal{K}$ composé par un seul sommet, k et un seul arc étiqueté `conjoint` qui constitue une boucle sur k.

Dans la suite, nous rappelons les notions classiques de la théorie des graphes et leur signification dans le contexte des réseaux d'objets. Tout d'abord, un *chemin* est une séquence d'arcs adjacents.

Définition 7.5. *Entre deux objets o et o' existe un chemin $\langle a_1, ..., a_n \rangle$ dans un réseau $R(\bar{o})$ ssi pour tout i de 1 à n*

$$\exists o_i \text{ tel que } \langle o_{i-1}, a_i, o_i \rangle \in E$$

et

$$o_0 = o,\ o_n = o'.$$

Un chemin correspond à une suite d'indirections sur les attributs des objets.

Ensuite, puisque nous nous intéressons à la circularité, un circuit est un chemin dont le début et la fin coïncident.

Définition 7.6. *Un **circuit** est un chemin dont l'origine et la destination sont le même sommet.*

On pourra dire d'un ensemble d'objets se trouvant sur un circuit qu'ils sont *co-définis* dans la mesure où chacun des objets est impliqué, directement ou non, dans la définition de tout autre objet de l'ensemble. Pour un objet particulier, l'ensemble des objets co-définis se trouvent sur un circuit du réseau sous-jacent. Cet ensemble correspond à une *composante fortement connexe* (CFC) du réseau. Rappelons que dans une CFC pour chaque couple d'objets existe un chemin qui les lie. Dans un réseau il peut y avoir plusieurs CFC, nous nous intéressons seulement à celle qui inclut l'objet d'origine du réseau.

Définition 7.7. *La CFC du réseau $R(o)$ à laquelle appartient l'objet origine est notée $\mathcal{CFC}(o)$.*

Afin de simplifier le discours, nous allons considérer désormais que le réseau de chaque objet ne comporte qu'une seule CFC, c'est-à-dire $\mathcal{CFC}(o) = R(o)$.

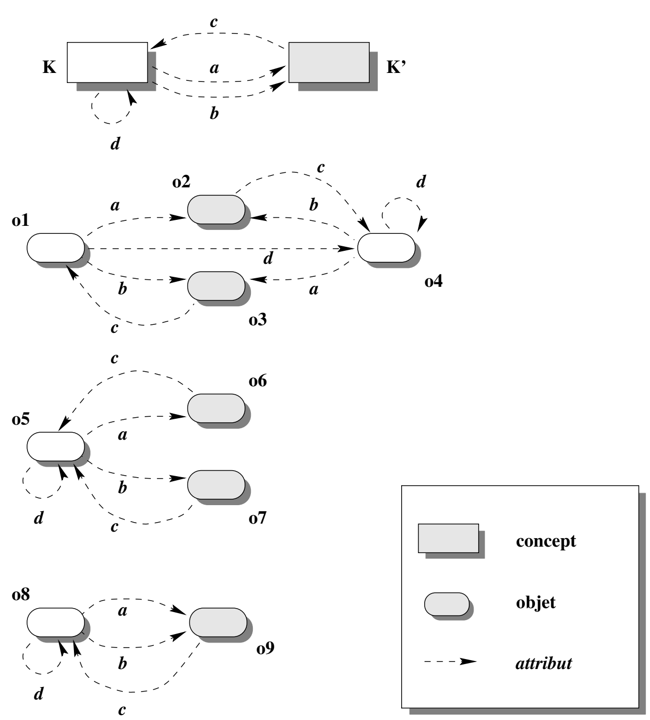


FIG. 7.4: *Représentation d'un graphe et trois réseaux d'objets.*

7.2.3 Appariement entre réseaux

Notre but sera de décrire une manière de comparer les objets par rapport à leurs relations avec d'autres objets dans leurs réseaux respectifs. Cela nécessite en particulier d'identifier l'ensemble des autres couples d'objets dont la comparaison sera nécessaire. Soit un couple d'objets, o, o' du concept k dont la dissimilarité $d_k^o(o, o')$ est à calculer. Soient encore $\mathcal{CFC}(o) = \langle X, E\rangle$ et $\mathcal{CFC}(o') = \langle X', E'\rangle$ les CFC dans les réseaux associés.

Nous allons distinguer l'ensemble des couples d'objets des deux réseaux dont la dissimilarité dépend récursivement de de la dissimilarité $d_k^o(o, o')$.

Définition 7.8. *Soit $Couples(o, o')$, l'ensemble des couples d'objets (o_i, o'_j) tels que :*

- *$o_i \in X$ et $o_j \in X'$,*
- *o_i est atteignable depuis o dans $\mathcal{CFC}(o)$ par le même chemin par lequel o'_j est atteignable depuis o' dans $\mathcal{CFC}(o')$,*
- *o est atteignable depuis o_i dans $\mathcal{CFC}(o)$ par le même chemin par lequel o' est atteignable depuis o'_j dans $\mathcal{CFC}(o')$.*

On pourra remarquer que o_i et o'_j appartiennent au même concept car tous les attributs sont typés. Prenons l'exemple des trois réseaux sur la Fig. 7.4 qui sont déjà des CFC. Si l'on compare les réseaux non-restreints, c'est-à-dire avec $A_K = \{a, b, d\}$ et $A_{K'} = \{c\}$, alors les couples extraits des CFC $\mathcal{CFC}(o_5)$ et $\mathcal{CFC}(o_8)$ sont

$$Couples(o_5, o_8) = \{(o_5, o_8), (o_6, o_9), (o_7, o_9)\}.$$

Pour $\mathcal{CFC}(o_1)$ et $\mathcal{CFC}(o_5)$ les couples sont

$$Couples(o_1, o_5) = \{(o_1, o_5), (o_2, o_6), (o_4, o_5), (o_2, o_7), (o_3, o_7), (o_3, o_6)\}.$$

Le couple (o_2, o_7) y est inclus car d'une part, les objets o_2 et o_7 sont accessibles par un chemin *acb* depuis o_1 et o_5 respectivement et d'autre part, o_1 et o_5 sont respectivement accessibles par un chemin *cac*.

Si maintenant on restreint les attributs, soit $A_K = \{a\}$, alors certains couples n'apparaissent plus dans $Couples(o_1, o_5)$:

$$Couples(o_1, o_5) = \{(o_1, o_5), (o_2, o_6), (o_4, o_5), (o_3, o_6)\}.$$

Ce dernier exemple vient de montrer que l'appariement particulier entre réseaux que nous venons de décrire ne comporte pas nécessairement tous les couples d'objets du même concept. Ce point marque une différence avec le cadre de la LPO dans KBG où tous les individus des exemples sont comparables *a priori*. Soulignons que le nombre des couples d'objets dans le calcul de la dissimilarité est un facteur de première importance pour la complexité algorithmique de l'approche.

Cependant, dans la suite nous n'allons pas considérer des restrictions sur les attributs des objets et cela par souci de simplifier les notations. En d'autres termes, nous considérons que tous les attributs des concepts sont pertinents et ignorons pour l'instant la notion de point de vue. Celle-ci peut être facilement intégrée en prenant des restrictions à chaque fois que l'ensemble des attributs est concerné.

7.2.4 Ensembles d'attributs pertinents

Nous allons de plus distinguer deux ensembles d'attributs. Tout d'abord, $Circ(o, o')$ est le sous-ensemble de A composé par les attributs qui lient o ou o' à d'autres objets dans $Couples(o, o')$.

Définition 7.9. *$Circ(o, o')$ est l'ensemble des attributs a tels qu'il existe deux circuits identiques dans les deux CFC, $\mathcal{CFC}(o)$ et $\mathcal{CFC}(o')$ qui passent par l'attribut a de o et o' respectivement.*

En d'autres termes, les attributs dans $Circ(o, o')$ sont ceux dont la contribution va être calculée de manière simultanée, le reste des contributions pouvant être évalué préalablement. Par exemple, dans $\mathcal{CFC}(o_1)$ et $\mathcal{CFC}(o_5)$ sur la figure 7.4, on retrouve $Circ(o_1, o_5) = \{a, b, d\}$.

Parmi $Circ(o, o')$, on distinguera $Boucle(o, o')$, les attributs qui forment un circuit de longueur un, c'est-à-dire qu'ils bouclent sur l'objet qui devient source et destination en même temps. Formellement,

Définition 7.10. *$Boucle(o, o') = \{a | a \in A, o.a = o, o'.a = o'\}$.*

Par exemple, $Boucle(o_1, o_5) = \emptyset$, alors que $Boucle(o_4, o_5) = \{d\}$.

La section suivante est dédiée à la mise en œuvre des notions définies pour un calcul effectif de la fonction d^o sur un couple de réseaux comportant des objets co-définis.

7.3 Calcul effectif de la mesure

Nous allons voir comment les structures des deux réseaux se traduisent en une structure calculatoire qui reflète les dépendances entre couples d'objets. Cette nouvelle structure, exprimée sous forme d'un système d'équations, permet d'évaluer les influences mutuelles entre ces couples en leur donnant une valeur numérique.

Dans la suite, nous montrons d'abord la procédure de composition du système d'équations. Ensuite, l'existence d'une solution pour ce système est prouvée. L'extension de la technique sur les collections d'objets est présentée, avec une preuve de la convergence du calcul. Finalement, une indication est donnée sur la méthode de résolution du système que nous discutons avec plus de détail dans la section 7.6.1 de ce chapitre.

7.3.1 Construction d'un système d'équations

Soit n le nombre des couples dans $Couples(o, o')$. À chaque couple (o_i, o'_j) nous faisons correspondre une variable x_t. Soient k_t le concept de o_i et o'_j et A_t l'ensemble des attributs de k_t. Pour chaque variable x_t, nous tirons une équation de l'expression de la dissimilarité du couple d'objets respectif (voir equation 5.4) :

$$d^o_{k_t}(o_i, o'_j) = \sum_{a \in A_t} \lambda_a * \delta^f_a(o_i.a, o'_j.a) \tag{7.1}$$

L'équation est obtenue en posant pour chaque membre $\delta^f_l(o_i.a, o'_j.a)$ une expression équivalente. Dans un premier temps, à x_t est associé l'expression de $d^o_{k_t}(o_i, o'_j)$ qui est séparée en deux selon les attributs

$$x_t = \sum_{a \in Circ(o_i, o'_j)} \lambda_a * \delta^f_a(o_i.a, o'_j.a) + \sum_{a \in A_t - Circ(o_i, o'_j)} \lambda_a * \delta^f_a(o_i.a, o'_j.a) \tag{7.2}$$

Si l'attribut a est dans $Circ(o_i, o'_j)$, cela veut dire que c'est un attribut relationnel qui lie chacun des deux objets o_i et o'_j à d'autres objets qui se trouvent sur les CFC respectives. Dans ce cas, la fonction $\delta^f_a(o_i.a, o'_j.a)$ ne peut être calculée préalablement à la résolution du système d'équations. Le couple $(o_i.a, o'_j.a)$ se trouve nécessairement dans $Couples(o, o')$ et donc une variable, soit x_s, lui correspond. Ainsi, x_s est substitué à $\delta^f_a(o_i.a, o'_j.a)$ dans 7.2.

Dans le cas contraire, l'attribut a est soit une propriété soit une relation qui ne se trouve pas simultanément sur les deux CFC. Puisque la fonction $\delta^f_a(o_i.a, o'_j.a)$ ne dépend pas de $d^o(o, o')$ sa valeur peut être établie sans calcul récursif.

De cette manière, nous obtenons une nouvelle expression pour x_t qui est une équation à plusieurs inconnues :

$$x_t = b_t - \sum_{s=1}^{n} c_{t,s} * x_s \tag{7.3}$$

Dans le membre de degré zéro b_t les contributions de tous les attributs qui ne participent pas à des circuits sont rassemblées :

$$b_t = \sum_{a \in A_t - Circ(o_i, o'_j)} \lambda_a * \delta_a^f(o_i.a, o'_j.a) \tag{7.4}$$

Cette quantité peut être calculée indépendamment de la résolution du système, ceci est donc fait auparavant.

Les coefficients $c_{t,s}$ sont la négation des sommes de poids d'attributs qui ont comme source les objets dans le couple correspondant à x_t et comme arrivée ceux du couple correspondant à x_s. En effet, les couples d'objets $(o_i.a, o'_j.a)$ liés à o_i et o'_j respectivement par l'attribut a ne sont pas nécessairement différents. Dans le cas d'une telle coïncidence entre plusieurs attributs, la variable x_s correspondant au couple va participer dans l'équation déterminant x_t avec un coefficient égal à la somme des poids respectifs. Plus précisément, soit pour un s donné $A_{t,s}$ le sous-ensemble de A_t tel que pour chaque a dans $A_{t,s}$, le couple $(o_i.a, o'_j.a)$ correspond à x_s. Alors, les coefficients $c_{t,s}$ sont calculés comme suit :

$$c_{t,s} = - \sum_{a \in A_{t,s}} \lambda_a \tag{7.5}$$

Si l'ensemble $A_{t,s}$ est vide, alors aucune dépendance directe entre x_t et x_s n'existe, la valeur de $c_{t,s}$ est donc zéro.

Finalement, la valeur de $c_{t,t}$ est obtenue par une soustraction : de 1, le facteur qui provient de la partie droite de l'expression 7.2, on soustrait la somme des poids des attributs qui bouclent simultanément sur les objets d'origine. En d'autres termes,

$$c_{t,t} = 1 - \sum_{a \in Boucle(o_i, o'_j)} \lambda \tag{7.6}$$

En faisant la même transformation pour chaque couple d'objets dans $Couples(o, o')$ on obtient un système à n équations et n inconnues.

$$\mathbf{C} * \mathbf{X} = \mathbf{B} \tag{7.7}$$

$$\mathbf{C} = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{pmatrix}$$

Ayant formé la matrice du système d'équations, il convient de s'interroger sur l'existence d'une solution. La section suivante montre que grâce à sa forme particulière, la dissimilarité d^o peut toujours être calculée comme la solution unique du système ci-dessus.

7.3.2 Existence et unicité de la solution

Le système d'équations est linéaire sous l'hypothèse que tous les attributs dans les CFC sont monovalués. Pour l'instant, nous allons nous tenir à cette hypothèse et voir plus loin comment les contraintes qu'elle impose peuvent être relâchées.

Afin de prouver qu'une solution unique existe pour le système obtenu, il suffira de prouver que la matrice $\mathbf{C}$ des coefficients et sa transposée $\mathbf{C}^T$ sont *diagonales dominantes* (voir [Golub et van Loan, 1983]). Cela signifiera que les deux matrices sont définies positives avec un déterminant strictement positif. Une matrice est diagonale dominante si pour chaque ligne, la somme des valeurs absolues des éléments hors la diagonale est strictement inférieure à la valeur absolue de l'élément sur la diagonale. Notons que dans le cas marginal où une égalité est atteinte, le système peut avoir un nombre infini de solutions car le système est sous-déterminé. Dans notre cas cela se produirait si la somme des $|c_{i,j}|$ devient égale à 1 (à cause de la normalisation des poids et d'après l'égalité 7.6). Cela signifierait que pour un couple d'objets (o_i, o'_i), tous les attributs qui participent à l'évaluation de la dissimilarité sont dans $Circ(o_i, o'_i)$. Cette éventualité est très peu probable[29], nous allons donc l'exclure en prenant l'hypothèse qu'au moins un attribut de $A_i = attributs(concept(o_i))$ n'est pas dans $Circ(o_i, o'_i)$. Formellement, nous allons prouver que

Proposition 7.2. *Si pour tout i, $A_i - Circ(o_i, o'_i) \neq \varnothing$, alors l'inégalité suivante est vérifiée pour tout i aussi :*

$$\forall i, \sum_{j=1, j\neq i}^{m} |c_{i,j}| < |c_{i,i}| \tag{7.8}$$

Nous avons donc démontré que la matrice du système est diagonale-dominante et donc que le système possède une solution unique.

7.3.3 Résolution du système

Les solutions du système peuvent être calculées à l'aide d'une méthode *directe*, en inversant la matrice $\mathbf{C}$. Celle-ci a l'avantage de fournir des solutions exactes, mais peut être très coûteuse en temps de calcul.

Une méthode *itérative* de calcul pourra aboutir à une solution approchée de qualité suffisante (donnée par un seuil ε). L'utilisation d'une telle méthode est conseillée par G. Bisson dans Kbg [Bisson, 1993]. Une procédure de calcul itératif est proposée dans la section 2.1.3 de ce chapitre.

Dans la section suivante, le principe du calcul itératif est utilisé dans la démonstration d'un ensemble de propriétés intéressantes de notre mesure d^o.

7.3.4 Propriétés des fonctions calculées

Cette section décrit la fonction que le système d'équations permet de calculer.

Normalisation

Notre premier but sera de montrer que les fonctions d^o calculées à l'aide d'un système sont normalisées, c'est-à-dire que toutes les valeurs de x_i pour un couple d'objets sont comprises entre 0 et 1. Dans les deux paragraphes suivants nous fournissons les résultats permettant d'affirmer la normalisation de d^o.

29. Troeps interdit la circulairité dans les attributs-clés des concepts. Cependant, si l'utilisateur écarte les clés lors de la pondération – par des poids nuls – rien ne peut être garanti.

Non-négatives Dans un premier temps nous allons démontrer que toutes les solutions d'un système sont non-négatives.

Proposition 7.3. *Toute fonction d^o calculée à l'aide du système d'équations ci-dessus n'admet que des valeurs non-négatives :*

$$d^o(o, o') \geq 0.$$

Bornées par un Le second but est de démontrer que ces valeurs sont inférieures ou égales à un.

Proposition 7.4. *Toute fonction d^o calculée à l'aide du système d'équations ci-dessus n'admet que des valeurs qui ne dépassent pas* 1 *:*

$$d^o(o, o') \leq 1$$

Propriétés de semi-métrique

Considérons les propriétés des fonctions calculées à l'aide du système ci-dessus sur un ensemble d'objets du même concept.

Proposition 7.5. *Pour tout k la fonction d^o_k calculée à l'aide d'un système d'équations du type ci-dessus est une semi-métrique.*

Il serait également intéressant de pouvoir prouver des propriétés de métrique pour les fonctions d^o_k. Malheureusement, la structure hétérogène des CFC ne permet pas une preuve suffisamment générale. Toutefois, il est possible de montrer que si tous les écarts δ^f_a sont des métriques et si les réseaux pour les divers d'objets d'un concepts k sont isomorphes, alors la fonction d^o_k est aussi une métrique.

7.4 Extension sur les collections d'objets

Dans les développements précédents, nous avons pris l'hypothèse que tout attribut relationnel a liant des objets en co-définition est monovalué. Dans cette section nous examinons les conséquences de l'introduction des relations multivaluées pour le calcul de la dissimilarité.

7.4.1 Problèmes

Lorsque les attributs conduisant à des co-définitions peuvent être multivalués, les dépendances entre les variables du système ne sont plus purement linéaires. Rappelons que les écarts sur les attributs multivalués sont calculés à l'aide de la fonction δ^m et que celle-ci comporte une étape d'appariement entre les collections comparées.

Dans la suite nous reprenons les notations introduites dans la section 7.3.4, ci-dessus. Soit un couple d'objets (o, o') du concept k et la variable x_t correspondant au couple. Soit l'attribut multivalué a de type k' tel que : $o.a = \{o_1, o_2, \ldots, o_p\}$ et $o'.a = \{o'_1, o'_2, \ldots, o'_q\}$. On supposera, pour simplifier, que tous les couples (o_i, o'_j) appartiennent à $Couples(o, o')$. Soient x_{t_1}, x_{t_2}, … ,$x_{t_{pq}}$ les variables dans le système d'équations correspondant aux couples (o_1, o'_1), … ,(o_p, o'_q). Sans perte de généralité nous pouvons considérer que $q \leq p$. Soit encore M_{opt} l'appariement de poids minimal des deux collections $o.a$ et $o'.a$.

La dissimilarité δ^m est calculée sur les couples dans M_{opt} et uniquement sur ces couples. Cela veut dire que seule une petite partie de tous les couples imaginables entre membres des collections contribuent au résultat de la comparaison entre $o.a$ et $o'.a$. On en déduit que la variable x_t ne dépend réellement que d'un sous-ensemble des variables $x_{t_1}, x_{t_2}, \ldots, x_{t_{pq}}$ et que seules ces variables parmi l'ensemble total ont des coefficients non-nuls dans l'équation associée à x_t.

Cependant, l'ensemble exact de ces variables est une fonction de M_{opt} qui est calculée à son tour d'après les dissimilarités entre couples d'objets. Cet ensemble dépend donc des valeurs des variables x_{t_j}. En résumé, les coefficients des variables x_{t_j} dépendent de leurs valeurs, les équations du système ne sont donc plus linéaires.

Dans ce cas, le système ne peut pas être résolu par une méthode directe.

7.4.2 Une solution perfectible

Une solution est proposée par Bisson qui consiste à résoudre le système non-linéaire comme une suite de systèmes linéaires. Pour cela, la méthode de résolution itérative est complétée par un mécanisme de calcul d'un appariement entre collections. À noter qu'en KBG l'équivalent d'une collection est l'ensemble des individus ayant le même rang dans différentes occurences du même prédicat au sein d'une entité complexe. L'appariement permet de fixer les coefficients dans les équations et de calculer les valeurs des variables x_i de l'étape courante.

Par ailleurs, lorsque les coefficients sont tous connus, les équations redeviennent linéaires et, qui plus est, grâce à la définition des fonctions d^o et δ^m, le système d'équations obtenu a encore une matrice qui est diagonale-dominante. Ce système a donc toujours une solution.

Bisson propose de prendre un appariement à chaque étape de la résolution itérative. Pour le calculer, un algorithme heuristique est utilisé (voir section 6.2 du chapitre 6). Le résultat est un appariement M_{app} sub-optimal, exclusif et de cardinalité maximale. Si l'on note par $d^r(o_i, o'_j)$ la valeur x_{t_l} qui correspond à (o_i, o'_j) lors de l'étape r du calcul itératif, alors le poids de l'appariement M_{app} est :

$$\sum_{(o_i, o'_j) \in M_{app}} d^r(o_i, o'_j).$$

Ce poids représente une bonne approximation du poids de l'appariement de dissimilarité minimale M_{opt}[30]. Il peut donc être utilisé dans le calcul de la valeur de x_t à l'étape $r+1$.

Le bien-fondé d'une telle technique repose sur le fait que malgré la nature non-linéaire des calculs, les suites des variables x_i^r peuvent converger. Dans ce cas, le calcul aboutira nécessairement à une solution approchée de qualité pas très mauvaise.

D'après l'auteur, le gain d'une telle approche est double, dans le coût du calcul et dans l'explicabilité du processus. Toutefois, comme nous allons le démontrer, cette procédure de calcul ne garantit pas la convergence de la résolution itérative du système d'équations. En effet, dans [Bisson, 1993] une preuve par récurrence de la non-croissance des suites de valeurs de similarité est donnée. Dans notre contexte, la proposition sous-jacente peut être formulée comme suit (en suivant les notations introduites pour la preuve de l'existence d'une soution dans la section C.3 de l'annexe C) :

> Si pour un r donné toutes les valeurs $\{x_i^{r+1}\}$ vérifient $x_i^r \leq x_i^{r+1}$, alors la somme des éléments de $Base$ dans un appariement sub-optimal calculé à l'étape r est nécessairement inférieure à la somme sur l'appariement à l'étape $r+1$.

30. Bisson prend le maximum car il travaille avec des similarités.

Si l'on note par M^r_{app} l'appariement à l'étape r entre les collections $\{o_1, \ldots, o_p\}$ et $\{o'_1, \ldots, o'_q\}$ selon les valeurs de $d^r(o_i, o'_j)$ et la somme sur cet appariement :

$$\sum_{(o_i,o'_j)\in M^r_{app}} d^r(o_i, o'_j)$$

par

$$\sum_{M^r_{app}} x^r_t$$

alors la propriété peut être écrite comme suit :

$$\sum_{M^r_{app}} x^r_t \leq \sum_{M^{r+1}_{app}} x^{r+1}_t$$

Remarquons que M^r_{app} n'est qu'une approximation de M_{opt}. Dès lors, il est possible que, malgré la non-décroissance des valeurs de x_t à une étape $r+1$, par rapport à l'étape r, la somme sur l'appariement décroisse. Intuitivement, il peut arriver qu'à l'étape r un résultat particulièrement mauvais ait été calculé. Par un jeu de changement de rang entre couples, la somme sur l'appariement peut s'améliorer à tel point qu'elle dépasse en qualité celle de l'étape précédente.

Ci-dessous un exemple d'une telle situation concerne deux ensembles d'objets hypothétiques $\{o_1, o_2, o_3\}$ et $\{o_4, o_5\}$, pour qui un appariement doit être calculé. Les deux tables suivantes présentent les valeurs de la dissimilarité entre membres lors de l'étape r (table 7.1) et $r+1$ (table 7.2) respectivement. Ici $M^r_{app} = \{(o_1, o_4), (o_2, o_5)\}$ et la somme sur M^r_{app} est de $2,8$.

TAB. 7.1: *Valeurs de la dissimilarité à l'étape k*

	o_1	o_2	o_3
o_4	1	1,1	1,2
o_5	1,2	1,8	1,8

Lors de l'étape suivante, toutes les valeurs augmentent, mais pas de la même quantité. Du coup, l'ordre entre couple est changé (voir table 7.2).

TAB. 7.2: *Valeurs de la dissimilarité à l'étape $r+1$*

	o_1	o_2	o_3
o_4	1,25	1,2	1,7
o_5	1,35	1,95	2

À cette étape, $M^{r+1}_{app} = \{(o_1, o_5), (o_2, o_4)\}$ et la somme est de $2,55$ qui est inférieure à la somme de l'étape précédente. On en déduit que la valeur de la variable x_t qui dépend de cette somme peut décroître à l'étape $r+1$. Ceci remet en cause la convergence du calcul global des solutions du système d'équations. Signalons que ceci n'est pas un contre-exemple de la méthode, mais de la preuve. Autrement dit, on n'a pas montré que dans un cas précis le calcul ne converge pas, mais que les arguments pour cette convergence ne sont pas suffisants.

7.4.3 Une technique exacte

Quant à notre propre approche, afin d'établir sa convergence, montrons d'abord qu'avec le calcul exact une telle situation ne peut se produire. Formellement, nous allons démontrer par récurrence que les suites x_t^i restent non-décroissantes. Pour cela, il suffit de montrer que si à une étape r les valeurs de x_t^r ne diminuent pas, alors tous les appariements optimaux ne peuvent que mener à une augmentation de la dissimilarité globale.

Proposition 7.6. *Pour tout* r*, s'il est possible de garantir que* $\forall i, i = 1, m$ $(x_t^r \leq x_t^{r+1})$ *alors*

$$\sum_{M_{opt}^r} x_t^r \leq \sum_{M_{opt}^{r+1}} x_t^{r+1}$$

Nous avons donc prouvé que notre définition de la dissimilarité entre collections permet de satisfaire simultanément les deux conditions, la convergence du calcul et l'exactitude de la solution partielle à chaque pas.

Toutefois, l'approche reste relativement coûteuse car à chaque pas itératif, des appariements sont calculés avec une complexité au mieux quadratique en le nombre des éléments qui participent. Dans un bon nombre de cas la complexité peut être réduite. Pour cela, nous pouvons ne calculer l'appariement entre deux collections qu'une seul fois, en un pas préliminaire à la résolution du système. Ce calcul se base sur les valeurs des x_t^1, c'est-à-dire la dissimilarité immédiatement calculable. Pour cela, on dispose de plusieurs arguments :

- la constitution de M_{opt} ne dépend que du rang des couples et non de la valeur exacte de leur dissimilarité,
- dans les circuits de faible longueur, l'influence des couples liés peut être facilement anticipée. Pour un approximateur de x_t^∞ on peut prendre les valeurs x_t^1. Dans le cas de circuits de longueur deux (par exemple, relations `père` et `filles`), l'appariement établi de cette manière correspond à l'appariement définitif car le calcul itératif ne peut changer l'ordre entre les valeurs des dissimilarités.

De cette manière, on *gagne* un facteur multiplicatif dans le calcul itératif qui est $O(n_m^2)$ où n_m est la cardinalité moyenne d'une collection. Ce que l'on *perd*, éventuellement, est de la *précision*. La fonctionalité très appreciable par rapport à KBG, est qu'on peut **toujours** garantir la convergence du calcul itératif.

7.5 Exemple de calcul et d'utilisation de la mesure d^o

Nous illustrons le fonctionnement de notre mesure à l'aide de l'exemple déjà présenté dans la section 4.2.2 du chapitre 4 (voir figure 7.3).

7.5.1 Jeu de données

Rappelons que les objets sont des instances du concept `humain` liées par l'attribut `conjoint`. Les deux autres attributs considérés sont `âge` et `salaire` (voir table 7.3). L'attribut `salaire` indique le revenu mensuel d'une personne (en Euros), son type de base est réel, `intervalle [1,8 3,3]`. L'attribut `âge` est de type entier, avec pour étendue `intervalle [20 35]`.

L'attribut `conjoint` est de nature lien et de type `humain`. Les six objets constituent trois couples d'époux : (o_1, o_2), (o_3, o_4) et (o_5, o_6). Les couples forment donc de petits réseaux homogènes de deux objets du même concept. Chaque réseau constitue une seule et unique CFC.

TAB. 7.3: *Exemple d'ensemble d'objets dans* `humain`

attribut	o_1	o_2	o_3	o_4	o_5	o_6
`âge`	26	23	27	30	32	35
`salaire`	3	2,3	2,2	2,7	2,9	3,2
`conjoint`	o_2	o_1	o_4	o_3	o_6	o_5

Pour le calcul d^o, fixons les poids des attributs à `0,4` pour `conjoint`, `0,3` pour `salaire` et `0,3` pour `âge`.

7.5.2 Calcul des dissimilarités entre objets

Calculons la valeur de la dissimilarité pour le couple (o_1, o_4). D'après la formule 7.1, $d^o(o_1, o_4)$ devient :

$$d_h^o(o_1,\ o_4) = 0,3 \cdot \overline{\delta}_{age}(26,\ 30) + 0,3 \cdot \overline{\delta}_{sal}(3,\ 2,7) + 0,4 \cdot \overline{\delta}_{conj}(o_2,\ o_3)\ . \tag{7.9}$$

Les deux premiers écarts sont calculés par des dissimilarités de propriété. En prenant en compte les étendues des deux domaines respectifs, le résultat total est de `0,18`. Aucune structure de classes n'étant disponible sur `humain`, à $\overline{\delta}^a$ sur `conjoint` est substituée la dissimilarité d'objets d_h^o sur $(o_2,\ o_3)$. La formule respective pour la dissimilarité sur les objets liés, $(o_2,\ o_3)$ est :

$$d_h^o(o_2,\ o_3) = 0,3 \cdot \overline{\delta}_{age}(23,\ 27) + 0,3 \cdot \overline{\delta}_{sal}(2,3,\ 2,2) + 0,4 \cdot \overline{\delta}_{conj}(o_1,\ o_4)\ . \tag{7.10}$$

La partie calculable de la dissimilarité est de `0,14`. Avec la même substitution de d_h^o à $\overline{\delta}_{conj}$ sur `conjoint` une expression de $d_h^o(o_2,\ o_3)$ en fonction de $d_h^o(o_1,\ o_4)$ est obtenue. Globalement, par le biais du circuit induit par l'attribut `conjoint`, les deux valeurs sont mutuellement dépendantes :

$$d_h^o(o_2,\ o_3) = const_1 + F(d_h^o(o_1,\ o_4)) \tag{7.11}$$

$$d_h^o(o_1,\ o_4) = const_2 + F(d_h^o(o_2,\ o_3)) \tag{7.12}$$

Afin de calculer les valeurs respectives, un système d'équations est composé où, pour chaque valeur, une variable est introduite, soit x_1 et x_2. Le système d'équations devient :

$$x_1 \ = \ 0,18 \ + \ 0,4x_2 \tag{7.13}$$

$$x_2 \ = \ 0,14 \ + \ 0,4x_1 \tag{7.14}$$

Les solutions du système ci-dessus sont $x_1 = 0,28$ et $x_2 = 0,26$. La table 7.4 présente les résultats des quatre premiers pas de la méthode de résolution itérative sur les couples (o_1, o_4) et (o_2, o_3).

Les valeurs de d_h^o pour l'ensemble entier sont données dans la table 7.5. Celle-ci doit être lue comme suit : les entrées sous la diagonale principale représentent le résultat du calcul de d_h^o à l'aide des systèmes d'équations.

TAB. 7.4: *Premiers pas du calcul itératif de* $d^o_h(o_1,\ o_4)$ *et* $d^o_h(o_2,\ o_3)$

variable	pas 0	pas 1	pas 2	pas 3	pas 4	...
x_1	0,18	0,236	0,264	0,272	0,276	...
x_2	0,14	0,212	0,232	0,244	0,250	...

7.5.3 Comparaison entre mesures

Une manière d'étudier le comportement de notre mesure de dissimilarité est de la comparer à d'autres mesures qui ne possèdent pas ses caractéristiques. Le but de la comparaison est de mettre en évidence les différences entre la nouvelle mesure et une mesure standard. Ces différences peuvent se concevoir à plusieurs niveaux. Nous en retenons trois qui sont pertinents pour notre objectif, à savoir la construction de classes compactes.

Au plus *bas niveau*, les valeurs pour un jeu de données peuvent être directement comparées afin de dégager des tendances et des biais dans la manière dont la nouvelle mesure évalue les ressemblances entre objets. À un *second niveau*, on pourra comparer les structures classificatoires qui peuvent être construites à partir de ces valeurs. Des conclusions intéressantes peuvent être tirées sur la capacité de la mesure de conduire à la constitution de structures équilibrées (voir [McKusick et Langley, 1991] pour une discussion sur les critères de *bonne formation* pour les arbres de concepts). Au *niveau le plus haut*, les descriptions des classes que les mesures ont permis d'identifier peuvent être confrontées. Cette confrontation des résultats finaux du processus de construction de taxonomies permet de juger la réponse que la mesure apporte aux problèmes qui ont motivé sa définition.

Dans le cadre de l'exemple cité précédemment, une mesure adaptée à la comparaison est une fonction dont les valeurs ne prennent pas en compte la relation `conjoint`. Nous l'avons appelée $\hat{d}^o_h$. Cependant, un certain accord entre les deux fonctions est nécessaire pour que leurs résultats soient comparables. Cet accord s'exprime au niveau des contributions à la valeur globale de la dissimilarité des attributs considérés par les deux mesures. Or, les contributions sont contrôlées par les poids de chaque attribut. Ainsi, ils nous semble important que les deux fonctions préservent les mêmes proportions entre les poids des attributs communs.

Les valeurs de $\hat{d}^o_h$ sont calculées exclusivement sur les deux attributs propriétés, `salaire` et `âge`, pris à poids égaux (`0,5`). Dans la table 7.5, ces valeurs son placées au dessus de la diagonale et imprimées en *italique*.

Un examen plus approfondi des valeurs respectives de d^o_h et $\hat{d}^o_h$ met en évidence les phénomènes suivants. Tout d'abord, on pourra noter l'égalité des valeurs des deux fonctions sur les couples de conjoints (en gras). Ainsi, pour deux objets o et o' qui représentent un couple de conjoints ($o.conjoint = o'$ et $o'.conjoint = o$), on observe :

$$d^o_h(o, o') = \hat{d}^o_h(o, o')\ . \tag{7.15}$$

Ce fait indique une stabilité de la fonction d^o_h : sur les couples de conjoints, et donc liés l'un à l'autre, elle a la tendance de préserver les valeurs des dissimilarités calculées exclusivement sur les attributs. En effet, sur l'exemple on obtient les deux équations :

$$d^o_h(o_2,\ o_1) = const_1 + F(d^o_h(o_1,\ o_2)) \tag{7.16}$$

$$d^o_h(o_1,\ o_2) = const_2 + F(d^o_h(o_2,\ o_1)) \tag{7.17}$$

TAB. 7.5: *Valeurs des dissimilarités d^o_h et $\hat{d}^o_h$ pour les objets de table 7.3*

	o_1	o_2	o_3	o_4	o_5	o_6
o_1	0	***0,38***	*0,32*	*0,3*	*0,33*	*0,51*
o_2	**0,38**	0	*0,23*	*0,48*	*0,65*	*0,9*
o_3	0,36	0,26	0	***0,31***	*0,48*	*0,73*
o_4	0,28	0,43	**0,31**	0	*0,17*	*0,42*
o_5	0,5	0,61	0,46	0,33	0	***0,25***
o_6	0,56	0,74	0,57	0,44	**0,25**	0

Les deux valeurs $const_1$ et $const_2$ sont égales car elles sont calculées sur les mêmes valeurs d'attributs et selon des opérations qui commutent (les fonctions d'écart sont toutes symétriques). Par définition :

$$d^o_h(o_1,\ o_2) = d^o_h(o_2,\ o_1)$$

Le système est donc composé de deux équations identiques :

$$x = const + \lambda x \tag{7.18}$$

Observons aussi que la partie calculable de la dissimilarité $const$ s'exprime comme la moyenne des écarts sur les propriétés modulo la différence entre la somme totale des poids, un, et le poids de l'attribut circulaire, λ :

$$const = \frac{1-\lambda}{2}(\overline{\delta}_{age}(o_1, o_2) + \overline{\delta}_{sal}(o_1, o_2))$$

En conséquence, on obtient que la solution de l'équation est exactement la dissimilarité sur les propriétés prises aux poids égaux :

$$x = \frac{const}{1-\lambda} = \frac{\overline{\delta}_{age}(o_1, o_2) + \overline{\delta}_{sal}(o_1, o_2)}{2} \tag{7.19}$$

À noter que cette égalité entre la dissimilarité «locale» et la dissimilarité sur tout attribut, y compris les relations induisant des circuits, n'est valable que dans le seul cas de poids égaux dans $\hat{d}^o$. Toutefois, pour qu'elle soit vérifiée, il est nécessaire de choisir ces poids de manière cohérente dans les deux fonctions.

Dans le cas de poids inégaux et d'un nombre arbitraire d'attributs qui ne participent pas à des CFC, si un seul attribut est circulaire, cette condition peut être formulée comme suit. Soit un couple d'objets o, o' et sa variable correspondante x_t. Si o, o' sont des instances du concept k_t, alors la dissimilarité $d^o_{k_t}(o, o')$ s'écrit comme suit :

$$d^o_{k_t}(o, o') = \sum^{a \in A_t} \lambda_a * \delta^f_a(o.a, o'.a) \tag{7.20}$$

Soit a_r l'attribut relationnel introduisant une circularité et dont le co-domaine est k_t. Une dissimilarité $\hat{d}^o_{k_t}$ qui est directement calculable, c'est-à-dire sans recours à un système d'équations, et comparable avec $d^o_{k_t}(o, o')$ sur les couples mutuellement liés, peut être définie sur l'ensemble des attributs $A - \{a_r\}$. Pour cela, il suffit de choisir, pour chaque attribut $a \neq a_r$, un poids $\underline{\lambda}_a$ tel que :

$$\underline{\lambda}_a = \frac{\lambda_a}{1 - \lambda_{a_r}} \tag{7.21}$$

Le même type de raisonnement peut être appliqué dans le cas où plusieurs attributs participent à une CFC, c'est-à-dire que $|Circ(o, o')| > 1$. Toutefois, la recherche d'égalité entre les deux fonctions n'a de sens que si tous ces attributs lient les deux objets o, o' dans un sens comme dans l'autre. En particulier, il est inutile de chercher une généralisation dans le cas d'attributs dont le co-domaine n'est pas k_t. Cependant, l'absence de « points de recoupement » n'empêche pas la comparaison entre une fonction de type $d^o_{k_t}$ et une autre fonction $\hat{d}^o_{k_t}$ définie selon le même principe que ci-dessus. Une telle comparaison a pour but de mettre en évidence le gain obtenu par l'utilisation de l'information complète par rapport à une comparaison basée sur le sous-ensemble d'attributs directement manipulables par les fonctions d'écart.

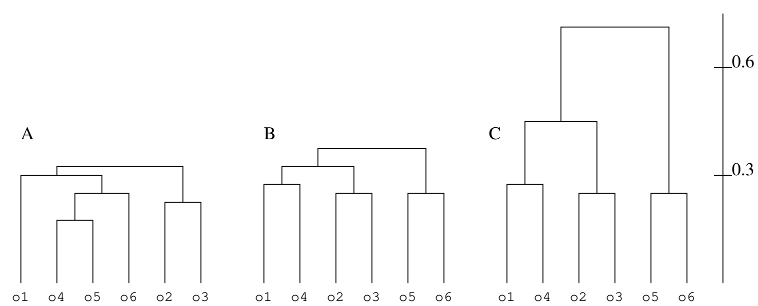


FIG. 7.5: *Trois dendrogrammes obtenus par classification hiérarchique sur les objets de l'exemple. À gauche (A.) est dessiné le résultat d'une méthode de type lien simple sur la matrice produite avec la fonction $\hat{d}^o_h$. Au centre (B.) le résultat de la même méthode sur les données de d^o_h est dessiné. À droite (C.) le résultat d'une méthode de lien complet avec d^o_h est visualisé. Le relatif accord entre les dendrogrammes B. et C. en ce qui concerne la structure des classes est, en grande partie, déterminé par la formation des trois groupes de deux objets au premier niveau.*

Si la fonction d^o_h préserve les valeurs de la dissimilarité « locale » $\hat{d}^o_h$ sur les couples de conjoints, elle diverge systématiquement de cette dernière sur le reste des couples de l'exemple ci-dessus. Cela s'explique par l'influence mutuelle entre la valeur de d^o_h pour un couple d'objets et la valeur sur le couple des conjoints respectifs. Globalement, on peut noter une tendance à l'attraction mutuelle qui traduit cette influence : les valeurs plus fortes pour $\hat{d}^o_h$ diminuent, alors que les plus faibles augmentent. Par exemple, $\hat{d}^o_h(o_4, o_5)$ est très faible, $0,17$, alors que la valeur sur les conjoints respectifs est relativement forte ($\hat{d}^o_h(o_3, o_6) = 0,73$). Lors du passage à la dissimilarité relationnelle, la différence entre les valeurs respectives de d^o_h a tendance à décroître ($0,33$ dans le premier cas et $0,57$ dans le second). Ce phénomène peut être observé sur tous les couples d'objets en dehors de ceux qui représentent des couples d'époux.

Soulignons que les différences entre les deux fonctions ne sont pas de simples modifications de valeurs qui, admettons-le, n'ont pas un sens intrinsèque. En effet, le rôle de la dissimilarité n'est pas tant de quantifier de manière exacte les ressemblances entre deux objets que de permettre le tri des couples d'objets pour en regrouper les plus similaires. C'est pourquoi, plus qu'à des quantités effectives, il est nécessaire de s'intéresser à des éventuels changements d'ordre entre les couples lors du passage de $\hat{d}^o_h$ à d^o_h. Sur l'exemple ci-dessus, de tels changements sont nombreux, ce qui témoigne d'une réelle restructuration dans la topologie de l'espace induite par d^o_h par rapport à celui de $\hat{d}^o_h$. Pour illustrer ce fait, prenons l'exemple du couple o_4,o_5 qui est le plus frappant : du rang un selon $\hat{d}^o_h$ celui-ci passe au rang cinq selon d^o_h.

7.5.4 Confrontation des deux taxonomies obtenues

L'effet global de la prise en compte des relations conjugales peut être estimé en comparant les groupes construits par une méthode hiérarchique à partir de d^o_h et $\hat{d}^o_h$ (voir figure 7.5). Observons, par exemple, l'évolution du regroupement des objets o_1 et o_4 : dans un groupe avec o_5 et o_6 par rapport à $\hat{d}^o_h$ (figure 7.5.A), mais séparés de ceux-ci lorsque d^o_h est utilisée (figure 7.5.B et C).

Le gain de l'utilisation de la mesure n'est apparent que lorsqu'on observe les classes que celle-ci permet de construire. Celles-ci sont obtenues, selon notre méthode de construction décrite dans le chapitre 3, à partir des groupes construits automatiquement. Ainsi, en reprenant l'exemple ci-dessus, le groupe contenant o_1, o_4 sur la figure 7.5.B va donner lieu à la classe `cl#1` (voir figure 7.6). De même, le groupe de o_2 et o_3 sera transformé en la classe `cl#2`. Le groupe commun de ces quatre objets formera la classe `cl#3`, la super-classe de `cl#1` et `cl#2`.

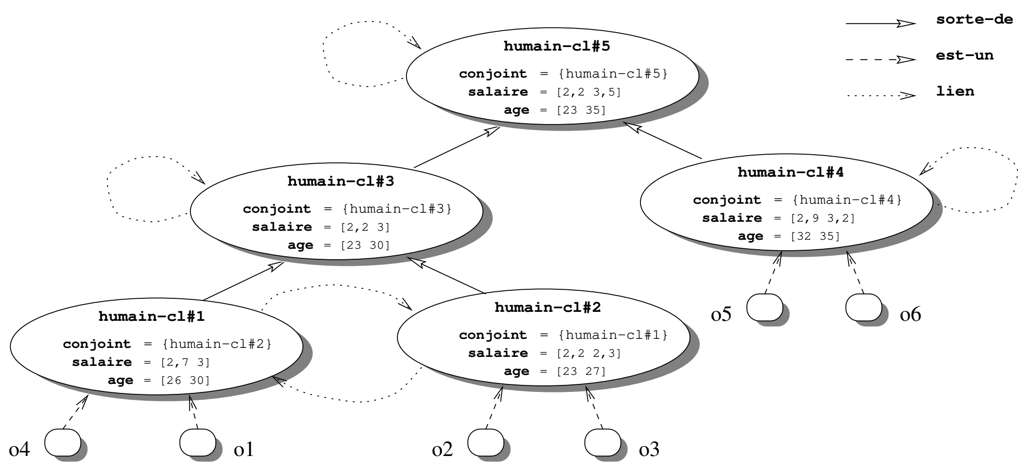


FIG. 7.6: *Les classes obtenues par caractérisation à partir des groupes découverts par le regroupement hiérarchique ascendant selon la mesure d^o.*

L'opération de création de classes est effectuée de manière descendante : les classes les plus générales sont créées d'abord, en tant que sous-classes directes de la classe racine. Les classes intermédiaires sont créées en respectant l'ordre de l'inclusion, ainsi, lors de la création d'une classe, sa super-classe directe est déjà connue. Les classes les plus spécifiques, c'est-à-dire les feuilles de la taxonomie, sont créées en dernier. Finalement, les objets sont attachés à leurs classes respectives une fois pour toute. Ainsi, o_1, o_4 ne seront déclarés comme des membres de la classe `cl#2` que lorsque la totalité des classes seront créées.

Après l'exécution de cette étape de création de classes sur l'ensemble des concepts d'une CFC du graphe des dépendances, les classes peuvent être décrites selon *tous* leurs attributs. Conformément aux principes définis dans la section 3.5, pour chaque attribut d'une classe, nous considérons l'ensemble des valeurs que l'attribut prend dans les instances de la classe. La contrainte sur l'attribut dans la classe est synthétisée comme la description la plus spécifique que cet ensemble admet. Elle est choisie parmi les descriptions admissibles que le type de l'attribut offre au moment de l'opération (voir chapitre 4).

Ainsi, pour caractériser `cl#1` selon `conjoint`, on prendra la classe la plus spécifique qui contient les objets référencés par o_1 et o_4, c'est-à-dire o_2 et o_3, donc `cl#2`. Inversement, dans `cl#2` le domaine de `conjoint` sera `cl#1`, ce qui mène à une co-définition.

Des classes peuvent être obtenues à partir des groupes selon $\hat{d}^o_h$ (voir figure 7.6). En effet,

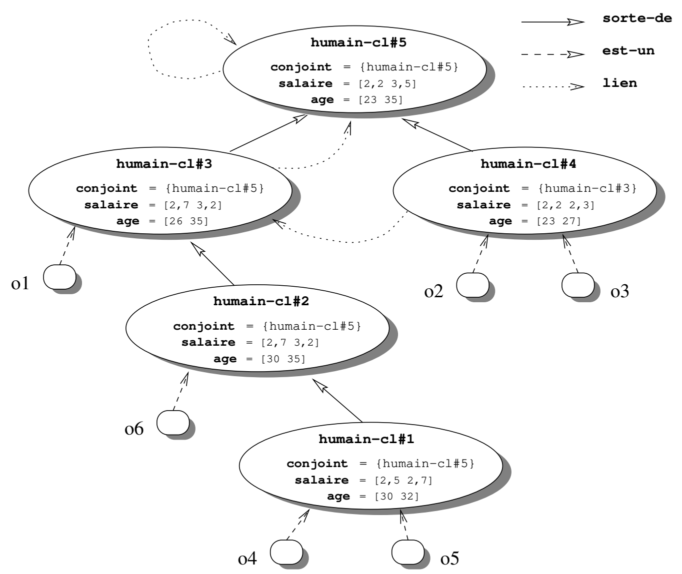


FIG. 7.7: *Les classes obtenues par caractérisation à partir des groupes découverts par le regroupement hiérarchique ascendant selon la mesure $\hat{d}_h^o$. Les liens entre classes que celles-ci peuvent hériter ne sont pas dessinés pour des raisons de simplicité.*

en suivant la même procédure, on obtient des descriptions complètes des classes sur tous les attributs, y compris les relations introduisant une co-définition. Toutefois, ces dernières classes ont des descriptions qui sont globalement moins compactes et elles ne révèlent aucune structure particulière sur l'attribut `conjoint`. Par exemple, la classe `cl#1` de la figure 7.6 a pour valeur de cet attribut la classe `cl#5` qui est la classe la plus générale.

En somme, grâce à la nouvelle mesure, il est possible de détecter des régularités dans les descriptions qui portent sur tous les attributs indépendamment de leur nature et les sortes de dépendances qu'ils induisent entre les objets.

Dans la suite, nous allons décrire des procédures pour calculer la dissimilarité de la manière la plus efficace.

7.6 Aspects calculatoires

Dans cette section, nous traitons quelques aspects liés au calcul effectif des structures décrites précédemment.

7.6.1 Calcul direct par couples

Tout d'abord, la fonction d^o a été définie par référence aux réseaux associés à un couple d'objets. Logiquement, son calcul exigerait la délimitation des réseaux pour chaque couple d'objets, ainsi que leur mise en correspondance afin d'en extraire les couples en co-définition. Le principe du calcul pour un concept k est esquissé par l'algorithme 10.

Afin d'obtenir une matrice de dissimilarité sur chaque concept, l'algorithme doit être exécuté autant de fois qu'il y a de concepts. Il faut remarquer que les couples d'objets d'un concept peuvent être liés sans être en co-définition. Lors du traitement du concept, beaucoup de couples

DISSIM-CONCEPT(k)

- entrée : k - un concept

pour tout couple d'objets (o, o') dans k **faire**
    trouver les réseaux associés $R(o)$ et $R(o')$
    calculer $Couples(o, o')$ en appariant $R(o)$ et $R(o')$
    composer le système d'équations
    résoudre le système

Algorithme 10: Calcul de dissimilarité pour un concept k

seront donc examinés plusieurs fois, alors qu'ils ont déjà été munis d'une valeur pour leur dissimilarité. Il y aura donc un nombre important de vérifications à faire.

7.6.2 Calcul global

Une idée plus astucieuse peut être dérivée de notre stratégie de construction multiple de classes. Elle consiste à conduire un seul calcul global, intégrant tous les couples d'objets d'un concept d'une CFC du graphe des dépendances, soit $\mathcal{C} = \{k_1, k_2, ..., k_p\}$. Cela revient à résoudre un seul et unique système d'équations dont les variables sont les couples d'objets du même concept au sein de la CFC. Cet ensemble est noté $Couples^{CFC}(\mathcal{C})$. À première vue, ce procédé semble augmenter la complexité car le calcul d'un pas devra inclure un parcours de toutes les variables et donc de tout couple d'objets. Manifestement, pour un couple donné on n'a besoin, dans le cas maximal, que d'un sous-ensemble très restreint de l'ensemble total, c'est-à-dire $|Couples(o, o')| \ll |Couples^{CFC}(\mathcal{C})|$. La composition et la résolution d'un système séparé par couple semblent donc plus économiques. Cependant, sur un plan global, il n'y a pas de calculs *inutiles* dans un cas, comme dans l'autre. En effet, toutes les valeurs obtenues par la résolution du système global sont nécessaires pour l'étape de regroupement. De plus, le calcul effectif des solutions de ce système pourra se faire de sorte que la complexité n'augmente pas par rapport à la résolution séparée. Tout se passe donc comme si l'on résout plusieurs systèmes en parallèle. Finalement, l'intérêt de la résolution globale réside dans le fait qu'elle ne nécessite plus la recherche des réseaux associés aux objets et leur appariement. Le gain est substantiel car l'opération d'appariement entre deux réseaux est hautement complexe, surtout dans un cas de multiples connections entre objets par des relations. Son coût, pour un couple d'objets o et o' a été évalué en $O((n_o)^3)$ où n_o est le nombre moyen d'objets sur la CFC du réseau d'un objet du concept k. Des algorithmes de calcul pour l'ensemble $Couples(o, o')$, ainsi qu'une analyse de sa complexité, peuvent être trouvés dans l'appendice B.

Le procédé global peut être résumé ainsi :

1. composition de tous les couples (o, o') d'un concept k de la CFC ;
2. pour tout couple calculer et mémoriser la dissimilarité locale (dans un vecteur) ;
3. pour tout couple, calculer les coefficients de l'équation caractérisant celui-ci (dans un tableau carré) ;
4. résoudre le système représenté par le tableau par une méthode itérative (de précision ε).

Dans la suite, une réalisation de cette technique est proposée. Pour cette réalisation, nous avons choisi la variante qui applique un calcul unique et préalable de tous les appariements entre collections. Cette variante est de loin plus économique, bien que potentiellement moins précise. Toutefois, dans un grand nombre des cas elle aboutira à des résultats exacts[31].

7.6.3 Description de l'algorithme

Un algorithme en pseudo-code réalisant la stratégie de la section précédente est donné ci-dessous (voir algorithme 11). Il effectue un traitement de la CFC $\mathcal{C}$ donnée en entrée en quatre étapes.

Au début, pour chaque concept k tous les couples d'objets sont calculés (appel à la primitive `CALCULER-COUPLES()`) L'ensemble des couples sur la CFC est stocké dans un vecteur ($Couples[]$).

Les deux étapes suivantes concernent l'initialisation du système d'équations représenté par un tableau $n \times (n+1)$ où n est le nombre des couples ($dep[][]$ ci-dessous). Tout d'abord, la partie «locale» de la dissimilarité de chaque couple d'objets est calculée. Celle-ci est composée par les dissimilarités sur tous les attributs du couple dont le type n'est pas un concept de la CFC en cours de traitement (primitive $DISSIM(o, o', a)$). À cette valeur s'ajoutent toutes les parties constantes de la dissimilarité sur un attribut multivalué dont le type est un concept de la CFC. Rappelons que la partie constante de la dissimilarité entre deux collections reflète leur différence de cardinalité. Elle s'élève, pour deux collections de cardinalité p et q à

$$1 - \frac{\mathbf{min}(p, q)}{\mathbf{max}(p, q)}.$$

Cette valeur ne dépend pas d'un appariement particulier, ni des dissimilarités entre les membres des collections. Elle peut donc être calculée auparavant. La somme de toutes les dissimilarités ainsi calculée est affectée au membre de degré zéro, noté b_t ci dessus (voir section 7.3.1) qui est stocké dans un vecteur ($loc[t]$).

Ensuite, lors d'un second passage sur l'ensemble des couples, les autres coefficients de l'équation caractérisant celui-ci sont établis. Ceux-ci sont stockés dans une table carrée, $dep[][]$. Ainsi, pour chaque attribut a dont le type est dans $\mathcal{C}$, le poids de l'attribut est cumulé dans la colonne dont l'indice correspond au couple référencé, $(o.a, o'.a)$ ($dep[i][j]$).

Finalement, lorsque le système a été initialisé, il peut être résolu par une méthode itérative (avec un seuil de précision ε). Pour cela, il faut corriger, par un pas préliminaire, les coefficients du tableau avec les poids des attributs qui bouclent sur le couple d'objets ($Boucle(o, o')$). Ainsi, pour chaque ligne t, les coefficients sont divisés par le complément à un de l'élément sur la diagonale, c'est-à-dire par $1 - dep[t][t]$. La résolution est ensuite faite pas à pas, à chaque pas les inconnues x_t étant substituées dans les équations par leurs valeurs obtenues au pas précédent (sauf le pas 0). Celle-ci s'arrête lorsque toutes les différences entre deux valeurs successives pour une inconnue deviennent inférieures à ε.

7.6.4 Primitives

Les primitives utilisées sont les suivantes :

- `CALCULER-COUPLES`(k) calcule l'ensemble de tous les couples du concept,
- $index(o, o')$ donne l'index de (o, o') dans le vecteur de la correspondance couple – variable,

31. Seule une étude approfondie de la question, avec un nombre suffisant de tests comparatifs permettra de valider ce choix.

– `APPARIEMENT-OPT`$(o.a, o'.a, loc[])$ fournit un appariement optimal entre les collections dans les deux premiers arguments ; l'optimalité est prise par rapport à la dissimilarité donnée sous forme de matrice par le troisième argument,

– & designe la concaténation de vecteurs.

Les variables locales utilisées sont comme suit :

– $Couples[]$ un vecteur de couples d'objets,

– $dep[][]$ le tableau qui décrit les poids des liens entre les couples d'objets, les coefficients du système d'équations sont calculés à partir de celui-ci,

– $loc[]$ le vecteur qui contient les parties de la dissimilarité calculable avant la résolution du système d'équations,

– $Variables[]$ le vecteur qui établit la correspondance entre un couple et la variable dans le système d'équations.

L'algorithme 12 montre le procédé de résolution du système.

Nous donnons également l'algorithme de résolution du système d'équations, algorithme 12. Celui-ci consiste à effectuer un certain nombre de pas itératifs de calcul de $\mathbf{F}$ à partir de $\mathbf{0}$, initialement contenu dans le vecteur D. La condition d'arrêt est donnée par un seuil ε : elle est remplie lorsque toutes les différences entre les valeurs respectives pour deux pas consécutifs du calcul sont inférieures au seuil.

7.6.5 Complexité

Il est intéressant de pouvoir évaluer, bien que de manière très approximative, le coût du traitement des circularités. Soient :

– n_o le nombre maximal d'objets d'un concept k,

– n_k le nombre des concepts à considérer dans une CFC,

– n_a le nombre maximal d'attributs dans un concept,

– n_c le nombre maximal d'attributs dans un ensemble $Circ(o, o')$,

– n_m le nombre maximal d'attributs multivalués dans un ensemble $Circ(o, o')$,

– c_m le nombre maximal des objets dans une collection.

Dans la liste ci-dessus, le facteur prépondérant est le nombre des objets, n_o. En effet, il est largement supérieur au nombre de concepts ($n_o \gg n_k$), qui ne dépasse pas d'habitude quelques dizaines. Il en va de même pour les trois quantités liées aux attributs, n_a, n_c et n_m, elles sont bornées par des valeurs qui sont fixes pour une application donnée. La seule quantité qui puisse croître, au moins en théorie, autant que n_o est c_m. Dans certains domaines la cardinalité des collections peut même dépasser le nombre des objets dans la base, essentiellement grâce aux répétitions dans les collections (séquences génétiques, suites temporelles, etc.). Il faut tout de même remarquer que les éléments de ces collections sont rarement implantés sous forme d'objets pour raisons évidentes d'éfficacité. Dans une application moyenne, la cardinalité maximale des collections est bornée par quelques dizaines (par exemple les collections de satellites naturels des planètes du système solaire).

```
CALCULER-DISSIM(C,ε)

- entrées : C = {k1, k2, ..., kp} un ensemble de concepts formant une CFC du graphe
-           : ε - seuil de précision pour le calcul de d^o
- variables : Variables[] - un vecteur de couples d'objets
-           : dep[][] - un tableau de réels
-           : loc[] - un vecteur de réels
-           : n - un entier

{============= initialisation =============}
Variables[] : = ∅
pour tout concept k de C faire
  Variables[] : = Variables[] & CALCULER-COUPLES(k)
n : = longueur(Variables[])
pour i : = 1 à n faire
  loc[i] : = 0
  pour j : = 1 à n faire
    dep[i][j] : = 0
{============= calcul des b_t =============}
pour tout (o, o') ∈ Variables[] faire
  i : = index(o, o')
  pour tout a ∈ attributs(attributs(o)) faire
    si (nature(a) ≠ lien) ou (type(a) ∉ C) alors
      loc[i] : = loc[i] + DISSIM(o, o', a)*poids(a)
    sinon
      si cons(a) ≠ un alors
        p : = |o.a| ; q : = |o'.a|
        loc[i] : = loc[i] + 1 − min(p, q)/max(p, q)*poids(a)
{============= calcul des c_i,j =============}
pour tout (o, o') ∈ Variables[] faire
  i : = index(o, o')
  pour tout a ∈ attributs(attributs(o)) faire
    si (nature(a) = lien) et (type(a) ∈ C) alors
      si cons(a) = un alors
        j : = index(o.a, o'.a)
        dep[i, j] : = dep[i, j] + poids(a)
      sinon
        {============= cas multivalué =============}
        p : = |o.a| ; q : = |o'.a|
        M_opt : = APPARIEMENT-OPT(o.a, o'.a, loc[])
        pour tout (o_i, o'_j) ∈ M_opt faire
          j : = index(o_i, o'_j)
          dep[i, j] : = dep[i, j] + poids(a)/max(p, q)
RESOLUTION-SE(dep[][],loc[],ε)
```

Algorithme 11: Calcul de la dissimilarité sur une CFC $\mathcal{C}$

```
RESOLUTION-SE(dep[][],loc[],ε,n)

- entrées : dep[][] - un tableau de réels
-           : loc[] - un vecteur de réels
-           : ε - seuil de précision
-           : n - le nombre de couples
- sorties   : D[] - un vecteur de réels, les solutions
- variables : D_N[] - un vecteur de réels
-           : liens[] - un vecteur d'ensembles d'entiers
-           : arret - un booléen

 {============= initialisation =============}
pour i := 1 à n faire
   D[i] := 0
   si dep[i,i] ≠ 0 alors
      pour j := 1 à n faire
         dep[i,j] := dep[i,j]/(1 − dep[i,i])
      loc[i] := loc[i]/(1 − dep[i,i])
      dep[i,i] := 0
      liens[i] := ∅
      pour j := 1 à n faire
         si dep[i,j] ≠ 0 alors
            liens[i] := liens[i] ∪ {j}
 {============= boucle principale =============}
repeter
   pour i := 1 à n faire
      D_N[i] := loc[i]
      pour tout j ∈ liens[i] faire
         D_N[i] := D_N[i] + dep[i,j] ∗ D[j]
   arret := vrai
   pour i := 1 à n faire
      arret := arret et (D_N[i] − D[i] < ε)
   D := D_N
jusqu'a arret
```

Algorithme 12: Résolution du SE

Algorithme principal

La complexité du pas d'initialisation est quadratique en fonction du nombre d'objets dans chaque concept. Le nombre des concepts y intervient en facteur multiplicatif, on a donc une complexité de $O(n_k * n_o^2)$.

La complexité de la seconde étape, c'est-à-dire le calcul de la dissimilarité «locale», est une fonction du nombre des couples et du nombre de leurs attributs car on effectue essentiellement une seule opération par attribut. En effet, le coût de l'appel de `DISSIM()` est estimé constant car la valeur recherchée peut être calculée au préalable et stockée dans une matrice associée au concept/type de données sous-jacent. En conséquence, la complexité est de l'ordre de $O(n_k * n_o^2 * n_a)$.

Finalement, la troisième étape est légèrement plus coûteuse que la seconde car on calcule de plus un appariement. Sa complexité est de l'ordre de $O(n_k * n_o^2 * (n_a + n_m * c_m^3))$.

L'algorithme a donc une complexité en temps de $O(n_k * n_o^2 * (n_a + n_m * c_m^3))$. Même si ceci peut sembler trop, il faut remarquer que les facteurs liés à la structure des objets : nombre d'attributs d'une certaine sorte, cardinalité des collections, nombre de concepts dans une CFC, ne dépendent pas du nombre des objets. Il peuvent être bornés donc par des constantes propres à un domaine. Ainsi, la complexité se réduit en une fonction quadratique de n_o.

Résolution du système d'équations

Soit i le nombre des pas effectués par l'algorithme itératif. La complexité du pas d'initialisation est quadratique en fonction du nombre de couples à considérer, c'est-à-dire $O(n_k^2 * n_o^4)$. Cette étape est la plus coûteuse de tout l'algorithme. Cependant, elle peut être accellérée par diverses techniques de représentation compacte de la matrice qui est extrêment creuse. Par manque de temps, ces techniques n'ont pas été examinées.

La seconde étape, le calcul itératif a une complexité asymptotyque de $O(n_k * n_o^2 * n_c * i)$. En sachant que les calculs convergent relativement rapidement (maximum une dizaine de pas) et que n_c est indépendant de n_o, nous pouvons conclure que la complexité est quasi-quadratique.

Complexité totale

Comme nous l'avons souligné, le seul facteur de réelle importance est n_o et dans certains cas rares, c_m doit aussi être pris en compte. Ainsi, la complexité globale du calcul de la dissimilarité sur une CFC de concepts est de l'ordre de :

$$O(n_k * n_o^2 * (n_a + n_m * c_m^3 + n_k * n_o^2 + n_c * i)).$$

En simplifiant les facteurs qui sont bornés, on arrive à une complexité qui est fonction de degré quatre du nombre des objets :

$$O(n_o^4).$$

7.7 Discussion

Confrontés à l'impératif de traiter les descriptions des objets co-définis sous leur forme originale, nous avons pu constater l'inadaptation des approches de regroupement basées sur des critères de nature globale pour ce type fortement structuré d'individus. Nous pouvons le dire maintenant, un traitement complet et cohérent nécessite la définition des opérateurs de point-fixe sur les outils de base employés par une méthode de construction.

À notre avis, l'introduction d'un tel opérateur sera difficilement conciliable avec l'appareil statistique sur lequel l'approche probabiliste est fondée. En revanche, la bonne formalisation de l'approche par treillis laisse supposer une introduction possible de tels opérateurs dans la méthode. Une question à se poser impérativement concerne l'influence mutuelle entre les classes construites de manière simultanée. En effet, si la construction simultanée peut être utile, elle est certainement une source de difficultés nouvelles pour l'interprétation des résultats du regroupement.

Une autre question qui demeure ouverte est l'interaction avec l'utilisateur. Si pour une méthode classique de regroupement il est encore possible, bien que délicat, de fournir une aide, voire un conseil à l'utilisateur, avec une construction circulaire le défi est encore plus ambitieux. En effet, l'aide est, en général, basée sur un lien établi entre les entrées et les sorties du processus. Ce lien permet de suggérer des modifications dans la manière dont le problème est posé pour mieux adapter la construction aux objectifs de son utilisateur. Dans le cas circulaire, un tel lien est extrêmement délicat à obtenir, entre autre à cause des influences mutuelles entre objets et classes.

Cependant, nous ne considérons pas cette difficulté comme rédhibitoire, elle n'est en aucun cas propre à notre approche. Au contraire, nous pensons que toute approche qui s'attache à un traitement direct des circularités va se heurter aux mêmes difficultés.

Il en va de même pour la complexité algorithmique de la construction. Celle-ci, bien que polynomiale, reste assez élevée. La forte complexité est le prix à payer lorsqe l'on s'offre le luxe de traiter directement les structures complexes, circulaires ou non. Même si certains résultats encourageants ont été avancés récemment [Augier *et al.*, 1995; Zucker et Ganascia, 1996], il n'y a pas encore d'évidence que les formalismes relationnels peuvent être analysés, dans toute leur généralité, de manière efficace.

Dans notre cas, la complexité est concentrée sur l'étape de calcul de la dissimilarité. Cela permet de varier les taxonomies finales en appliquant des algorithmes de regroupement automatique différents sans pour autant multiplier le coût par le nombre des exécutions.

Un tel gain semble difficilement réalisable avec une méthode de construction incrémentale et probabiliste.

Finalement, lorsqu'il s'agit d'aider la conception d'une taxonomie pour une application à objets, la question du coût algorithmique ne doit pas être éxagérée. En effet, le processus de création d'une application à objets est long et le fait de devoir attendre pour un résultat de regroupement n'est pas si handicapant.

Pour résumer, la variante optimale semble être de fournir un outil qui s'adapte aux besoins de son utilisateur en offrant un compromis variable entre précision du résultat et coût. L'outil doit être suffisamment complet pour pouvoir fournir une aide dans le plus grand nombre de situations. D'autre part, une certaine flexibilité permettra à l'utilisateur d'accorder une prépondérance plus ou moins forte au facteur temps. C'est vers cette solution que nous nous sommes dirigés avec la composition du module T-TREE.

7.8 Conclusion

Un système de regroupement d'individus structurés, s'il se veut complet, doit pouvoir traiter les relations entre individus de manière directe, y compris les relations circulaires. Celles-ci se présentent de manière naturelle dans les données provenant de domaines fort variables. Leur prise en compte peut conduire à la découverte de groupes d'individus qu'il est difficile de construire autrement.

La technique de calcul présentée dans ce chapitre permet d'appliquer *directement* la mesure de dissimilarité entre objets définie au chapitre 5 sur des objets co-définis. Elle consiste à poser et résoudre un système d'équations dont les solutions sont les valeurs de la dissimilarité d^o. L'utilisation de la mesure s'intègre dans une stratégie de construction automatique de classes en deux étapes : regroupement puis caractérisation. Ainsi, il devient possible de construire des classes qui sont en co-définition, c'est-à-dire que les classes découvertes automatiquement sont décrites en termes d'autres classes qui sont elles aussi résultat du même processus de construction.

En somme, la mesure complète permet non seulement de trouver des regroupements nouveaux, mais aussi de construire des classes dont les descriptions sont compactes sur toutes les relations. Ainsi, elle permet de répondre aux exigences de **complétude** et de **flexibilité**. Son principe de calcul, la recherche de point fixe d'une fonction numérique, la rend **cohérente** car il correspond parfaitement à la sémantique qui est accordée aux circularités dans les formalismes de représentation structurés.

Ce chapitre clôt la partie du mémoire réservée à la présentation des outils mathématiques nécessaires à la construction de taxonomies sur les objets d'une base. Nous allons maintenant nous tourner vers la partie dédiée à la mise en œuvre effective de l'approche au sein d'un SRCO.

Troisième partie

Implantation et exemples

8

Construction de taxonomies au sein de TROEPS

Introduction

Dans la première partie de ce mémoire, nous avons présenté le cadre de notre travail avec la problématique abordée ainsi que l'état d'avancement sur les problèmes spécifiques que celle-ci comprend. Dans la seconde partie, il a été question des solutions envisagées pour chacun de ces problèmes. Nous avons présenté une stratégie de construction de plusieurs taxonomies reliées, qui optimise les applications successives d'une méthode de construction effective dans un cadre multi-concepts. La méthode s'appuie, à son tour, sur deux outils : une mesure de dissimilarité adaptée aux objets et un opérateur de généralisation sur les objets.

La troisième partie du document propose une vue d'ensemble à travers de la mise en œuvre des techniques employées dans la construction de taxonomies. Dans ce chapitre, nous montrons l'enchaînement des diverses étapes, à partir du début, l'indication d'un ensemble d'objets (un concept) par l'utilisateur, jusqu'à l'obtention de la taxonomie définitive et son intégration au sein de la base de connaissances.

Les techniques ont été développées au sein du système T-TREE (TROEPS Taxonomy building Tool [Euzenat, 1993a]). À l'origine, le système comportait la réalisation de quelques algorithmes d'analyse de données dans leurs versions classiques, c'est-à-dire n'admettant que des caractéristiques numériques dans la description des objets. À ce système nous avons ajouté une mesure de dissimilarité générique (dans la version 1.1 de TROEPS) puis l'ensemble des techniques présentées ici (version 1.3, 1999).

Au début de ce chapitre, nous précisons les objectifs de la création du système (section 8.1). La section 8.2 présente le cycle global de construction d'une taxonomie dans T-TREE, permettant ainsi de replacer chacun des éléments de notre approche dans le contexte. L'architecture du système organisée en un ensemble de modules est présentée dans la section 8.3. Une attention particulière est réservée à l'interface graphique de T-TREE (section 8.4) qui a sa propre contribution dans la satisfaction des exigences de flexibilité et de paramétrisation du processus de construction. Finalement, la section 8.5 présente quelques fonctionnalités annexes de T-TREE qui ne relèvent pas directement de la construction de taxonomies mais peuvent néanmoins trouver leur place dans le processus global de conception et intégration d'une taxonomie au sein de la base de connaissances.

Le chapitre suivant présente un exemple de construction conjointe de taxonomies au sein d'une base de connaissances à objets gérée par TROEPS.

8.1 Objectifs de T-TREE

Le système T-TREE est un module de construction de taxonomies de classes intégré à TROEPS. C'est un atelier composé de plusieurs outils destinés à aider l'utilisateur dans la tâche de la conception de la ou des taxonomies d'une base de connaissances à objets.

La création du module T-TREE devait satisfaire plusieurs objectifs. D'une part, le but est d'offrir une assistance à l'utilisateur de la manière la plus **complète**, la plus **flexible** et la plus **efficace** possible. Pour cela, un ensemble d'outils satisfaisant ces exigences sont conçus, leur réalisation s'appuyant sur les résultats exposés dans la partie précédente de ce mémoire.

D'autre part, des fonctionnalités permettant l'exploitation des résultats de l'analyse sont proposées. Il s'agit tout d'abord de la confrontation de plusieurs taxonomies issues de méthodes différentes de construction, ainsi que définies de manière manuelle. De plus, la cohérence entre les classes construites automatiquement et les classes existantes au préalable est assurée.

Un troisième objectif poursuivi, dans la même direction de la plus grande flexibilité, est d'offrir réellement une extensibilité de l'environnement de construction de taxonomies. Cette extensibilité ne se limite pas aux moyens de description des objets, mais porte également sur l'ensemble des outils de construction disponibles.

Parmi les trois groupes seuls le premier et le troisième sont pertinent pour le travail exposé, le reste pouvant être atteint par des outils qui ont été développés par ailleurs[32]. Dans la suite, nous allons montrer comment ces objectifs ont été atteints.

8.2 Cycle de construction d'une taxonomie

Une construction de taxonomie dans TROEPS à l'aide de T-TREE comporte plusieurs étapes de traitement d'informations de nature variée. Tout d'abord, elle peut nécessiter la construction préalable d'autres taxonomies sur des familles d'objets liées. Ensuite, chaque construction se décompose en étapes successives qui transforment progressivement un ensemble d'objets en une taxonomie de classes : calcul de dissimilarité, regroupement des objets selon leurs dissimilarités, création des classes et inférence de descriptions. De plus, lorsqu'une co-définition se présente dans les familles d'objets à regrouper, d'autres étapes s'ajoutent à ce processus classique de regroupement. Au début, les composantes fortement connexes (CFC) du graphe des dépendances sont formées afin de découper la tâche globale de la construction de taxonomies en sous-tâches minimales. Cette étape est suivie d'une étape de calcul de dissimilarité qui peut conduire, si des relations circulaires existent au sein de la CFC, à la composition et la résolution d'un système d'équations. Dans ce qui suit, nous précisons le contenu de chacune des étapes ainsi que leur enchaînement.

8.2.1 Situation de départ

Lorsqu'on s'adresse à la construction automatique de classes, on se trouve typiquement dans une situation où un concept vu sous un point de vue spécifique ne possède pas une taxonomie satisfaisante. Le but est donc d'en construire une en partant de l'ensemble des instances du concept. Les données du processus sont donc :

- un concept k,
- un point de vue pv sur k,

32. C'est le cas de la vérification de l'adéquation entre les nouvelles classes et les classes déjà existant.

- une classe c sur k qui appartient à pv, pour des raisons de simplicité c est considéré comme étant la racine de pv, ce qui n'est pas nécessairement le cas en général, de cette façon, le concept k et la classe c peuvent être confondus,
- un ensemble d'objets O à répartir en classes, O est l'ensemble des objets qui sont rattachés à c,
- un ensemble A d'attributs à considérer pour la construction de classes, un sous-ensemble A^r de A est composé par les attributs relationnels, A est l'ensemble des attributs de la classe c,
- un poids λ_a pour chaque attribut a dans A,
- un ou plusieurs points de vue pv_a^i pour chaque attribut a dans A^r, pv_a^i offre une perspective sur le concept qui sert à typer a, $k_a = cod(a)$; pour des raisons de simplicité, nous prenons l'hypothèse qu'un seul point de vue est spécifié pour chaque a, celui-ci est nécessaire pour le calcul de la dissimilarité sur O lorsque les valeurs de a dans les objets de O (instances de k) sont comparées moyennant un écart de classes δ^{cl} ; dans le cas d'une comparaison par une dissimilarité d'objets, d^o, pv_a^i sert essentiellement à accueillir les nouvelles classes sur k_a ; dans ce cas, exactement un point de vue, pv_a, est spécifié.

Les éléments énumérés ci-dessus sont choisis en plusieurs étapes à l'aide de l'interface graphique de TROEPS/T-TREE. Les détails sur la procédure de choix sont disponibles dans la section 8.4.

8.2.2 Enchaînement des étapes

Une construction de classes traditionnelle comporte deux tâches, la partition de l'ensemble des objets en groupes homogènes (formation) et la production de descriptions (caractérisation). De plus, lorsqu'il s'agit d'une construction de classes selon une mesure de proximité, il est possible de séparer la construction des groupes en deux sous-tâches : calcul de la proximité et formation de classes proprement dit.

Comme nous l'avons vu dans le chapitre 3, dans TROEPS, mais aussi dans n'importe quel environnement faisant coexister plusieurs familles d'individus, la construction sur une famille peut nécessiter des constructions préalables sur d'autres familles. Il existe donc des dépendances entre les processus de construction de classes sur les différentes familles et celles-ci doivent être respectées par l'outil de construction automatique. Ces dépendances se matérialisent sous forme d'attributs relationnels que le concepteur d'une taxonomie a considéré comme pertinents pour le regroupement des objets. Elles constituent une relation binaire entre familles qui n'est pas nécessairement une relation sans circuit car les familles peuvent être co-définies à travers d'attributs relationnels. Cela veut dire, en particulier, qu'il n'est pas possible d'établir un ordre entre les familles pour la construction de classes et que certaines d'entre elles doivent être considérées simultanément. En conséquence, aux deux tâches ci-dessus s'en ajoute une troisième, le découpage de l'ensemble de familles d'objets en sous-ensembles qui sont indissociables pour la construction de classes.

Dans notre approche basée sur la dissimilarité entre objets, chacune des ces tâches est accomplie par une étape spécifique du calcul. L'enchaînement des différentes étapes de la construction de taxonomies, est visualisé sur la figure 8.1. À noter la saisie des paramètres de la construction de la part de l'utilisateur. Elle est effectuée par une interface graphique qui permet de saisir la totalité des paramètres de la construction pour l'ensemble des concepts qu'il est nécessaire de considérer.

Au début, un processus de saisie alterne entre la sélection des points de vue sur lesquels une taxonomie doit être construite et la spécification des paramètres pour la construction. Après la saisie de toutes les informations nécessaires, la construction commence par la détection des CFC dans le graphe des dépendances entre concepts. Chaque CFC est ensuite traitée et cela dans l'ordre topologique : les couples d'objets de la CFC sont tout d'abord comparés pour en extraire une dissimilarité. Ensuite, pour chaque concept de la CFC un regroupement est effectué, suivi d'une création et une caractérisation des classes obtenues. À la fin de chaque construction de taxonomie, celle-ci est intégrée au reste des taxonomies sur le même concept. Après la fin de

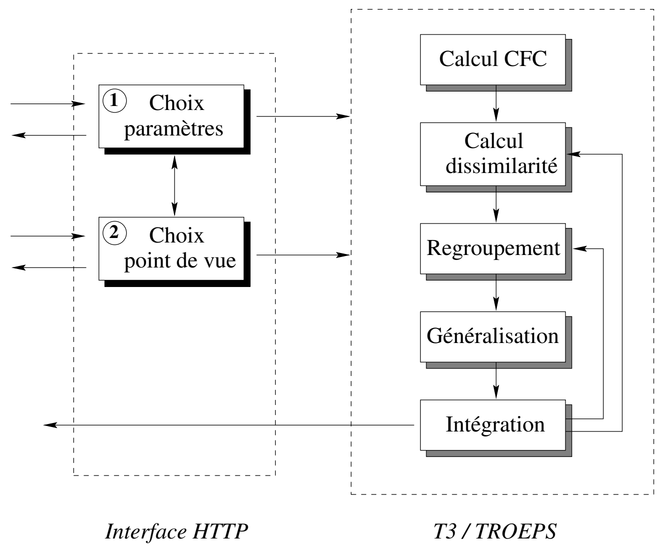


FIG. 8.1: *Vue d'ensemble sur les différentes étapes de la construction de classes en mode multi-concepts dans* TROEPS.

la dernière construction, la main est rendue à l'interface afin d'entamer un processus interactif d'inspection/modification des nouvelles taxonomies en vue de leur validation (ce processus n'est pas dessiné ici). Les opérations sont dessinées en tant que rectangles ombragés. Les ombres noires désignent un processus interactif, alors que les ombres grises indiquent une autonomie pour le processus.

8.2.3 Survol des étapes de la construction

Chacune des étapes successives est présentées dans la suite en précisant les données et le résultat des calculs effectué, ainsi que la nature de ces calculs. L'ordre suivi est celui de l'exécution dans T-TREE.

Calcul des composantes fortement connexes du graphe

À cette étape, l'ensemble des concepts qui sont atteignables depuis le concept de départ, k, est construit. Cet ensemble est divisé en sous-ensembles qui constituent les CFC du graphe des dépendances. Pour cela, l'algorithme présenté dans [Aho *et al.*, 1972] est utilisé. Les données et le résultat du calcul sont présentés dans la table ci-dessous.

Données	**Résultat**
– un point de vue pv sur un concept k, – un ensemble d'attributs $A = \{a_i\}_{i=1,m}$ de k visibles sous pv dont le sous ensemble A^r sont les attributs relationnels, – un point de vue pv_a pour chaque attribut a dans A^r.	– une liste d'ensembles de points de vues sur des concepts différents.

Dans le résultat, chaque ensemble de points de vue correspond à une CFC du graphe $\mathcal{K}$. Les CFC sont présentées dans un ordre qui correspond à un tri topologique sur le DAG extrait de $\mathcal{K}$.

Calcul de proximités

Dans notre approche, le regroupement des objets en groupes de forte similitude se décompose en deux étapes dont la première est l'établissement d'une mesure de ressemblance entre objets. À cette étape du calcul, la collection d'objets O correspondant à une classe c est traduite en une matrice carrée. Les entrées de la matrice sont des objets et le contenu des cases est la valeur de la fonction d^o pour le couple d'objets - indices.

Données	**Résultat**
– une CFC S composée de m concepts k_t, – un point de vue pv_t pour chaque concept k_t, – un ensemble d'attributs a_i, $i = 1, m_t$ pour chaque pv_t – un poids λ_a, pour chaque attribut a	– une matrice de valeurs de dissimilarité D_{dissim}.

Compte-tenu de la complexité de la tâche, de nombreuses d'opérations de nature variée sont employées lors de cette étape. Elle sont toutes consacrées à la comparaison d'un aspect particulier de la description des objets (valeurs simples, relations, collections). La table 8.2.3 rappelle les primitives que nous avons introduites dans les parties antérieures de ce mémoire.

Formation des groupes d'objets

Les groupes d'objets fournis à la fin constituent un ensemble de partitions emboîtées des objets. Ils sont construits à partir de la matrice de proximités à l'aide d'une procédure de regroupement automatique. Il existe une énorme variété de telles procédures (le lecteur intéressé peut consulter [Mirkin, 1997; Gordon, 1999] à ce sujet). Comme nous l'avons souligné dans la partie précédente de ce mémoire, toute méthode de regroupement qui ne se base pas sur des notions géométriques comme le centre de gravité ou l'inertie d'un groupe d'individus (représentés comme un nuage de points), est potentiellement applicable dans notre contexte. Parmi ces méthodes, nous avons choisi pour T-TREE, un ensemble d'algorithmes dits de *classification hiérarchique ascendante* (CAH) (voir section 2.3.4 du chapitre 2) et un algorithme de regroupement

Description	**Nom de primitive**
calcul des dissimilarités sur des objets hors co-définition	DISSIM()
calcul des écarts sur les attributs simples	DISSIM-VAL()
calcul des proximités des couples de collections	DISSIM-MULTI()
calcul d'appariement entre collections qui est optimal, algorithmes 8 et 9 de la section 6.6	CALCUL-APPARIEMENT-OPT()
calcul des proximités des couples de collections	DISSIM-MULTI()
formation des équations pour la dissimilarité des couples d'objets, algorithme 11 de la section 7.6	CALCULER-DISSIM-CFC()
calcul itératif des solutions approchées du système d'équations, algorithme 12 de la section 7.6	RESOLUTION-SE()

TAB. 8.1: *Les principales primitives utilisées durant l'étape de calcul de la dissimilarité.*

par partitionnement successif, les *k-means* (décrit dans la section 2.3.2 du même chapitre).

Données	**Résultat**
– un ensemble d'objets O, – une matrice D_{dissim} de valeurs de dissimilarité entre les objets dans O.	– un ensemble d'ensembles d'objets.

Les ensembles d'objets fournis en résultat correspondent aux extensions des classes découvertes. Ils sont liés par une inclusion ensembliste qui donnera lieu à la spécialisation entre classes.

Les opérations élémentaires pertinentes pour le travail exposé ici sont :

1. la comparaison de valeurs de dissimilarité,
2. le calcul de dissimilarité entre groupes d'objets.

Cette étape peut comporter également des calculs de dissimilarité. Pour certaines méthodes de regroupement, il s'agit de calcul de dissimilarité entre classes, alors que d'autres comparent les objets aux classes, voir les objets à d'autres objets, inexistant dans l'ensemble de départ.

Caractérisation des classes

Les descriptions des classes formées à partir des groupes d'objets sont obtenues en appliquant l'opérateur de généralisation dont les fondements et les principes de calcul ont été présentés au chapitre 4. Rappelons, que celle-ci permet à partir d'un ensemble de valeurs d'attributs, valeurs simples ou objets, de synthétiser une description de type. Le type décrit est le plus spécifique qui inclut les valeurs de départ.

Données	**Résultat**
– un ensemble de groupes d'objets, G.	– une description pour chaque groupe.

Parmi les opérations élémentaires de cette étape citons :

1. synthèse du type le plus spécifique pour chaque attribut simple et monovalué,
2. recherche de la classe la plus spécifique pour chaque attribut relationnel monovalué,
3. aplatissement des collections, extraction des ensembles de leur membres de plus bas niveau, objets ou valeurs simples.

8.3 Architecture de T-TREE

Le système T-TREE est implanté en IlogTalk v.3.3 qui est un dialecte de Lisp muni d'une couche orientée-objet. Il est divisé en plusieurs modules qui sont décrits séparément dans la suite. Le module englobant, appelé d'accueil, assure la communication avec TROEPS et avec l'utilisateur. Un exemple d'utilisation de T-TREE est proposé dans le chapitre suivant.

8.3.1 Module d'accueil

Le module d'accueil assure une structure englobante qui permet de coordonner le travail des autres modules. Il réalise la stratégie de construction conjointe de classes en appelant en séquence ces modules. Lors de ces appels, il redirige les données et le contrôle en provenance et à destination de l'interface utilisateur.

Le module d'accueil est également responsable de l'extensibilité de T-TREE. Ainsi, il offre une interface pour chacun des autres modules imbriqués de sorte que celui-ci puisse être remplacé par un autre module effectuant un traitement différent. Grâce à la modularité, tout nouvel outil de construction de classes peut s'y intégrer facilement.

8.3.2 Module de calcul du DAG des CFC

Pour des raisons d'économie, le graphe $\mathcal{K}$ n'est pas représenté de manière explicite lors du calcul. Il est simulé à l'aide du mécanisme des annotations offert par TROEPS.

Les principales primitives sont :

– l'algorithme de construction du DAG des CFC :

  `(.construire-dag` *`point-de-vue`*`)`

– l'algorithme de recherche en profondeur d'abord dans un graphe :

  `(.recherche-dfs` *`point-de-vue`*`)`

La première construit le DAG entier, la seconde est une recherche en profondeur d'abord en suivant les liens entre concepts (points de vue) établis moyennant les attributs relationnels.

8.3.3 Module de dissimilarité

Le module de dissimilarité englobe les outils fournis pour le calcul d'une dissimilarité entre objets complexes. Il se divise en trois sous-modules qui sont responsables des aspects différents de la notion de dissimilarité. Le sous-module de la normalisation fournit les valeurs maximales de la dissimilarité sur un ensemble d'entités, objets ou valeurs simples. Le sous-module de la dissimilarité est responsable de l'établissement d'une dissimilarité normalisée pour chaque couple d'entités. Finalement, le sous-module de résolution d'équations a pour tâche le calcul de la

nom primitive	**sous-module**	**description**
`(.top-dissimilarity` *`poids* o1 o2`*`)`	proximité simple	dissimilarité entre objets en tant que structures d'attributs et sans co-définition
`(.top-ecart` *`type v1 v2`*`)`	proximité simple	dissimilarité entre valeurs d'attributs, valeurs simples ou objets établie par rapport à un type
`(.top-norme` *`classe attribut`*`)`	normalisation	dissimilarité maximale sur un attribut d'une classe donnée
`(.meilleur-app` *`collection collection #fct-dissim`*`)`	proximité collections	recherche d'appariement optimal entre deux collections
`(.dissim-cfc` *`point-de-vue*`*`)`	proximité co-définition	établissement de la dissimilarité à l'aide d'un système d'équations, composition et résolution du système

dissimilarité sur les objets co-définis. La forte collaboration entre ces trois sous-modules au sein de T-TREE a justifié leur regroupement dans une seule unité de programme.

Les deux premiers modules sont en rapport direct avec le système de gestion de types de TROEPS. Leur constitution est telle qu'elle assure la généricité de la fonction calculée : tous les appels passent par l'interface offerte par les ADTs. De plus, la définition des primitives s'appuie sur la structure de classes du langage hôte qui est définie sur les principales familles d'ADT dans TROEPS. Cela permet en particulier une extension facile vers les nouveaux types de données qu'un concepteur peut facilement introduire dans TROEPS depuis l'extérieur.

Le module peut être utilisé indépendamment des autres modules (regroupement, calcul des CFC) pour des objectifs variés dont la recherche d'objets similaires à un objet particulier dans la base.

Entrées	**Sorties**
– couples d'objets, – ensemble d'attributs, – poids pour chaque attribut.	– une valeur pour la dissimilarité.

À l'heure actuelle, les dissimilarités disponibles sont de la famille de Minkowski. Elles sont implantées sous forme de fonction unique et instanciable dont l'unique paramètre est le degré

`(*tr-minkowski-coef*)`.

Cette valeur permet d'obtenir différentes fonctions concrètes (voir section 2.3) dont deux sont particulièrement utilisée : la *distance Euclidienne* et la *distance de Manhattan*.

Les principales primitives du module sont donnés dans la table 8.3.3 : Le module communique avec TROEPS via son API afin d'extraire les valeurs des divers attributs des objets. Il fait appel aux distances élémentaires entre deux valeurs simples attachées aux types abstraits. De cette manière une intégration du module au sein de TROEPS est réalisée. La paramétrisation de la dissimilarité et l'affichage éventuel de ses valeurs, c'est-à-dire les résultats du travail du module, est à la charge du module d'accueil décrit au début de cette section.

Finalement, mis à part leur utilisation dans le calcul des dissimilarités d'un ensemble d'objets à regrouper, les primitives du module peuvent être utilisées de manière directe. Par exemple, elles sont appelées lors de la comparaison entre un objet particulier et le reste des objets d'une classe. La fonctionnalité correspondante de T-TREE est décrite dans la section 8.5 de ce chapitre.

8.3.4 Module de regroupement automatique

Le module englobe plusieurs sous-modules réalisant des algorithmes de regroupement automatique concrets. Il communique directement avec le module de dissimilarité si l'algorithme choisi par l'utilisateur nécessite des calculs supplémentaires.

Entrées	**Sorties**
– matrice de valeurs de proximité, – ensemble d'objets – ensemble d'attributs (poids)	– un ensemble de groupes d'objets

Actuellement, les algorithmes disponibles ou en préparation sont :

– un algorithme générique de classification ascendante hiérarchique : lien simple, lien complet, [Lance et Williams, 1967a].
– un algorithme de classification descendante : centres mobiles adapté [MacQueen, 1967].

L'algorithme de classification hiérarchique ascendante est implanté sous une forme efficace qui s'inspire des résultats publiés dans l'article de Day et Edelsbrunner [Day et Edelsbrunner, 1984].

Appels des algorithmes :

– appel des algorithmes de *classification automatique ascendante* (liens simple, complet, moyen) :

  `(.clustering-cah` *`méthode classe`*`)`,

– appel de l'algorithme des centres mobiles :

  `(.clustering-k-means` *`k classe`*`)`,

8.3.5 Module de généralisation

Le module inclut les primitives de synthèse de descriptions pour les classes produites par T-TREE. Il est appelé immédiatement après la construction des classes TROEPS qui suit l'étape de regroupement. Il travaille sur des groupes d'objets dont il extrait les valeurs des attributs pour en produire des descriptions de types.

Entrées	**Sorties**
– une classe c – ensemble d'objets O (attachés à la classe) – ensemble d'attributs A (définis dans la classe)	– la description la plus spécifique des objets de O selon les attributs de A

Les principales primitives du module sont :

– la généralisation sur les objets pris comme des structures d'attributs :

(.generaliser-objets *objet* attribut**)

– la généralisation sur un ensemble de valeurs d'attribut :

(.generaliser-attr *valeur* type*)

Lors du calcul des nouvelles descriptions des classes, les primitives du module communiquent intensivement avec le système de gestion des types de TROEPS. Les remarques, concernant l'adaptation de l'architecture interne du module à la structure des types abstraits et l'extensibilité du module vis-à-vis des nouveaux types, faites dans le cas de la dissimilarité s'appliquent également à la généralisation. D'autres primitives de l'API TROEPS sont utilisées lors du parcours des taxonomies de classes disponibles à la recherche des classes les plus spécifiques.

8.3.6 Intégration dans TROEPS

À l'heure actuelle, T-TREE est fortement couplé à TROEPS. Les deux systèmes utilisent leurs API respectives pour effectuer les échanges de données dans les deux sens. Vu de l'extérieur, T-TREE étend l'API de TROEPS par les primitives correspondant aux divers composants du module. De plus, les deux systèmes partagent la même interface graphique : T-TREE est intégré dans l'interface de TROEPS. L'interface est complété afin d'assurer l'accès direct et à distance aux fonctionnalités spécifiques de T-TREE

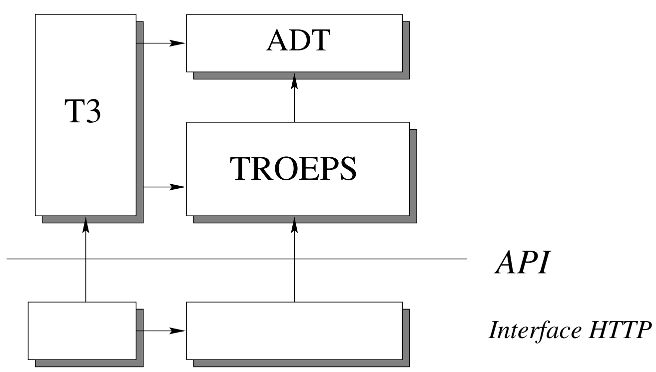


FIG. 8.2: *Les appels de* T-TREE *et de la partie de l'interface graphique qui lui est associée :* T-TREE *utilise l'API de* TROEPS *et celle du système de gestion des types ; l'interface graphique de* T-TREE *utilise les fonctionnalités offertes par l'interface de* TROEPS.

La communication avec T-TREE à travers de son interface graphique est détaillée dans la section suivante.

8.4 Interface graphique

Dans la section 8.2 nous avons présenté une décomposition algorithmique de la construction de taxonomies en étapes de traitement successives permettant d'aboutir, à partir d'un ensemble d'objets à une taxonomies de classes. La traduction de cette décomposition vers une architecture modulaire d'un système de construction automatique de taxonomies a été proposé dans la section suivante, 8.3. Dans cette partie du chapitre, nous adoptons une troisième perspective sur le processus de construction, celle de l'utilisateur.

8.4.1 L'interface HTTP de Troeps

Le système Troeps est muni d'une interface HTTP qui le transforme en serveur de connaissances. Le serveur permet d'accéder aux objets de la base au travers du Web : chaque objet se voit associer une ou plusieurs pages HTML. Ces pages sont générées dynamiquement. L'interface de T-tree fait partie intégrale de l'interface Troeps. Le système peut être appelé à partir de la page d'un concept à condition de se trouver en mode édition.

Afin d'illustrer la vision d'un utilisateur de Troeps sur T-tree, nous décrivons les étapes du calcul qui sont « visibles » pour l'utilisateur au travers de l'interface. La présentation est faite à l'aide de quelques copies d'écran qui retracent le déroulement d'une session avec le serveur Troeps. La base de connaissances Troeps qui est utilisée dans la session est une base dédiée au domaine immobilier. Elle sera décrite avec plus de détails dans le chapitre 9 de ce mémoire.

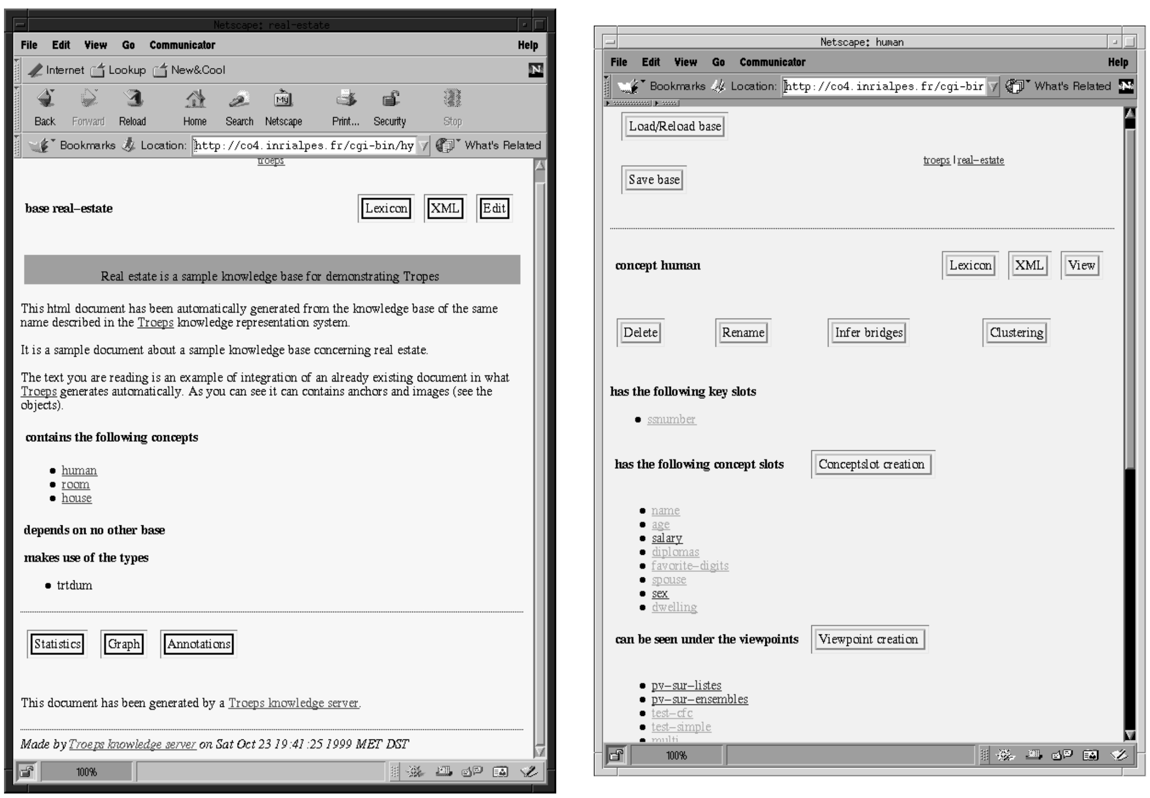


Fig. 8.3: *Pages de visualisation des entités* Troeps *: à gauche, la page d'une base de connaissances est donnée ; à droite, celle d'un concept, humain.*

8.4.2 Les données du processus de construction

Depuis la fenêtre d'accueil du serveur de connaissances il est possible d'accéder à l'ensemble des bases de connaissances servies à un moment donné. Sur la figure 8.3 à gauche, la page de la base `real-estate` est présentée. La base est composée par trois concepts : humain (`human` en *angl.*), appartement (`house` en *angl.*) et pièce (`room` en *angl.*). À droite, la page du concept humain (`human` a été sélectionné depuis la page de la base de connaissances) est montrée. À condition que l'utilisateur possède les droits requis, il peut accéder aux concepts de la base en mode édition. Dès lors, il est possible de faire appel aux services de T-TREE pour analyser l'ensemble des objets instances du concept : l'appel se fait par le bouton « Clustering », dont l'effet est d'activer la partie de l'interface associée à T-TREE. Il donne accès au formulaire de saisie des paramètres pour la construction automatique de taxonomies (*cf.* figure 8.4).

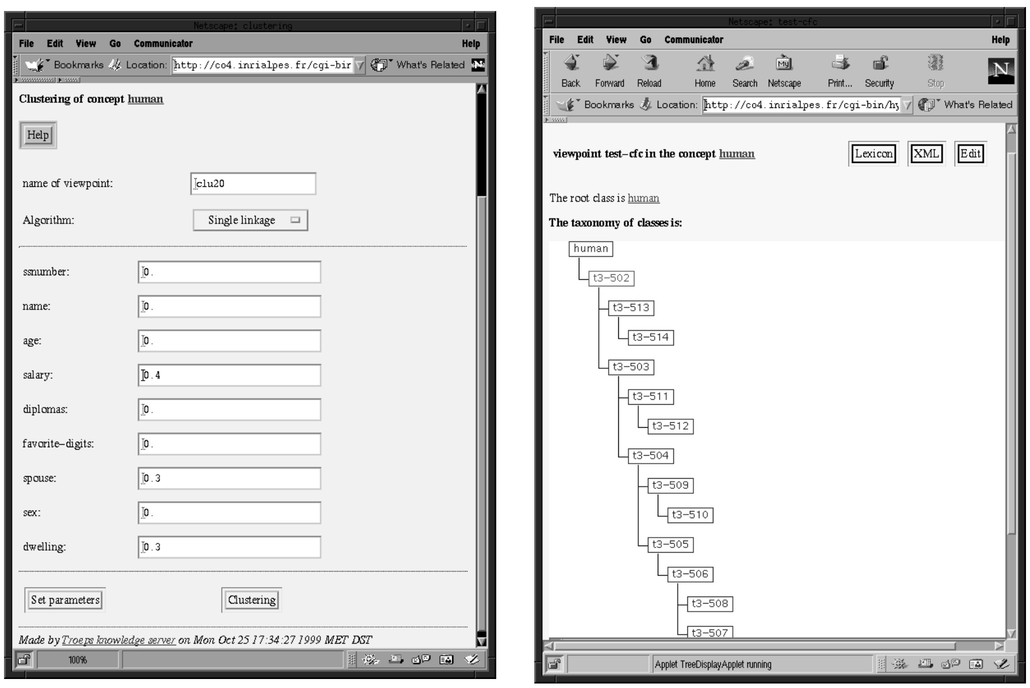


FIG. 8.4: ***Gauche :*** *Écran de saisie des paramètres du regroupement sur un concept : l'utilisateur peut spécifier des poids pour les attributs et choisir un algorithme de regroupement. Les attributs ayant des poids nuls sont éliminés, le reste participe à l'élaboration d'une dissimilarité, puis à la caractérisation des classes. Le choix de la création du point de vue est également offert.* ***Droite :*** *Écran de visualisation d'un point de vue ; cette page sert à visualiser les résultats du travail de* T-TREE.

8.4.3 Les paramètres du processus

L'accès à T-TREE passe par une étape préliminaire de saisie des paramètres pour la construction de classes. À l'heure actuelle, ces paramètres concernent essentiellement l'ensemble des attributs à prendre en compte lors de la construction ainsi que leurs poids respectifs. Ceux-ci sont saisis dans les champs textuels correspondant aux attributs (figure 8.4 à gauche). De plus, un choix est offert entre plusieurs algorithmes de regroupement, tous de type CAH. Finalement, le nom du point de vue sous lequel la nouvelle taxonomie sera hébergée est disponible pour d'éventuelles modifications. Lorsque les paramètres sont correctement saisis, la construction peut démarrer (validation des champs par le bouton « Clustering ») en bas. L'étape de saisie doit être itérée lorsque certains attributs sont de nature relationnelle. Dans ce cas, l'utilisateur peut choisir entre un point de vue déjà existant et la création d'un nouveau point de vue. Dans le premier cas la taxonomie sous le point de vue est déjà disponible et on peut y appuyer une mesure d'écart pour les valeurs de l'attribut sous-jacent. Dans le second cas, la taxonomie à utiliser sur l'attribut sera construite par le processus automatique et hébergée sous le point de vue créé

Après la saisie des paramètres, la construction de taxonomies est lancée. L'interaction avec l'utilisateur reprend à la suite de sa terminaison afin de présenter ses résultats.

8.4.4 Les résultats

Les résultats de la construction sont visualisés à son issue : le serveur envoie la page du point de vue qui contient la nouvelle taxonomie (figure 8.4 à droite). Les classes portent des noms qui sont créés automatiquement. Il revient à l'utilisateur de T-TREE de leur affecter des noms plus intelligibles, en fonction de l'interprétation qu'il peut leur assigner. L'interface assurée par une appliquette JAVA permet l'accès direct aux nouvelles classes (voir figure 8.5 à gauche).

Dans le cas d'une construction conjointe, seule la page du point de vue de départ (celui invoqué en premier) est visualisée.

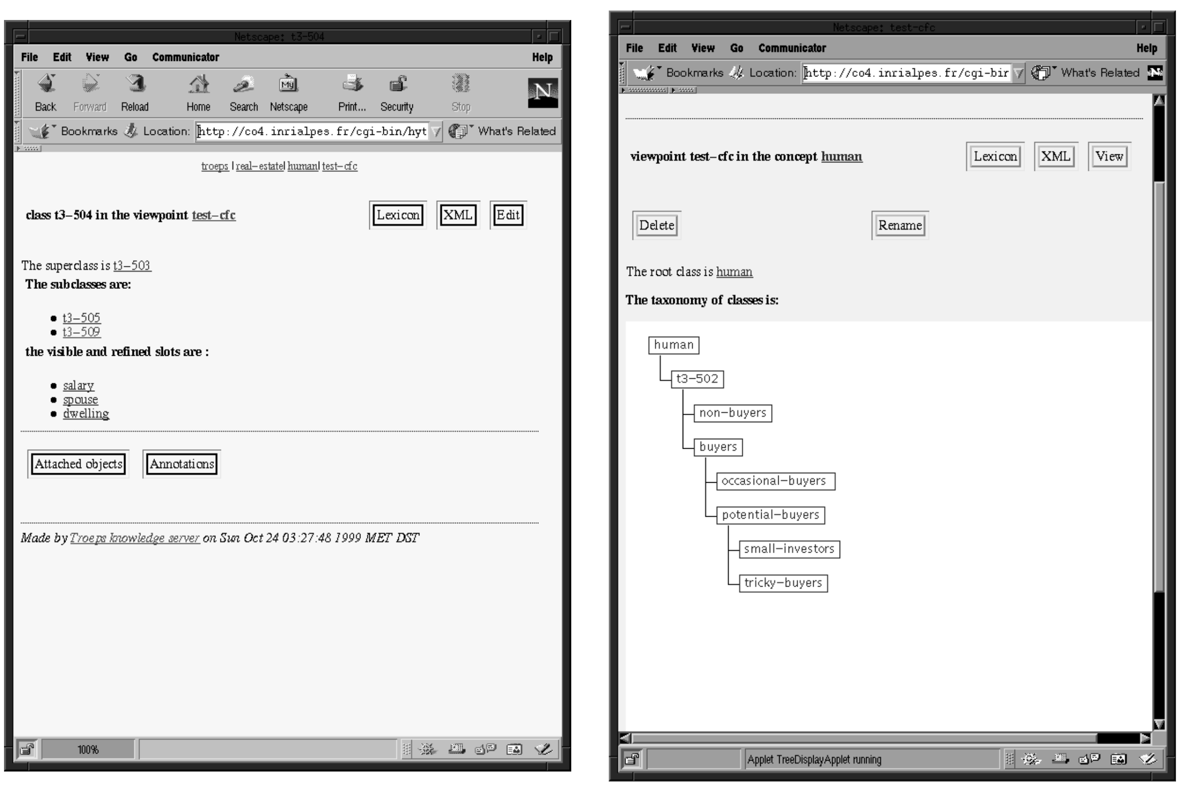


FIG. 8.5: ***Gauche :*** *Écran d'édition d'une classe :* ***Droite :*** *L'état de la taxonomie après une étape d'interprétation qui conduit à supprimer certaines classes et de donner des noms parlants au reste.*

8.4.5 La modification des résultats

Les pages associées aux nouvelles classes sont accessibles afin de faciliter l'interprétation et la validation des résultats de la construction automatique. De plus, il est possible, pour les classes construites par T-TREE, de modifier leur nom, mais aussi le reste de leurs caractéristiques (ensemble des attributs définis, les types et les autres contraintes sur les attributs définies par la classe, etc.). Pour cela, il suffit d'accéder à la page associée à la classe, puis de modifier l'aspect nécessaire en passant par l'interface. Une solution encore plus radicale consiste à supprimer la classe.

Le résultat d'une hypothétique interprétation/modification de la taxonomie sur la droite de la figure 8.4 est donnée sur la figure 8.4, à droite. On observe que seules les classes d'un niveau haut de la taxonomie ont été préservées. Leurs noms indiquent l'interprétation qui leur a été affectée (dans un contexte commercial).

À l'heure actuelle, toutes les modifications sont effectuées de manière manuelle par l'utilisateur. Pour cela des opérations offertes par l'API de TROEPS et accessibles par l'interface sont utilisées. Cependant les opérations disponibles ne couvrent qu'une partie minimale des besoins de transformation sur une taxonomie. Il serait intéressant de définir un ensemble complet de tels opérateurs, à l'instar des opérateurs dans COBWEB (cf. section 2.5.1 du chapitre 2) dont une version étendue est présentée dans [McKusick et Langley, 1991]. Il ne s'agira pas ici, au moins pas dans un plan proche, d'automatiser l'application de tels opérateurs comme cela est réalisé dans le cadre du regroupement probabiliste. Au contraire, nous pensons qu'une étape d'utilisation manuelle est indispensable afin de mieux cerner les besoins de tels opérateurs et leurs éventuels modes d'utilisation. D'ores et déjà, on pourra en noter quelques-uns qui auront incontestablement leur place dans cet étude : un opérateur de «promotion de classe» qui permet d'«élever» une classe dans la taxonomie ; un opérateur d'éclatement d'une classe en plusieurs sous-classes ; un opérateur de fusion d'un ensemble de classes en une nouvelle classe.

8.5 Fonctionnalités annexes

À coté de son rôle principal, l'encadrement du processus d'identification et de définition de classes, T-TREE offre d'autres services qui sont liés à l'aspect exploration des structures de connaissance de la base.

8.5.1 Comparaison d'objets

L'utilisation de la mesure de dissimilarité que nous avons implanté au sein de T-TREE ne se limite pas à l'étape de transformation d'un ensemble d'objets en une matrice de proximités. En effet, la mesure est intégrée au sein de T-TREE sous forme d'outil indépendant : l'accès est fait via l'API. Ainsi, elle peut être appliqué indépendamment des autres outils de construction. Une utilisation possible consiste à comparer deux objets de la base afin d'évaluer leurs ressemblances. La recherche des entités semblables à une entité donnée est une opération fondamentale pour de nombreux mécanismes de manipulation utilisés dans le cadre de l'analyse de données, la recherche d'informations ou le raisonnement par cas. Dans un cadre plus restreint autour de la représentation de connaissances elle peut également épauler une classification (au sens de la représentation de connaissances par objets) comme cela a été suggérée dans [Salotti, 1992] ou encore dans [Bisson, 1995].

La recherche d'objets similaires est une des fonctionnalités de T-TREE, en dehors de la construction de taxonomies. L'utilisateur du système a la possibilité de comparer un objet d'un concept à l'ensemble des autres instances du concept. Pour cela, il a le choix de la fonction de comparaison à utiliser : elle peut être soit une dissimilarité d'attributs, c'est-à-dire de type d^o, soit une dissimilarité de classe multi-point-de-vue, de type δ^{mpv}. Dans les deux cas, l'utilisation de la dissimilarité demande la spécification d'un ensemble de paramètres. Ce sont des poids pour les différents constituants de la dissimilarité : attributs ou points de vue.

La saisie des poids est faite par des formulaires qui ressemblent à ceux utilisés pour la construction de taxonomies. Un tel formulaire est est visualisé à gauche de la figure 8.6. À noter que le même formulaire est utilisé pour la saisie des poids des deux types de fonctions. À droite de la même figure, le résultat de la requête sur les objets similaires à l'objet de prénom Pierre est retracé. Le résultat est présenté à l'utilisateur sous forme d'une liste d'objets triée selon la valeur de la dissimilarité (en ordre décroissant). Les objets les plus similaires se trouvent à la fin de la liste

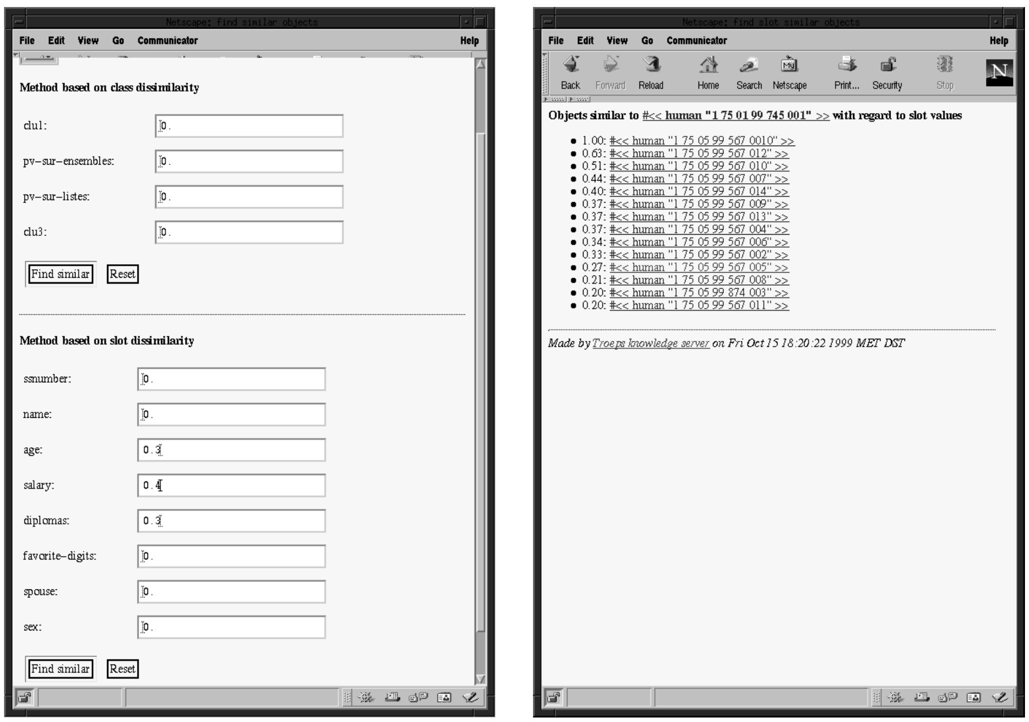


FIG. 8.6: ***Gauche :*** *Écran de saisie pour les paramètres de la dissimilarité de recherche d'objets similaires : le choix est offert entre une dissimilarité de type d^o et de type δ^{mpv}.* ***Droite :****Résultat d'une requête sur les objets similaires.*

La recherche d'objets similaires et la visualisation des valeurs de la dissimilarité a un intérêt même dans le cadre de la construction de taxonomies. D'une part, elle peut permettre de détecter des anomalies dans la description des objets (si la valeur de leurs proximités sont inattendues). Elle peut également aider à peaufiner l'évaluation de l'importance des divers attributs. D'autre part, et cela est propre à l'environnement de TROEPS, elle peut servir dans la vérification de l'adéquation entre les entrées et les sorties d'une procédure de regroupement. Pour cela, on peut comparer les valeurs de la dissimilarité d'objets d^o utilisée par la construction avec celles de la dissimilarité des classes δ^{cl} induite par la taxonomie produite automatiquement.

8.5.2 Inférence de passerelles et comparaison de taxonomies

En TROEPS, plusieurs taxonomies, une par point de vue, existent sur le même ensemble d'objets. Cette multiplicité conduit naturellement à la question des liens possibles entre les classes situées sur différents points de vue. En particulier, il est intéressant de savoir si un même groupe d'objets se trouve à l'intérieur d'une classe (différente de la racine) sous plusieurs perspectives.

Les passerelles offrent un moyen d'exprimer l'existence de ce type de groupements stables entre points de vue (voir section 1.4 du chapitre 1).

Un algorithme d'inférence de passerelles est implanté dans T-TREE [Euzenat, 1993a]. Son effet est la détection automatique de toutes les passerelles qui existent au sein d'un ensemble de classes. Lors de la construction automatique de taxonomies à l'aide de T-TREE, l'inférence de passerelles retrouve une utilisation naturelle : elle est le moyen adéquat de comparaison entre les nouvelles taxonomies et celles déjà disponibles. L'interprétation d'un ensemble de passerelles entre deux taxonomies est intuitive : le plus grand nombre de passerelles à un niveau bas des deux taxonomies indique une plus forte «similitude» entre elles. Cette comparaison «structurelle» peut venir en complément d'une comparaison plus orientée vers l'établissement d'une valeur numérique comme cela est d'usage en analyse de données.

En effet, les problèmes liés à la comparaison de différentes structures de classes sur un ensemble d'individus donné sont étudiés sous l'appellation du *consensus de hiérarchies* (voir par exemple [Leclerc, 1998] sur ce sujet). Le consensus a pour but la construction d'une taxonomie qui «résume» l'ensemble des taxonomies de départ et cela dépasse nos propres ambitions. Cependant, une des multiples manières pour aborder le problème, la *méthode de la médiane* [Margush et McMorris, 1981] passe par la comparaison explicite des taxonomies à l'aide d'une mesure de distance. La question de la comparaison des taxonomies basée sur l'ensemble des passerelles qu'elles admettent mériterait une étude plus approfondie. Il est intéressant, en particulier, d'établir le lien entre cet ensemble et les mesures habituelles de consensus d'arbres.

8.6 Conclusion

Le cycle de construction de taxonomies que nous avons présenté au début de ce chapitre est la réalisation de notre stratégie décrite au chapitre 3. Il met en œuvre les outils proposés dans les chapitres de la partie II du mémoire : les mesures de dissimilarité et les opérateurs de généralisation. Le processus global de construction hérite donc de leurs qualités de complétude vis-à-vis des descriptions des objets et de leur composabilité face à une forte structuration dans ces descriptions.

L'implantation au sein de T-TREE qui est parfaitement intégré avec TROEPS permet de construire les taxonomies directement dans le contexte de la base de connaissances. Ainsi, l'ensemble des connaissances disponibles peuvent être mises à contribution lors de cette construction. Cependant, la méthode effective de construction offre une liberté considérable quant à la forme de cette contribution et son importance relative. Les choix de l'utilisateur sur ce point sont introduits dans le processus de construction par l'interface du système qui assure également une visualisation des résultats sous une forme qui facilite leur interprétation et leur modification.

En somme, le système T-TREE répond aux besoins soulevés dans la partie introductive du mémoire, à savoir la complétude des outils, la possibilité d'adapter la construction aux objectifs de l'utilisateur, l'intégration au sein de la base de connaissances à objets et la flexibilité par rapport à son degré de structuration.

Cependant, de nouvelles questions se posent concernant le processus de construction tel que nous l'avons dessiné ici. La plus immédiate de ces questions concerne la comparaison entre les taxonomies produites par différentes méthodes, ainsi qu'entre les nouvelles taxonomies et celles déjà dans la base. Parmi les autres sujets qui semblent intéressants, citons l'adaptation des outils à la comparaison de classes. En effet, dans beaucoup de cas de conception d'une taxonomie, des classes terminales sont déjà disponibles et le but devient de les organiser en des classes plus abstraites. L'extension nécessaire de nos outils, est loin d'être acquise, surtout dans le cas de classes co-définies et mériterait d'être examinée en profondeur.

Aprés avor décrit l'architecture de système T-TREE, nous présentons, au chapitre suivant, quelques exemples de ce que pourra être son utilisation au sein d'une base de connaissances.

9

Exemple d'utilisation de T-TREE

Dans ce chapitre nous illustrons le fonctionnement de T-TREE et par là l'approche proposée pour la construction de taxonomies au sein d'une base de connaissances à objets. Pour cela, nous utilisons une base de connaissances qui modélise le domaine immobilier.

Le chapitre est organisé comme suit. Au début, section 9.1, le modèle du domaine immobilier réalisé par la base est présenté : les diverses familles d'individus, les caractéristiques des membres de chaque famille et les relations entre familles sont décrites. La section 2.1.3 a pour objectif de montrer l'aspect flexible de la stratégie de construction dans T-TREE : elle offre des exemples de construction de taxonomies au sein de la base selon plusieurs scénarios différents. Les exemples de la section 9.4 illustrent quant à eux les variations dans les résultats de la construction en fonctions des paramètres fournis en entrée. Le chapitre se termine par une section dressant la liste des diverses expérimentations que notre système admet.

9.1 Modélisation

La base porte sur le domaine immobilier, elle a été en partie créée à partir des petites annonces parues dans la presse locale. Dans sa définition, nous nous sommes inspirés des deux exemples donnés dans le chapitre 7. La base demeure néanmoins une base de démonstration.

Les entités représentées sont des personnes, des habitations et des pièces des habitations. Ces trois sortes d'entités entretiennent des relations de nature variée. Tout d'abord, les personnes sont liés aux habitations en tant que propriétaires. Chaque habitation appartient donc à au moins une (ici exactement une) personne. Plusieurs personnes peuvent habiter dans un appartement, la relation sous-jacente est donc multivaluée. De plus, les personnes peuvent être liées par une relation conjugale. Finalement, les habitations sont considérées avec leurs pièces qui constituent leurs parties.

9.1.1 Le schéma de la base

La base TROEPS contient trois concepts : `humain` dont les instances sont des êtres humains, `appartement` qui regroupe les objets représentant des habitations et `pièce`, le concept modélisant les pièces des habitations.

Les trois concepts sont liés par les attributs `propriétaire`, `habitants`, `habitation`, `conjoint` et `pièces`. Les attributs `habitant` et `habitation` expriment les deux aspect de la même relation qui lie un appartement aux personnes qui l'occupent. La relation est donc de type *1 :N*. L'attribut `propriétaire` indique le possesseur d'un appartement qui n'est pas nécessairement parmi ses habitants. La relation inverse qui n'est pas explicitement représentée ici est aussi de type *1 :N*. Les liens conjugaux sont modélisés par l'attribut `conjoint`. Nous avons délibérément restreint la relation conjugale à un type bijectif, c'est-à-dire *1 :1* : une personne ne peut avoir qu'au plus un conjoint[33]. Finalement, la relation entre un appartement et ses pièces est une relation de composition classique (c'est-à-dire antiréflexive et antisymétrique). Cependant, et contrairement à beaucoup d'autres relations de composition, ici l'objet composite, l'appartement, ne se réduit pas à la simple somme de ses composants.

Le schéma Entité-Association de la base immobilière est présentée sur la figure 9.1.

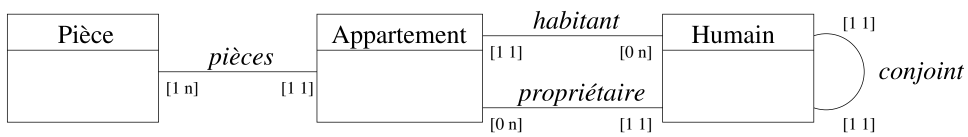


FIG. 9.1: *Schéma Entité-Association de la base immobilière.*

Plusieurs aspects du modèle ci-dessus mériteraient que l'on s'y arrête. Les aspects qui nous intéressent particulièrement sont :

– les relations multivaluées : `habitants` et `pièces`,
– les relations de co-définition entre concepts : `habitants` et `habitation`, `propriétaire` et `habitation`,
– les relations de co-définition entre objets : `conjoint`, `habitants` et `habitation`.

Du point de vue de la construction automatique de taxonomies, la base représente un graphe à deux composantes fortement connexes (CFC).

Dans la suite, nous prenons l'hypothèse que la base de connaissances est construite dans le but principal d'améliorer la satisfaction des personnes cherchant à acheter/louer des habitations mais aussi celle des propriétaires de ces habitations (surtout en cas de location). Nous considérons également que la base a pour autre objectif de servir de support lors de la commercialisation des nouveaux produits destinés aux locataires et/ou propriétaires (par exemple accès à des services de recherche, accès aux offres disponibles, assurances mobilières/immobilières, etc.).

Pour atteindre ces objectifs, il semble naturel de chercher à définir des groupes de personnes, propriétaires ou locataires/acheteurs, ayant un profil similaire compte tenu des objectifs de l'opération planifiée (offre d'une nouvelle assurance, par exemple). Ces groupes se constituent par rapport aux caractéristiques pertinentes des individus ou des objets physiques concernés.

Dans la suite, nous montrons, par le biais de quelques exemples, que l'approche proposée pour la construction automatique de classes peut être appliquée dans le contexte de la conception de la base de connaissances. Avant de procéder à la description des quelques situations où la tâche de la construction de taxonomies se dessine, nous présentons pour chaque famille d'objets ses caractéristiques et l'ensemble de ses membres.

9.1.2 Le concept appartement

Le concept `appartement` est intéressant pour notre problématique pour plusieurs raisons. D'une part, il combine des caractéristiques des trois natures distinguées dans TROEPS (mais aussi dans d'autres modèles à objets) : des propriétés et des relations qui se divisent encore en liens et compositions. D'autre part, deux des liens sont multivalués et l'un d'eux participe à un

33. Sans restriction sur les sexes des deux membres du couple.

circuit de co-définition. Le paragraphe suivant offre une description tabulaire des attributs du concept `appartement`. Il est suivi de la présentation des instances du concept faisant partie du jeu de données pour nos expérimentations.

Les attributs

Dans la base de connaissances TROEPS, le concept `appartement` possède un bon nombre de caractéristiques (12) dont nous n'avons retenu que les plus pertinentes pour les regroupements à effectuer. Ils sont énumérés dans la table ci-dessous. À noter l'exclusion de l'attribut `habitants` qui n'apporte pas d'informations importantes pour les objectifs des constructions de classes discutés dans la section suivante.

attribut	type	constructeur	nature
`adresse`	`chaîne`	`un`	`propriété`
`numéro`	`entier`	`un`	`propriété`
`surface`	`entier`	`un`	`propriété`
`prix`	`réel`	`un`	`propriété`
`nb-pièces`	`entier`	`un`	`propriété`
`loyer`	`entier`	`un`	`propriété`
`propriétaire`	`humain`	`un`	`lien`
`pièces`	`pièce`	`ensemble`	`composant`

Les instances

Le jeu de données est composé de dix instances identifiées dans TROEPS par leur adresse composée d'une rue/ville, un numéro et un étage. Dans le texte, afin de faciliter leur identification, nous les avons numérotées (de 1 à 10).

TAB. 9.1: *L'ensemble des instances du concept* `appartement`

id	adresse	nr	étg	surf.	nb-pcs	prix	proprio	pièces
1	r. de l'Usine	17	3	35	4	6000.	10	{1-1, 1-2, 1-3, 1-4}
2	meylan	176	3	35	3	8000.	9	{2-1, 2-2, 2-3}
3	st martin	95	3	20	1	3000.	8	{3-1}
4	grenoble	113	0	60	2	4000.	7	{4-1, 4-2}
5	meylan	60	6	74	4	11000.	6	{5-1, 5-2, 5-3, 5-4}
6	eybens	47	0	80	2	7000.	5	{6-1, 6-2}
7	eybens	13	4	65	8	11000.	4	{7-1, 7-2, 7-3, 7-4, 7-5, 7-6}
8	grenoble	21	2	70	3	8000.	3	{8-1, 8-2, 8-3}
9	grenoble	32	3	90	5	14000.	2	{9-1, 9-2, 9-3, 9-4, 9-5}
10	rue Monge	3	2	45	2	15000.	1	{10-1, 10-2}

9.1.3 Humain

Les humains sont décrits dans la base par des attributs reflétant leur profil de clients immobiliers : les revenus, l'âge et l'habitation. D'autres attributs modélisent des aspects qui ne sont pas en rapport direct, mais peuvent néanmoins influer sur le comportement des personnes en tant que clients : le niveau éducatif et les éventuels liens conjugaux.

Les attributs

Les attributs du concept sont décrits par la table ci-dessous. Deux des attributs de la table méritent plus d'attention. D'une part, l'attribut `diplômes` représente un ensemble de diplômes qui sont comparés en tant que symboles, c'est-à-dire comme des valeurs d'un type énuméré. D'autre part, l'attribut `conjoint`, dont il a été beaucoup question au chapitre 7, introduit une co-définition entre les couples de conjoints dans la base. Chacun de ces attributs pose des difficultés particulières lors de la construction automatique de classes. Nous reviendrons dans les sections 9.3 et 9.4 sur ces difficultés.

attribut	type	constructeur	nature
`nom`	`chaîne`	`un`	`propriété`
`sexe`	`booléen`	`un`	`propriété`
`âge`	`entier`	`un`	`propriété`
`salaire`	`réel`	`un`	`propriété`
`diplômes`	`chaîne`	`ensemble`	`propriété`
`conjoint`	`humain`	`un`	`lien`
`habitation`	`appartement`	`un`	`lien`

Les instances

Quinze instances du concept font partie du jeu de données. Elles sont identifiables par un numéro unique (le numéro d'INSEE) dans la base TROEPS. Pour les mêmes raisons que dans le cas des appartements, nous avons préféré une identification par des nombres allant de 1 à 15.

TAB. 9.2: *L'ensemble des instances du concept* `humain`

id	nom	âge	sexe	salaire	diplômes	conjoint	habitation
1.	Pierre	26	m	3.	{ "BS" "MS" "PhD" }	Anne	10
2.	Anne	23	f	2.3	{ "BS" "MS" }	Pierre	10
3.	Yves	27	m	2.2	{ "BS" "MS" "PhD" }	Léa	8
4.	Léa	30	f	2.7	{ "BS" }	Yves	8
5.	Claude	32	m	2.9	{ "BS" "MBA" "PhD" }	Sophie	6
6.	Sophie	35	f	3.2	{ "BS" "MS" "MBA" "PhD" }	Claude	6
7.	Olaf	29	m	3.3	{ "MBA" }	Karinne	3
8.	Karinne	25	f	2.6	{ "BS" "MS" }	Olaf	3
9.	Manuel	31	m	2.8	{ "BS" "MBA" }	Sonia	1
10.	Sonia	28	f	2.3	{ "Litt D" }	Manuel	1
11.	Thomas	34	m	3.	{ "BS" "MS" "PhD" }	Julie	2
12.	Julie	32	f	3.8	{ "MD" }	Thomas	2
13.	Marco	34	m	2.7	{ "BS" "MS" }	Carla	4
14.	Carla	29	f	4.	{ "BS" "PhD" }	Marco	4
15.	*undef*	*undef*	*undef*	*undef*	*undef*	*undef*	*undef*

Il est à noter la présence d'une instance, la quinzième dont aucun des attributs n'est renseigné (valeur *undef*). Son rôle ici est de témoigner de la correction de nos calculs : une procédure de calcul de la dissimilarité doit toujours aboutir à une valeur maximale lors qu'elle compare une autre instance dont les attributs sont tous renseignés (le reste des instances ci-dessous) à cette instance.

9.1.4 Pièce

Le concept `pièce` ne possède que deux attributs pertinents : la surface de la pièce et son orientation vis à vis des quatre points cardinaux.

Les attributs

À noter la représentation de l'orientation de la pièce par un attribut multivalué dont le type de base inclut les quatre points principaux. Cette représentation permet une comparaison entre orientations plus flexible que si celles-ci étaient représentées par des symboles.

attribut	type	constructeur	nature
`orientation`	`chaîne`	`ensemble`	`propriété`
`surface`	`entier`	`un`	`propriété`

Les instances

Les pièces sont identifiées à l'aide d'un numéro composé dont la première partie indique l'indice de l'habitation englobant, alors que la seconde est propre à l'objet pièce.

TAB. 9.3: *L'ensemble des instances du concept* `pièce`

id	surface	orientation	id	surface	orientation
1-1.	12.	("S")	1-2.	11.	("S" "E")
1-3.	16.	("N" "E")	living-room.	18.	("S" "E")
2-1.	13.	("S" "O")	2-2.	10.	("N" "E")
2-3.	19.	("E")	3-1.	22.	("S")
4-1.	12.	("N" "E" "S")	4-2.	15.	("N" "O")
5-1.	14.	("N" "E")	5-2.	16.	("S" "O")
5-3.	9.	("O")	5-4.	22.	("S")
6-1.	18.	("N")	6-2.	12.	("S" "E")
7-1.	10.	("N" "E")	7-2.	18.	("N" "O")
7-3.	13.	("N")	7-4.	11.	("O" "S" "E")
7-5.	16.	("N" "O")	7-6.	12.	("S")
8-1.	17.	("O" "N" "E")	8-2.	10.	("S" "E" "N")
8-3.	22.	("S" "O")	9-1.	18.	("O")
9-2.	15.	("S" "O")	9-3.	9.	("N")
9-4.	12.	("E")	9-5.	21.	("E")
10-1.	19.	("S" "O")	10-2.	20.	("N" "E")

9.2 Trois scénarios pour la construction de taxonomies

Nous présentons trois scénarios qui illustrent des situation différentes qui peuvent se présenter lors de la construction incrémentale de la base de connaissances. Le premier scénario qui est de loin le plus complexe consiste en une construction conjointe de taxonomies sur les trois concepts de la base. Il est examiné en détail avec la succession des diverses étapes de la construction. Les deux autres scénarios représentent la construction d'une taxonomie sur un concept auto-référant (c'est-à-dire en co-définition avec lui-même) et la construction d'une taxonomie par référence à une taxonomie existante respectivement.

Il est à noter que sur les copies d'écrans toutes les entités de la base portent des noms en anglais. Afin de faciliter la lecture du document, nous les citons dans le texte avec leur traduction française.

9.2.1 Construction multiple de taxonomies

Pour cet exemple nous avons imaginé une situation où les personnes, clients effectifs ou potentiels, doivent être répartis en fonction de leur éventuel intérêt pour une nouvelle formule d'assurance pour l'habitation. Les facteurs dont cet intérêt peut dépendre se rapportent soit directement à la personne soit à son habitation. Le seul facteur du premier type est le revenu de la personne. Quant aux facteurs relatifs à l'habitation, ils concernent sa taille, sa valeur (le prix), ainsi que son propriétaire et surtout ses revenus (la charge de l'assurance étant partagée entre propriétaire et locataire, s'il existe). La taille des appartements n'est pas prise dans sa globalité, mais répartie entre les différentes pièces ce qui permet un meilleur compromis entre le type de l'appartement et sa surface totale. Pour cela, les appartements sont considérés avec l'ensemble de leurs pièces, chacune caractérisée par sa surface.

Pour résumer, les groupes homogènes de personnes (dont on peut s'attendre à ce qu'ils aient une attitude similaire vis à vis, de la nouvelle assurance) se caractérisent par des revenus proches et des habitations de caractéristiques similaires. Une première tâche de construction est donc le regroupement des personnes en classes. Bien que cela ne soit pas explicitement exigé par le problème, un regroupement des habitations en classes s'impose. Mis à part l'intérêt que les classes ainsi construites peuvent représenter, elles ont aussi une utilité en tant que descriptions synthétiques pour leurs instances, descriptions qui seront utilisées pour caractériser les classes sur les personnes. Pour les mêmes raisons, il convient de construire des classes pertinentes sur les pièces aussi.

Il y donc trois taxonomies à construire dont une sur un concept d'objets simples. Les deux autres sont en revanche liés par une co-définition et doivent être traités simultanément. La situation est représentée de manière visuelle sur la figure 9.2.

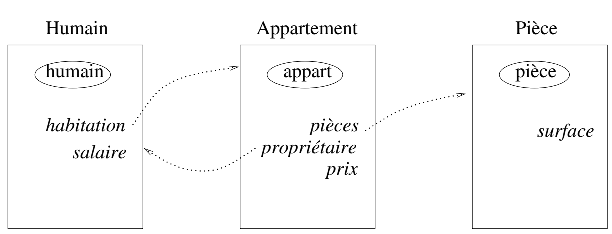


FIG. 9.2: *Les dépendances entre les tâches de construction de taxonomies entre les trois concepts : les relations entre concepts se traduisent par des relations entre les classes recherchées. Un nouveau point de vue est créé pour chaque concept, il est destiné à accueillir la nouvelle taxonomie.*

9.2.2 Construction d'une taxonomie auto-référante

Dans la section précédente nous avons cité un exemple de construction de classes où plusieurs concepts ont été traités simultanément (`humain` et `appartement`). La raison de ce traitement était l'existence de liens de co-définition entre les deux concepts. En toute généralité, la présence de plusieurs concepts n'est pas nécessaire pour qu'il y ait une co-définition. Nous l'avons vu au chapitre 7, l'attribut `conjoint` introduit, par la nature symétrique de la relation conjugale, des co-définitions qu'il est difficile d'éviter.

Le but de la construction de classes ici est de retrouver des groupes de personnes qui sont homogènes, en particulier par rapport à leurs relations. Un besoin de regroupement de ce type peut se présenter lorsque, par exemple, l'intérêt potentiel des personnes pour des éventuels prêts immobiliers est à évaluer. De nouveau, il faut prendre en compte les revenus d'une personne, mais aussi l'évaluer par rapport à ceux de son conjoint, s'il en existe un. En revanche, les caractéristiques de leurs habitations actuelles ne semblent pas nécessairement très pertinentes pour l'objectif. La situation est visualisée sur la figure 9.3.

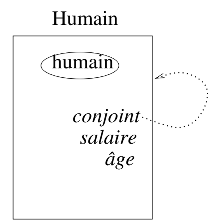


FIG. 9.3: *Un seul point de vue est défini et une seule taxonomie est à construire. Cependant, le lien circulaire entre les objets du même concept ne permet pas une application directe des algorithmes de regroupement standard.*

9.2.3 Utilisation d'une taxonomie existante

Les scénarios un et deux ont illustré des situations où des taxonomies devaient être construites sur l'ensemble de concepts liés. Cette manière de procéder, qui n'est pas sans rappeler l'exploration exhaustive des liens entre objets dans KBG, n'est pas toujours la plus adaptée : la construction de nouvelles taxonomies alors que de telles taxonomies existent déjà n'est pas forcement utile, ni très efficace. Il faut donc prévoir une intégration de ces taxonomies existantes dans le processus. Avec ce but en tête, nous suggérons un troisième scénario qui met en jeu de nouveau

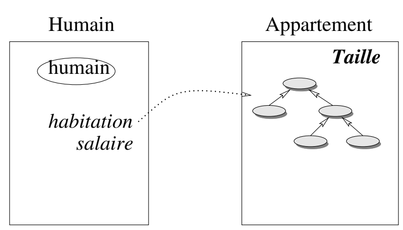


FIG. 9.4: *Une seule taxonomie est à construire. Elle utilise une autre taxonomie qui existe déjà.*

plusieurs concepts. Ce scénario pourra se présenter dans un cas où les personnes de la base doivent être regroupées selon leurs obligations fiscales (taxe d'habitation). Sous cette perspective, pour le concept `humain`, les attributs à prendre en compte sont `salaire` et `habitation`. Quant au concept `appartement`, le type d'habitation qu'il représente (ici le nombre des pièces, pour simplifier) et la `surface` sont les attributs pertinents. Il se trouve, que dans la base, une taxonomie qui rend compte de l'aspect taille des appartements existe déjà. Elle appartient au point de vue `taille` du concept. Cette taxonomie peut donc être utilisée dans la comparaison des objets du concept `appartement` (à l'aide d'une mesure de type δ^{cl}) plutôt que de construire une nouvelle taxonomie. Il reste donc à construire la taxonomie sur `humain` dont la structure va nécessairement refléter celle de la taxonomie sous `taille`. La situation est visualisée sur la figure 9.4.

9.3 Calcul des résultats

Dans cette section, les résultat de la construction de taxonomies pour chacun des scénarios sont présentés. De plus, la procédure de calcul dans le premier scénario est détaillée : les étapes intermédiaires sont retracées.

9.3.1 Construction multiple de taxonomies

Comme nous l'avons vu au chapitre 7, la co-définition nécessite un calcul de la dissimilarité par le point fixe d'une fonction. Cependant, dans T-TREE, cela reste transparent pour l'utilisateur, qui voit toujours la même interface, qu'il y ait ou pas de résolution de système d'équations.

Choix des paramètres

Les attributs choisis pour inclure dans le processus de construction ainsi que leurs poids sont donnés par la table ci-dessous :

humain		appartement		pièce	
salaire	0,5	pièces	0,2	surface	1.0
habitation	0,5	propriétaire	0,3		
		prix	0,5		

Pour l'algorithme de regroupement, une procédure de type *lien complet* a été retenue pour les trois concepts[34]. Les différentes étapes de la saisie de paramètres pour une construction multiple de taxonomies sont visualisées sur la figure 9.5. Le processus global commence par une saisie des poids des attributs : d'abord dans la fenêtre du regroupement sur `appartement` puis dans les fenêtres associées aux deux autres concepts, car de nouvelles taxonomies sont à créer sur ces concepts également. Sur la partie inférieure de la figure 9.5, les deux écrans correspondant aux concepts `pièce` et `humain` sont visualisés.

Recherche des CFC du graphe des dépendances

Tel qu'il a été défini par la sélection d'attributs ci-dessus, le graphe $\mathcal{K}$ des dépendances inter-concepts se décompose en deux CFC :

- $\mathcal{S}_1$ = `{pièce}` et
- $\mathcal{S}_2$ = `{appartement,humain}`.

La première CFC est triviale, la construction des classes peut être effectuée de manière traditionnelle. En revanche, sur $\mathcal{S}_2$ une application de la construction conjointe est nécessaire.

34. À noter que l'interface permet de choisir l'algorithme pour chaque construction de taxonomie, il n'est donc pas nécessaire d'utiliser le même pour les trois tâches ici.

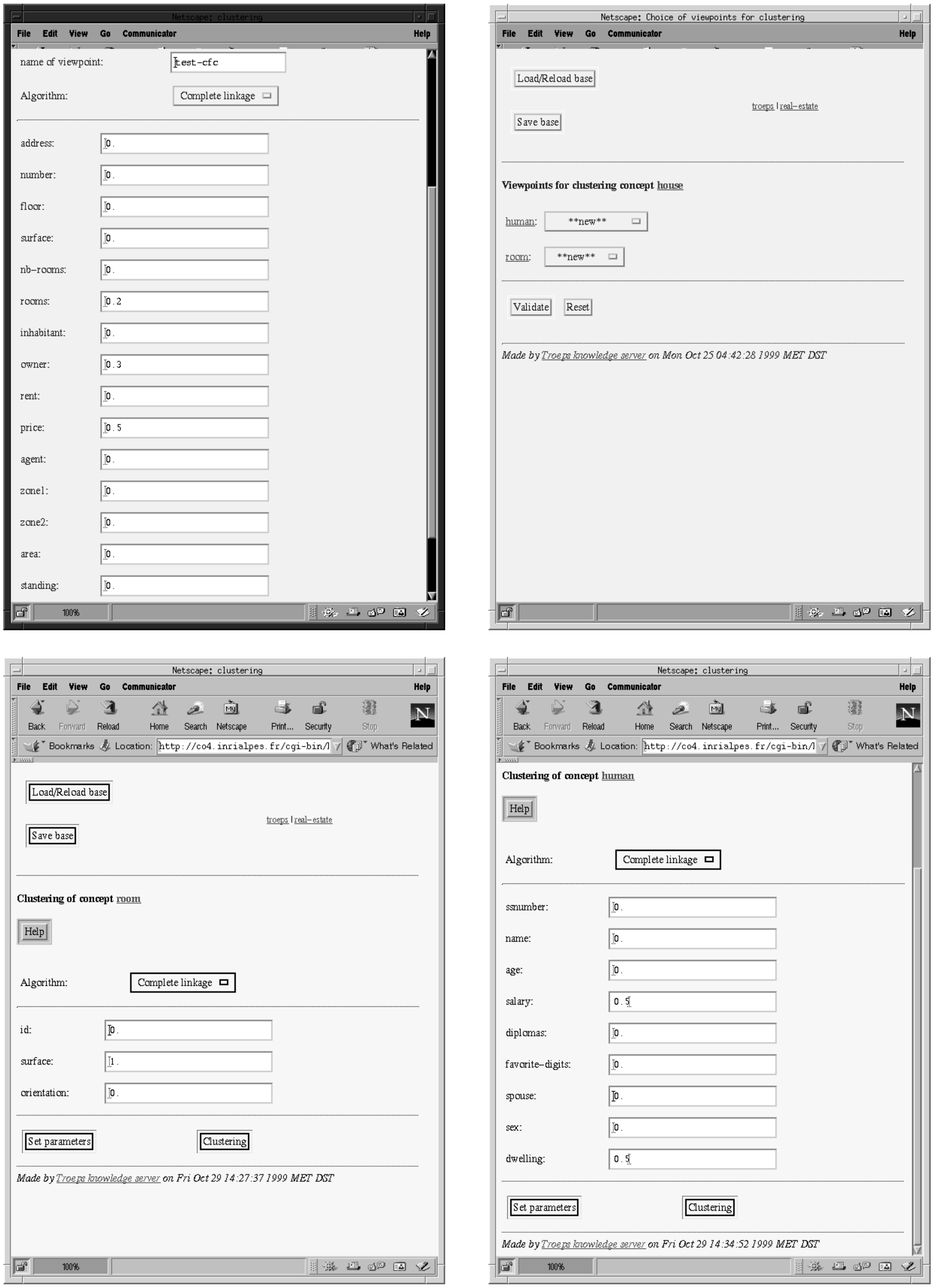


FIG. 9.5: *Les écrans de saisie des paramètres pour regroupement conjoint sur les trois concepts. Le concept de départ est* `humain` *(l'écran en haut à gauche). La sélection des attributs relations pour le regroupement conduit au choix de points de vue sur les concepts sous-jacents (l'écran en haut à droite). Si les points de vue sont nouveaux, des saisies de paramètres pour le regroupement va avoir lieu (les deux écrans en bas).*

Exemple de mesure de dissimilarité

Lors de l'étape préliminaire de la construction, les dissimilarités entre objets sont calculées. Ici les dissimilarités sur les couples d'appartements et sur les couples d'humains sont calculées de manière simultanée. À noter que la co-définition entre les deux concepts considérés n'entraîne pas nécessairement co-définition entre objets. Par exemple, la personne de nommée Pierre qui possède l'appartement numéro dix (3, rue Monge) est également un des habitants de celui-ci. En conséquence les deux objets se retrouvent en co-définition et lorsque ils sont comparés à un autre couple co-défini, par exemple, à Karinne et l'appartement trois (95, St Martin), les deux dissimilarités, celle des humains et celle des habitations, doivent être calculées à l'aide des équations. En revanche, une partie des personnes ne sont pas des propriétaires, par exemple Marco. Elles ne peuvent donc pas se retrouver impliqués dans une co-définition avec un appartement. En dépit de ces différences, les objets d'un concept qui possède des liens de co-définition sont systématiquement comparés moyennant une dissimilarité récursive.

Les valeurs de la dissimilarité sur le concept `humain` sont présentées dans la table suivante. Les valeurs minimales (inférieures à 20), celles qui sont susceptibles de conduire au regroupement des objets sous-jacents en des classes spécifiques, sont notées en gras.

TAB. 9.4: *Les valeurs de la dissimilarité sur le concept `humain`, calculée sur les attributs `salaire`, `conjoint` et `habitation`.*

	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.
1.	0.00	0.36	0.47	0.32	0.30	0.30	0.35	0.42	0.30	0.46	0.33	0.47	0.41	0.49	1.00
2.		0.00	0.29	0.40	0.38	0.44	0.52	0.35	0.42	0.27	0.46	0.60	0.39	0.65	1.00
3.			0.00	0.26	0.28	0.33	0.50	0.36	0.33	**0.19**	0.42	0.56	0.39	0.62	1.00
4.				0.00	**0.19**	0.25	0.40	0.32	**0.19**	0.30	0.33	0.47	0.30	0.53	1.00
5.					0.00	**0.15**	0.34	0.28	0.25	0.33	0.21	0.38	0.22	0.42	1.00
6.						0.00	0.25	0.37	0.30	0.39	0.25	0.30	0.31	0.33	1.00
7.							0.00	0.36	0.24	0.36	0.25	0.24	0.36	0.30	1.00
8.								0.00	0.22	0.21	0.23	0.41	**0.19**	0.50	1.00
9.									0.00	0.26	**0.19**	0.33	0.21	0.39	1.00
10.										0.00	0.28	0.42	0.25	0.49	1.00
11.											0.00	0.41	**0.14**	0.37	1.00
12.												0.00	0.39	**0.12**	1.00
13.													0.00	0.67	1.00
14.														0.00	1.00
15.															0.00

Il est à noter que la plus petite valeur, 0.12, est mesurée sur le couple `(Julie, Carla)` qui possèdent, en effet, des caractéristiques proches (si l'on se examine les entrées correspondantes de la table 9.2). Inversement, la plus forte dissimilarité est prélevée sur le couple de conjoints (Marco, Carla). Cette valeur est en grande partie due à l'écart sur les salaires entre les deux conjoints.

Taxonomies de classes

La séparation entre les étapes de regroupement et de caractérisation reste transparente pour l'utilisateur de T-TREE. À l'heure actuelle, il ne voit que le produit final du processus de construction : un ensemble de taxonomies dont les classes ont toutes une description en termes d'attributs. Les résultats de la construction conjointe sont donnés par la figure 9.6. À droite, une séparation au premier niveau entre les classes `t3-503` et `t3-513` est visible. La classe contient les trois personnes `(Julie, Carla, Olaf)` dont les représentations se détachent du reste des instances du concept `humain` par leurs salaires élevés.

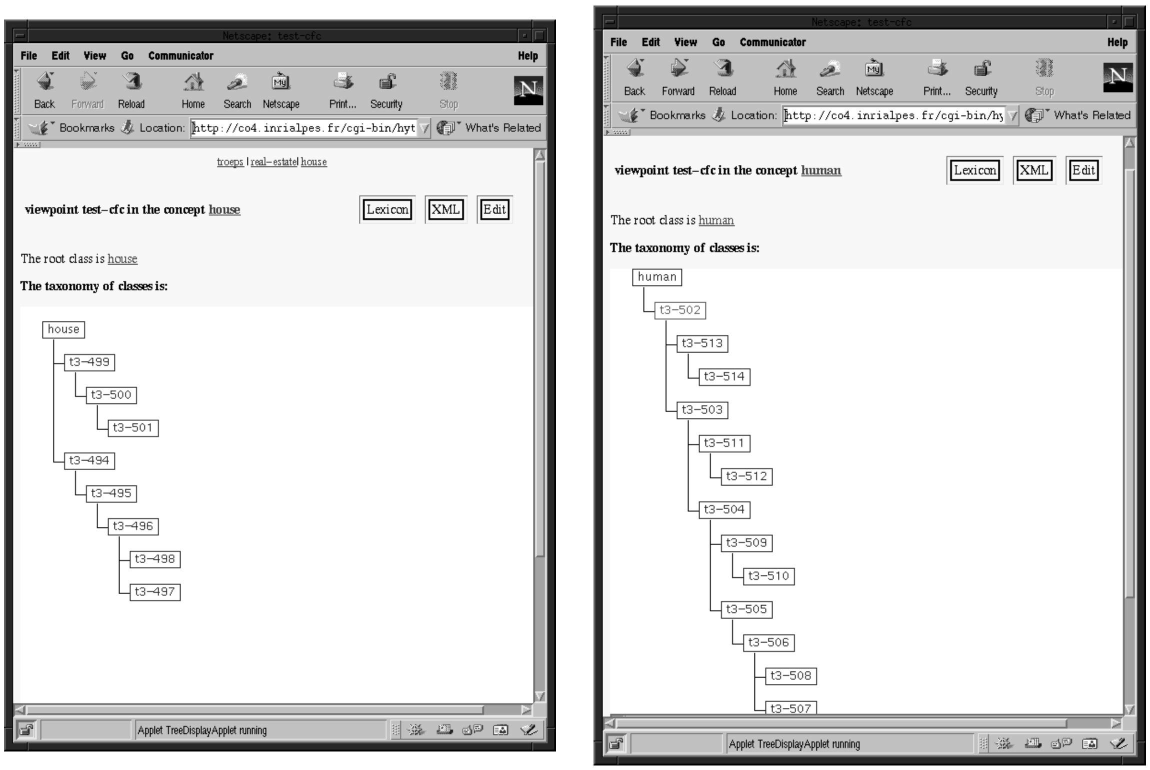


FIG. 9.6: *Écrans de visualisation des taxonomies de classes obtenues par regroupement.* ***À gauche****, les classes sur le concept* `appartement`. ***À droite*** *les classes sur le concept* `humain`.

Descriptions des classes

Pour mieux comprendre la nature du processus de construction automatique, il est intéressant d'examiner les descriptions des classes construites de manière automatique. Cela nécessite un examen des classes une par une. C'est sans doute aussi la stratégie qu'un utilisateur empruntera lors de la phase de validation pour les classes proposées par T-TREE. Sans que des outils particuliers soient dédiés à cette tâche, elle bénéficie d'une interface HTTP et donc d'un hypertexte qui permet un déplacement rapide entre les unités de la représentation.

Afin d'illustrer notre approche, nous avons choisi la seule classe `t3-509` qui est visualisée à gauche de la figure 9.7. Il est intéressant d'examiner ses attributs et plus particulièrement l'attribut `habitation` (`dwelling`) de la classe. L'attribut a pour type la classe `t3-499` du concept `appartement` (voir l'écran à droite de la figure 9.7). Celle-ci contient quatre objets dont trois sont 1, 3 et 4 qui sont exactement les habitations des trois personnes faisant partie de la classe `t3-509` : `Marco`, `Thomas` et `Karinne`.

Ici on voit se manifester l'influence entre époux que la prise en compte de l'attribut `conjoint` provoque. En effet, les trois objets cités ci-dessus sont exactement les conjoints des objets dans la classe `t3-513` (voir le paragraphe précédent). Pourtant, si on ne considérait pas les liens `conjoint`, les six objets en question auraient dû être répartis autrement : `Olaf` aurait dû se retrouver dans la même classe que `Marco`, `Thomas` et `Karinne`, laissant `Julie` et `Carla` seules dans une autre classe. Il suffit d'examiner des salaires et des habitatiions des six pour remarquer que les deux couples `Marco`, `Thomas` et `Julie`, `Carla` sont les plus similaires et par conséquent ils doivent former des classes fort cohérentes. De plus, ces classes sont contrastées à cause des fortes différences sur les salaires entre conjoints (les valeurs élevées dnas les cases correspondantes de la table 9.4 en témoignent). Pour découvrir toute la classification, il faut seulement trouver la quelle des deux classes chacun de `Karinne` et `Olaf` doit intégrer. Compte-tenu du fait que les deux classes en question sont constituées par les membres de deux couples conjugaux et que ceux-ci partagent les mêmes résidences, la différence sur l'attribut `habitation` n'est plus décisive pour le regroupement des deux derniers objets. Or, l'ordre des valeurs du seul attribut décisive, `salaire`, indique que les deux objets doivent se joindre à la classe de `Marco` et `Thomas`. Cette configuration ne se produit pas lorsque l'attribut `conjoint` intervient dans le calcul de la dissimilarité car les ressemblances de `Karinne` avec `Marco` et `Thomas` se repercutent sur les ressemblances de `Olaf` avec leurs épouses respectives et cela mène à la création de la classe `t3-513`.

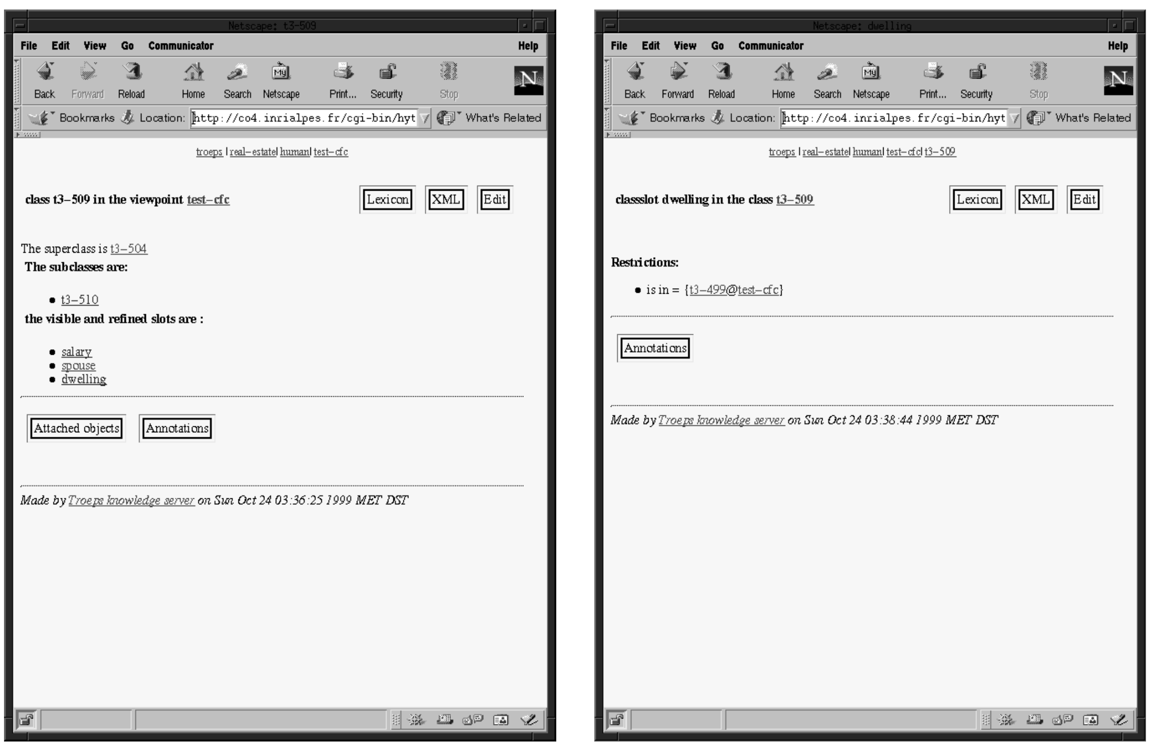


FIG. 9.7: *Une des classes obtenues par la construction (selon le scénario un). **À gauche**, la classe `t3-509` du concept `humain`. **À droite**, la restriction sur l'attribut `habitation` dans `t3-509` fait référence à la classe `t3-499` du concept `appartement`.*

9.3.2 Construction d'une taxonomie auto-référant

Dans la suite, nous montrons un exemple de construction de classes sur un concept auto-référant. Cette nouvelle tâche de construction de taxonomies correspond au second scénario : les personnes sont maintenant considérées indépendamment de leurs habitations.

Paramètres

Dans la suite du paragraphe, nous allons considérer le concept `humain` séparément des deux autres concepts. Les attributs concernés sont de nombre de trois : l'`âge`, le `salaire` et le `conjoint`. Les poids respectifs sont de 0,3, 0,3 et 0,4.

humain	
`âge`	0,3
`salaire`	0,3
`conjoint`	0,4

Ce choix conduit à la constitution d'un graphe de dépendances $\mathcal{K}$ qui ne comporte qu'un seul sommet, mais représente en soi une CFC non-triviale. Cela oblige un traitement à la manière des CFC de plusieurs membres. En particulier, la procédure de calcul de la dissimilarité passe par la résolution d'un système d'équations. La configuration est montrée à gauche de la figure 9.8.

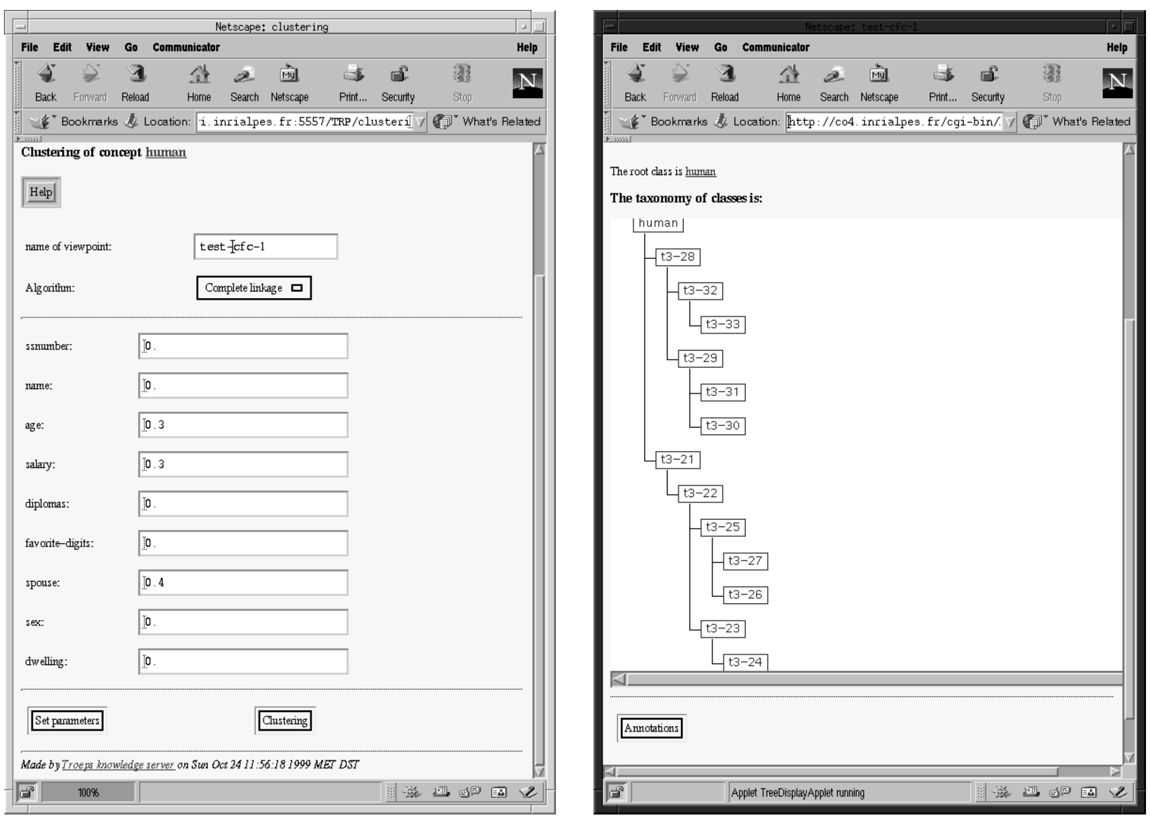


FIG. 9.8: ***Gauche*** *: écran de saisie des paramètres du regroupement sur le concept* `humain` *: les poids et l'algorithme de regroupement sont positionnés.* ***Droit*** *: taxonomie obtenue sur* `humain`*.*

Résultats

La taxonomie construite est visualisée à droite de la figure 9.8. Il est à noter le relatif équilibre dans la structure de la taxonomie qui est typique pour la méthode du lien complet. La taxonomie est de profondeur 4. Elle est à comparer au profondeur de la taxonomie du premier scénario (voir figure 9.6 à droite) qui est de 6. La bonne cohérence interne des classes par rapport aux liens `conjoint` peut être observé sur l'exemple des deux classes `t3-23` et `t3-32` (voir la figure 9.9). Les membres de la première sont les objets représentant `Sophie`, `Julie` et `Carla` qui sont toutes d'un âge proche de la trentaine et d'un salaire élevé, voire très élevé (le cas de Carla). L'attribut `conjoint` de la classe a pour type la classe `t3-32` dont les membres sont les objets représentant `Claude`, `Thomas` et `Marco` qui sont les conjoints respectifs des trois dames de la classe `t3-23`. Les trois partagent également des caractéristiques : âge entre 30 et 35, salaire assez élevé (mais inférieur à celui de leurs conjoints respectifs).

La découverte de ces deux classes mutuellement liées n'aurait pas pu se faire à l'aide d'une mesure non-récursive. Pour s'en apercevoir, il suffit de prendre le couple d'objets `(Manuel, Claude)` et de le comparer aux deux couples de la classe `t3-32` faisant apparaître `Claude` : `(Marco, Claude)` et `(Thomas, Claude)`. Dans les deux cas, la dissimilarité sur les attributs propriétés de `(Manuel, Claude)` est inférieure à la dissimilarité du couple de la classe. Cependant, le conjoint de `Manuel`, `Sonia`, est très peu semblable aux trois conjoints de la classe `t3-23`, ce qui explique le fait que `Manuel` se retrouve séparé de `Claude` à ce niveau de la taxonomie.

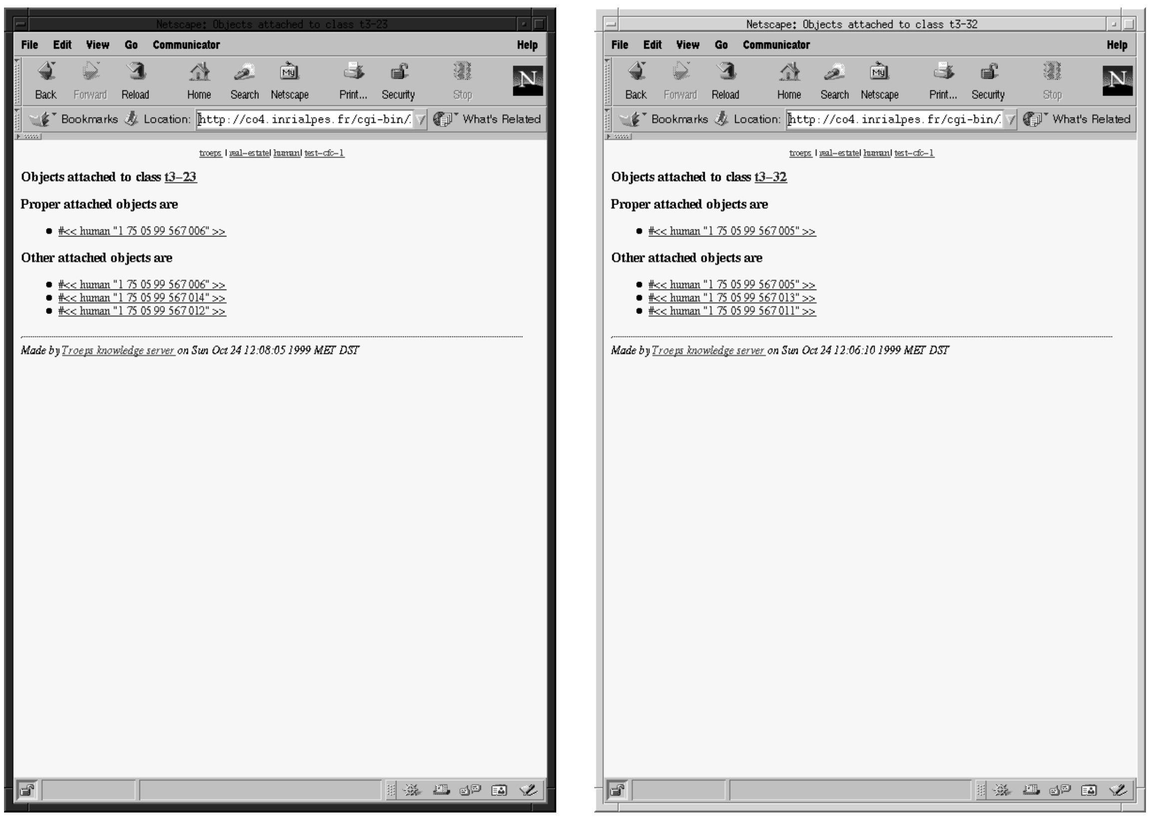


FIG. 9.9: *Écrans décrivant les extensions des classes* `t3-23` *et* `t3-32` *obtenues par regroupement automatique (scenario deux).*

9.3.3 Utilisation d'une taxonomie existante

Paramètres

Les poids des attributs concernés sont égaux : 0.5 pour chacun des deux attributs `salaire` et `habitation`. Comme dans les deux premiers cas, une procédure de type *lien complet* est utilisée.

La saisie est visualisée sur la figure 9.10 à gauche.

Résultats

La figure 9.10 à droite visualise le résultat de la construction pour le troisième scénario. On peut déjà observer une différence structurelle avec la taxonomie de la figure 9.5. La profondeur ici est de 5 contre 6 dans le premier scénario. De plus, la taxonomie sur la figure 9.10 est mieux équilibrée.

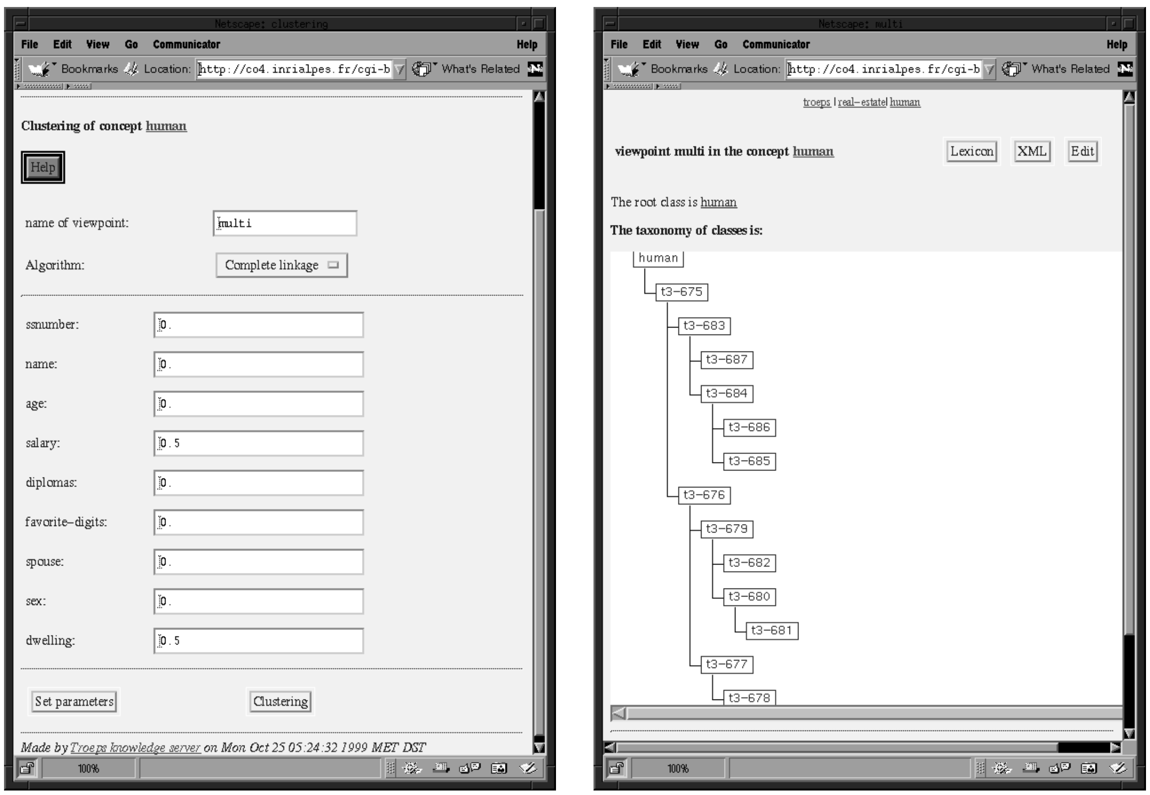


FIG. 9.10: ***Gauche :*** *écran de saisie des paramètres du regroupement sur le concept* **`humain`**, *l'attribut* **`habitation`** *est utilisé avec un poids de 0.5.* ***Droite :*** *la taxonomie produite par référence à la taxonomie déjà existante sur appartement (point de vue* `test-cfc` *qui peut être vu à gauche de la figure 9.6).*

Prenons par exemple la classe `t3-684` qui se trouve à une profondeur 3 dans la taxonomie construite. Les objets attachés à la classe sont : `Pierre`, `Anne`, `Yves` et `Léa`.

La valeur de l'attribut `habitation` de la classe `t3-684` indique la classe sur `appartement` dont les objets membres sont identifiés comme suit :

– "rue Monge", 3, 2,
– "grenoble", 32, 3,
– "grenoble", 21, 2,
– "meylan", 60, 6,
– "eybens", 13, 4,
– "eybens", 47, 0.

En effet, le premier et le troisième appartement sont ceux que les deux couples occupent. Cette fois-ci, la classe d'appartements référencée par une classe spécifique sur `humain` n'est pas elle-même une classe spécifique. Cela s'explique par le fait que les classes sur `appartement` ont été construites indépendamment de la valeur de l'attribut `habitants`. En conséquence, il n'y a aucune raison de se retrouver avec des classes spécifiques dans les valeurs de habitation des classes de la nouvelle taxonomie sur `humain`.

9.4 Calculs alternatifs

Dans la section précédente il a été question de l'application de T-TREE à des situations différentes et avec des objectifs différents. Les trois exemples cités ont mis en évidence la flexibilité de l'approche réalisée par T-TREE. Le but de cette section est d'illustrer les possibilités de faire varier les résultats du travail de T-TREE, c'est-à-dire la taxonomie de classes obtenues, en fonction des paramètres.

Pour cela, nous nous sommes limités à un seul concept, `humain` pris avec un lien de co-définition l'attribut `conjoint`. Les deux autres attributs sont `salaire` et `âge`. En d'autres termes, nous nous trouvons dans le second scénario de la section précédente. Deux types de variations sont présentés : l'un concerne les poids des attributs pour le calcul de la dissimilarité, alors que l'autre porte sur le type d'algorithme utilisé pour le regroupement

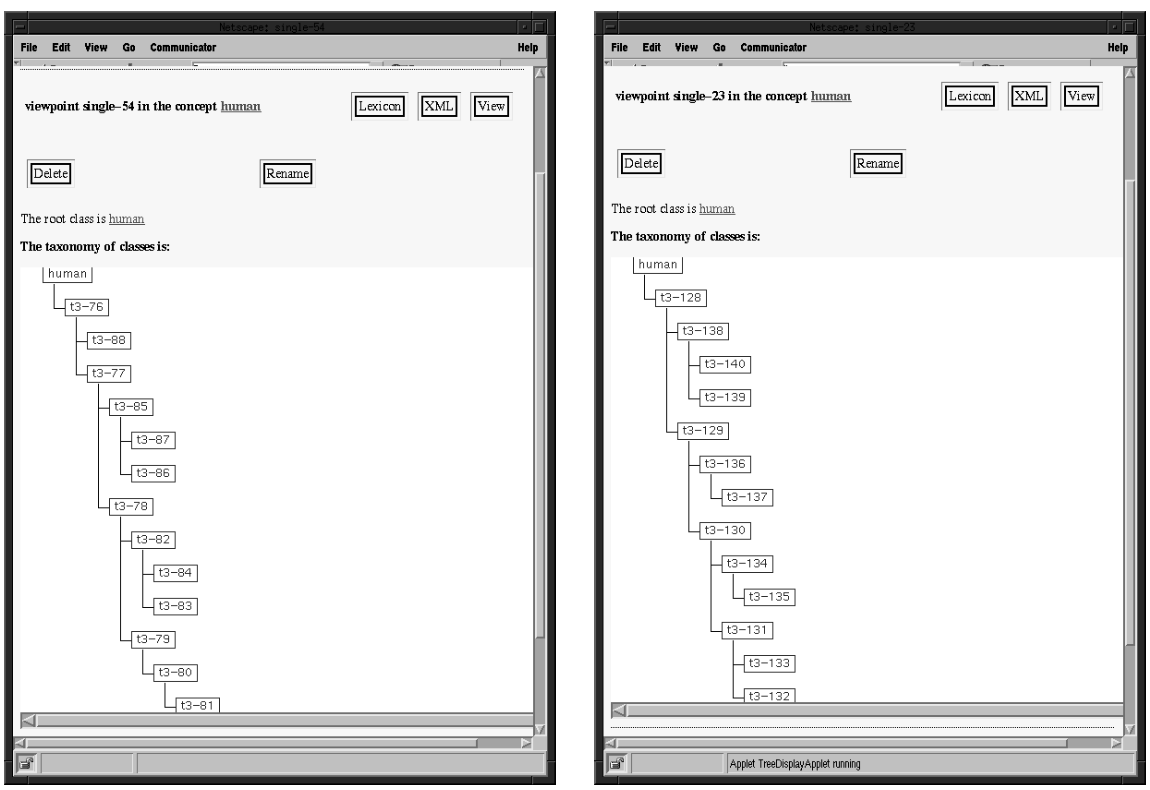


FIG. 9.11: *Les taxonomies sur* `humain` *obtenues à l'aide d'un algorithme de regroupement de type* ***lien complet*** *et en faisant varier les poids des attributs : à* ***gauche****, le résultat d'une construction avec des poids 0.2 pour* `conjoint` *et 0.4 pour chacun de deux autres attributs, à* ***droite*** *le résultat de la construction avec des poids 0.6 pour* `conjoint` *et 0.2 pour les deux autres attributs.*

9.4.1 Variation des poids

Lorsqu'une dissimilarité pondérée est utilisée pour la construction de taxonomies, une question se pose de manière immédiate : l'influence des poids sur la structure de la taxonomie obtenue. L'exemple que nous présentons ici n'a pas l'ambition de dévoiler des grandes tendances dans les rapports dissimilarité — taxonomie. La question est délicate et son élaboration mériterait le plus grand soin. Dans la section suivante nous proposons quelques pistes pour la conduite d'une étude structurée de la construction de classes d'objets par proximités.

Ici nous nous limitons de signaler le réel impact des paramètres sur le déroulement et le résultat du processus de construction. En d'autres termes, nous soulignons la possibilité pour l'utilisateur de guider ce processus par les paramètres fournis, tout en laissant de coté la question la plus intéressante (mais aussi plus difficile à répondre) de savoir comment ce guidage doit être effectué.

Sur la figure 9.11, les résultats d'une construction de type lien complet avec la dissimilarité paramétrée par des poids variés est visualisée. À gauche les poids des trois attributs `salaire`, `âge` et`conjoint` sont respectivement 0.4, 0.4 et 0.2, alors qu'à droite ce poids sont 0.2, 0.2 et 0.6. Ce que l'on pourra remarquer est que les deux taxonomies sont structurellement différentes. D'une part, leur profondeur des deux taxonomies est inégale : 6 à gauche et 5 à droite. D'autre part, l'équilibre entre les branches diverge aussi : à gauche une classe terminale est présente à profondeur 2, alors qu'à droite cette profondeur est de 3.

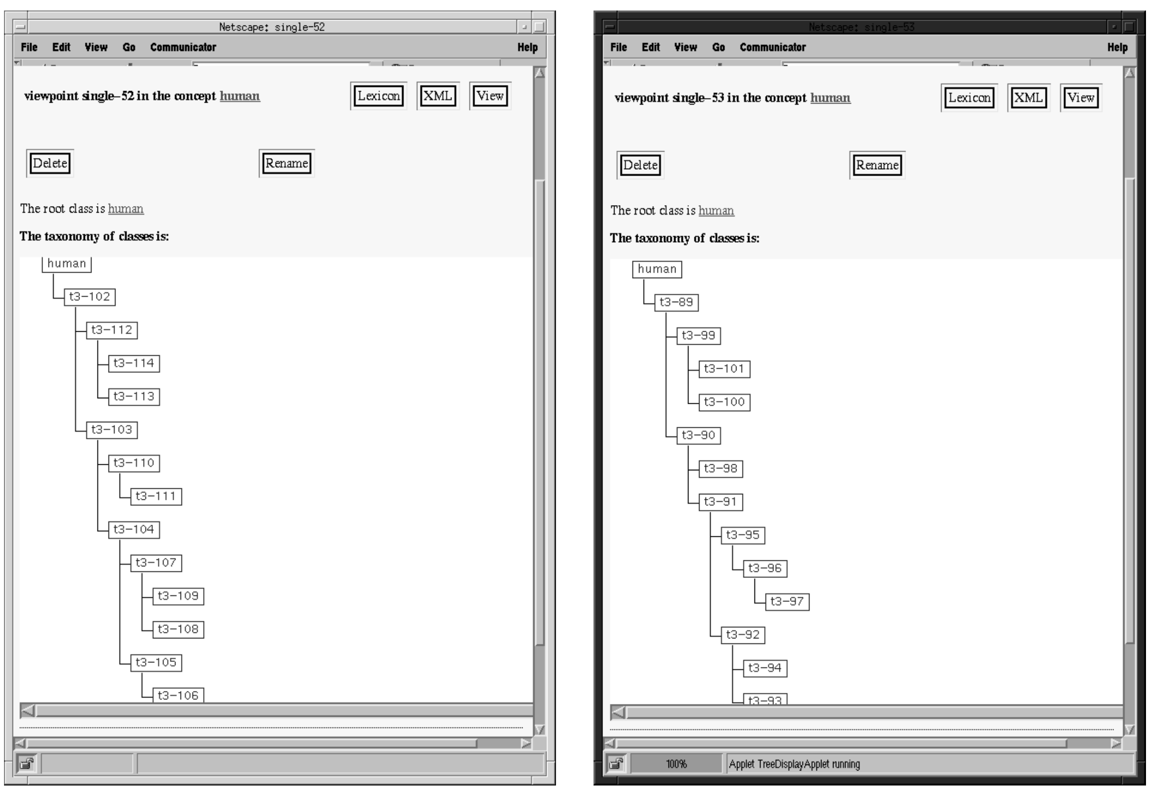


FIG. 9.12: *Les taxonomies sur* `humain` *obtenues en faisant varier l'algorithme de regroupement : à* ***gauche****, le résultat d'une application du* ***lien moyen****, à* ***droite*** *le résultat d'un* ***lien simple****.*

9.4.2 Variation des algorithmes de regroupement

Il convient de rappeler que pour l'instant, seuls les algorithmes de regroupement ascendant hiérarchique sont opérationnels dans T-TREE. Cela n'est pas pour autant handicapant lorsqu'il s'agit d'examiner l'impact des biais internes d'un algorithme sur le résultat final.

Pour donner un idée sur ces biais, nous avons fixé les poids des attributs à ceux utilisés dans la section 9.3.2 :

`humain`	
`âge`	0,3
`salaire`	0,3
`conjoint`	0,4

La figure 9.12 montre le résultat de la construction à l'aide d'un algorithme de type lien moyen (à gauche) et lien simple (à droite). Une simple inspection visuelle permet de constater les différences structurelles dans les deux taxonomies. À droite, on retrouve plusieurs sous-taxonomies ayant une structure en forme d'escalier (appelée encore un *peigne* dans le langage de l'analyse de données) qui représente un effet bien connu de l'heuristique du lien simple. En dépit de ces différences, une partie de la taxonomie reste commune : celle en dessous des deux classes `t3-99` et `t3-112`. En effet, les deux classes ont les mêmes extensions, c'est-à-dire l'ensemble des objets : `Anne`, `Yves`, `Karinne` et `Sonia`. Les quatre objets ci-dessus sont donc candidats à constitution d'une «forme forte» en analyse de données [Diday *et al.*, 1982].

9.4.3 Discussion

Les exemples cités ici ne sont qu'un cas particulier et il est difficile d'en tirer des conclusions sur le comportement conjoint des algorithmes de regroupement et de la nouvelle mesure d^o. Cependant, le fait de retrouver des tendances connues lors des tous premiers essais de l'approche a été jugé plutôt encourageant.

Bien évidemment, pour une étude plus détaillé de l'influence des algorithmes et/ou des mesures de dissimilarité sur la taxonomie finale, il sera nécessaire d'aller plus loin que le simple examen visuel de la structure d'arbre. En particulier, les places des objets au sein de la taxonomie doivent être considérées comme nous l'avons noté dans le chapitre précédent. Sur ce plan, une place toute nouvelle sera réservée aux liens existant entre les classes de la taxonomie, une particularité des taxonomies d'objets en représentation de connaissances.

9.5 Expérimentations possibles avec T-TREE

Dans cette section nous présentons quelques pistes pour les expérimentations préliminaires de notre méthode. Elle ne sont pas destinées à dévoiler des caractéristiques intrinsèques de la méthode de construction de classes car le jeu de données ne s'y prête pas tellement. Elles doivent plutôt donner une idée sur la nature des études qu'il est possible de mener à l'aide de l'outil conçu.

Parmi les buts des expérimentations citons :

Efficacité : La croissance du temps de traitement avec le nombre des objets à considérer est un aspect traditionnel des modules d'analyse automatique qui doit être analysé pour une utilisation raisonnable. Cependant, dans notre cas, un autre facteur prévaut, à savoir la structuration des objets. Curieusement, les expérimentations préliminaires ont démontré que les poids relatifs des attributs peuvent influencer le coût du traitement. En simplifiant, on pourra dire que le plus fort poids des attributs induisant une co-définition intraîne un plus grand nombre d'itérations nécessaires à la convergence du calcul. Sur ce point, il serait intéressant d'expérimenter avec des bases réalistes afin d'estimer l'impact du phénomène.

Impact de la structuration des objets : Bien que certaines études ont déjà été effectuées sur le sujet, il serait intéressant de savoir quelle est la dépendance effective entre le temps de construction d'une taxonomie et le degré de structuration dans les objets. Ce dernier peut être mesuré en fonction du chemin le plus long dans un réseau d'objets parcouru lors du calcul de dissimilarité.

Prédictivité : Bien que cela ne soit pas une fin en soi lors de l'organisation des connaissances, il peut être intéressant d'examiner le pouvoir prédictif des classes identifiées de manière automatique. **Protocole** (classique en apprentissage automatique [Fisher et Pazzani, 1991]) : séparation des objets en deux sous-ensembles, un ensemble d'*apprentissage* sur lequel le regroupement est effectué et ensemble de `test` qui sert à vérifier la prédictivité. La vérification consiste à classer les objets du second ensemble tout en masquant un ou plusieurs de leurs attributs. Un ensemble de classes *possibles* est donc obtenue pour chaque objet. La bonne adéquation entre les valeurs des attributs cachés de l'objet et les types de ces attributs dans les classes sélectionnées indique un fort pouvoir prédictif de la taxonomie.

Bénéfice de l'utilisation des classes pré-définies : Comparer les taxonomies sur un ensemble d'objets complexes obtenues en calculant les écarts sur leurs attributs relationnels de deux manières différentes : d'une part, par une dissimilarité de classes, δ^{cl} et d'autre part, par une dissimilarité d'objets, d^o. Les taxonomies peuvent être évaluées selon leur utilité pour l'utilisateur (critère subjectif), ou selon un critère mesurable : prédictivité, propriétés géométriques (largeur, profondeur, etc.), balance entre les extensions des classes sont juste quelques exemples de critères imaginables. L'éventuel gain dans un cas comme dans l'autre doit être considéré en lien avec les différences du temps de construction des deux sortes de taxonomies.

9.6 Conclusion

Ce chapitre a présenté un exemple d'application possible de T-TREE. Nous avons particulièrement insisté sur les fonctionnalités du système qui permettent de répondre aux exigences fixées dans la première partie de ce document. Tout, d'abord, la **richesse** des descriptions des objets a été illustrée à travers un large éventail d'attributs utilisés dans la description des objets manipulés (de nature, de type et de constructeur variables). L'évolution dans la structuration de la base et la flexibilité de T-TREE face à cette variabilité a été illustrée par les exemples de trois scénarios fort différents d'utilisation de T-TREE. Finalement, la possibilité de guider le processus à été démontrée à travers plusieurs variations dans les entrées du processus de construction. L'effet sur le résultat final a été observé. On peut remarquer en contrepartie la relative homogénéité de l'interface face au large éventail de tâches que le système aide à accomplir : pour une construction particulière, les choix de l'utilisateur se résument à un ensemble de poids d'attributs et un algorithme de regroupement, alors que pour un attribut relationnel pertinent, le choix est entre la création d'un nouveau point de vue et l'adoption d'un déjà existant.

En somme, nous considérons que les objectifs du départ ont été atteints, même si notre approche demande une vraie validation expérimentale. En effet, les expérimentations présentées ici ne tiennent pas lieu de validation. Elles illustrent cependant la complexité — mais aussi la flexibilité – inhérente au processus de construction de taxonomies tel que nous l'avons appréhendé.

Conclusion

Le formalisme des objets est un formalisme très puissant qui offre de multiples moyens pour représenter et manipuler des connaissances hautement structurées. La forte structuration des objets fait de leur manipulation une tâche délicate ouvrant ainsi un vaste champs d'application pour les divers mécanismes d'aide automatique. Le besoin d'aide est particulièrement prononcé lors de la conception d'une taxonomie de classes au sein de la base de connaissances et en présence d'un ensemble d'objets instances des classes à définir. Paradoxalement, cette même structuration semble être l'obstacle principal à l'application de méthodes d'analyse automatiques.

En effet, la grande majorité des méthodes d'analyse existantes a été conçue pour le traitement des ensembles d'individus faiblement structurés, décrits à l'aide d'un ensemble restreint de types de données, et considérés isolément par rapport à d'autres familles d'individus qui peuvent exister dans le domaine. Ce n'est que depuis peu que les formalismes plus structurés, logiques de description et graphes conceptuels ont fait leur apparition en apprentissage automatique, souvent encore traités sous une forme restreinte.

Cette tendance à limiter la complexité dans les descriptions des données est en fort contraste avec le soin que les SRC, en particulier les SRCO, portent à la richesse d'expression. En effet, une base de connaissances à objets représente un environnement où de multiples familles d'individus sont décrites par des attributs de types hétérogènes, dont une partie est construite et liée par des relations de nature variée. D'une part, cette richesse demande des outils nouveaux, suffisamment puissants pour extraire des régularités sur toute la vaste gamme de moyens de description dont une partie est ignorée en apprentissage automatique. D'autre part, la production d'une base opérationnelle passe par l'organisation des objets de chaque famille en une ou plusieurs taxonomies dont les classes sont aussi inter-reliées par des attributs. Toutefois, au cours de la construction, les différentes taxonomies ont des degrés de finition variables. La flexibilité par rapport à la quantité des connaissances disponibles sur le domaine est une exigence propre aux bases de connaissances que les méthodes d'analyse n'ont pas cherché à satisfaire.

Nous avons tendance à penser que le contraste souligné ci-dessus est le résultat d'un positionnement différent des deux disciplines par rapport au compromis inévitable entre l'expressivité des formalismes employés et la complexité des manipulations sur ceux-ci.

Dans ce travail nous avons adopté le point de vue des SRC en plaçant l'accent sur :

– le besoin d'analyser les objets sous leur forme d'origine, c'est-à-dire sans simplification ni transformation de leur description,
– la nécessité d'effectuer cette analyse dans l'environnement d'origine des objets qu'est la base de connaissances, et donc de tenir compte de l'ensemble des autres connaissances qui se trouvent à un moment donné dans la base,
– l'obligation d'adapter le processus de construction aux objectifs de l'utilisateur et donc de fournir à celui-ci des moyens pour guider le processus.

En conséquence, nous avons avancé la thèse que la définition d'un système d'aide à la conception de taxonomies passe par la satisfaction d'un ensemble d'exigences dont les plus conséquentes sont la **complétude** des outils mis en œuvre par rapport aux descriptions des objets, la **cohérence** de ces outils à l'égard de la sémantique des descriptions, la **paramétrabilité** de la méthode de construction ainsi que sa **flexibilité** vis à vis de l'état de structuration de la base de connaissances.

Contributions

Nous avons montré dans ce mémoire la faisabilité de la construction automatique de classes au sein d'un formalisme de représentation de connaissances à objets. Pour cela, nous avons défini une approche pour la construction qui permet de concilier la complexité structurelle des objets et l'efficacité d'une méthode classique de regroupement. Ainsi, plutôt que de créer une nouvelle méthode, nous avons fait appel à des techniques connues et validées de construction de classes tout en définissant un nombre d'outils et de mécanismes nécessaires à son application directe sur les objets.

L'**approche** proposée est fondée sur le principe de réduction progressive de la structure des objets en ses constituants. Ainsi chaque traitement sur une famille d'objets est décomposé de manière récursive en un ensemble de traitements sur les attributs de la famille. Les résultats de ces traitements sont ensuite combinés dans un résultat unique. Ceci permet non seulement de traiter avec succès la structure des objets, mais aussi de mettre à profit la structuration des objets pour accélérer l'analyse. Ainsi, les comparaisons des objets sont appuyées autant que possible sur les classes existantes afin d'éviter l'exploration exhaustive des structures d'objets. Cependant, l'utilisateur est libre de demander une recherche complète à travers cette structure si tel est son objectif. De cette manière, la flexibilité du processus de construction vis à vis de l'état de formalisation des connaissances de la base et de la confiance que l'utilisateur a dans son contenu est réalisée.

La **stratégie** de construction multiple correspond au besoin de travailler dans un environnement de multiples familles d'objets liées par des attributs relationnels. Elle permet d'optimiser l'effort de construction, mais aussi d'interprétation de la part de l'utilisateur. De plus, elle rend possible la construction de classes sur des objets co-définis, un problème qui n'a pas été abordé auparavant par les algorithmes de regroupement et assure ainsi la complétude de l'approche quant à la structure relationnelle dans les objets.

La **méthode** de construction effective de classes est la réalisation de notre approche. Elle effectue le regroupement des objets en classes et la caractérisation des classes de manière séparée moyennant quoi, elle est capable de construire des classes sur un ensemble de familles d'objets quelconques, y compris des familles en co-définition, et de refléter cette co-définition, dans les descriptions, ce qui est impossible par d'autres méthodes.

Les **outils** dont la méthode de construction fait usage reflètent la décomposition récursive des objets. Chaque outil est défini à deux niveaux : un niveau global sur les objets et un niveau local sur leurs attributs. La cohérence entre la sémantique des descriptions des objets et celle des traitements intervient dans la définition des opérateurs sur les attributs et leur combinaison au niveau des objets.

Outil principal de notre approche, la **mesure de dissimilarité** entre objets supporte le plus grand nombre d'éléments originaux. Ainsi, elle permet de traiter de manière simultanée plusieurs familles d'objets co-définies et de combiner des types variés selon un principe universel de comparaison. De plus, elle s'applique invariablement sur les types monovalués et multivalués.

Les **opérateurs de généralisation** sur les objets complètent l'éventail minimal d'outils pour la construction de classes. Leur mérite est de compléter les moyens offerts par le SRCO-hôte pour la caractérisation des classes.

Les outils ont été implantés au sein du SRCO Troeps. Des premières expérimentations ont montré la faisabilité de notre approche. La découverte des classes nouvelles, difficilement détectables avec les moyens traditionnels a mis en évidence son utilité.

Les limites de l'approche

Si notre approche présente de nombreux intérêts, elle possède également des limites. La première limite est liée à la complexité algorithmique des méthodes de construction de taxonomies conjointes. Cette complexité est, en grande partie, due à la résolution du système d'équations lors de l'établissement des valeurs de la dissimilarité. Cependant, lorsqu'aucune co-définition ne se présente, notre approche a la complexité de la méthode concrète de construction de classes choisie. Il y a donc ici un compromis entre la complexité structurelle des données que l'on veut traiter et le coût algorithmique qu'il est raisonnable de payer pour cela.

Une autre limitation de l'approche est liée à la nature des taxonomies qu'elle permet de construire. En effet, nous avons toujours considéré que les objets à regrouper partageaient l'ensemble de leurs attributs. Or, dans beaucoup de cas, certains attributs ne sont présents que dans un sous-ensemble des objets. Cela semble poser un problème majeur devant l'application directe de l'approche proposée ici. Cependant, dans une représentation par objets, un attribut est toujours introduit dans une classe dont les objets membres possèdent tous l'attribut (si les exceptions sont interdites). Il s'agit donc en premier lieu de découvrir ces classes génératrices d'attributs qui peuvent ensuite être raffinées par les techniques exposées ici. Le problème de la détection des classes génératrices au sein d'un ensemble d'objets est loin d'être trivial car ces classes peuvent être liées par une relation de spécialisation et former une taxonomie propre. Il s'agit donc d'une autre tâche de construction de taxonomie qui se base non plus sur les valeurs des attributs, mais sur la présence/absence de ces attributs. Des travaux en génie logiciel objet sur ce type particulier de regroupement ont démontré l'utilité des techniques de construction de treillis de Galois au sein d'un modèle à objets. Par ailleurs, la comparaison entre les deux approches du regroupement, les treillis et les mesures de proximité, constituent une piste de recherche très intéressante (voir partie Perspectives).

Perspectives

Notre travail a abouti à l'identification de certains problèmes intéressants concernant les méthodes de regroupement de données structurées et leur intégration au sein d'un SRC.

Méthodes de construction de classes

Complétion de la méthode par proximités. Les méthodes de regroupement utilisant un critère global de qualité se sont révélés inadaptées au contexte des objets à cause de leur dépendance envers les descriptions des classes. Nous leur avons préféré une méthode par proximité qui ne nécessite pas de critère global. Ce n'est pas pour autant qu'un tel critère est inutile. Une piste de recherche future sera de compléter la méthode avec un critère que l'on puisse mesurer sur les classes construites. Cela permettra une confrontation directe entre taxonomies et donc un meilleur résultat du regroupement. Toutefois, le critère recherché devra satisfaire une double condition. D'une part, il doit revêtir une certaine flexibilité afin de s'adapter aux exigences particulières d'un concepteur. D'autre part, seuls les critères en adéquation avec le reste des outils de construction peuvent avoir un succès.

Comparaison entre méthodes. Nous avons remarqué qu'il n'existe pas de méthode de regroupement universelle et que chacune des méthodes existantes s'adapte bien à certaines situations. De plus une certaine complémentarité entre les techniques de construction de treillis et les techniques à base de proximités a été mise en évidence. Il serait intéressant de mener une étude comparative des deux approches pour mieux délimiter les situations pour lesquelles chacune des approches est mieux adaptée. En particulier, l'adaptation de chaque technique aux trois formalismes de représentation de connaissances structurées, les objets, les graphes conceptuels et les logiques de description sera à examiner puis à confronter.

Combinaison entre méthodes. La complémentarité entre ces deux techniques a déjà été exploitée au sein des systèmes de regroupement [Girard, 1997]. A notre avis, la collaboration entre treillis et mesures de proximité au sein d'un même processus de regroupement peut être envisagée sous deux angles différents.

D'une part, les techniques par proximité peuvent s'appliquer en pré-traitement, avant la production du treillis, afin de fournir à cette dernière un ensemble de classes compactes d'individus ; le treillis sera ensuite construit sur ces classes et non sur la collection d'individus originale. Cette manière de procéder présente le double avantage de pouvoir réconcilier la construction de treillis avec les données numériques (par une réduction de la complexité liée au découpage en classes) et de pouvoir traiter les descriptions circulaires de manière directe.

D'autre part, les mesures de proximité peuvent trouver une application lors de l'étape de simplification des treillis pour l'extraction de structures de taille raisonnable [Girard, 1997] qui est une étape de post-traitement. Dans cette deuxième situation, une mesure de dissimilarité entre classes s'appliquera. Une mesure comme celle que nous avons esquissée au chapitre 5, mis à part sa capacité à traiter les co-définitions entre classes directement, représente un outil complet sur les moyens de description pour les classes.

Extensions de la méthode sur d'autres formalismes de représentation

Il serait intéressant d'évaluer la capacité d'application de notre approche sur d'autres formalismes structurés. Les bases de données orientées-objet constituent un domaine voisin de la RCO de par le partage du paradigme objet. Notre méthode devrait pouvoir s'adapter sans trop de difficultés dans un contexte de base de données et ainsi servir à des objectifs de fouille de données sur des objets [Han *et al.*, 1998; Han *et al.*, 1994], en particulier sur des objets fortement structurés [Faid *et al.*, 1999]. Elle peut également s'appliquer à la conception et la ré-ingénierie des schémas des BDOO [Astudillo, 1997]. Toutefois, des problèmes liés à la taille des ensembles d'objets à regrouper devront être affrontés.

Dans le champ de la représentation de connaissances, les logiques de description semblent particulièrement proches des objets et donc un bon candidat pour une extension de notre approche. Des premiers travaux ont été effectués sur l'analyse d'individus complexes à l'aide du système KLUSTER [Kietz et Morik, 1994]. Étendre les techniques développées dans un cadre supervisé pour des tâches non-supervisées est une première piste de recherche.

Applications

L'approche développée nécessite une validation par des applications issues de domaines à structure complexe, c'est-à-dire possédant des individus de multiples familles, liés par des relations, éventuellement multivalués et circulaires. Un domaine possible est la ré-ingénierie dans le génie logiciel objet. Notre approche semble particulièrement adaptée au traitement des circularités qui peuvent être rencontrées dans les descriptions des objets et des classes, non seulement en représentation de connaissances et BDOO, mais aussi en conception par objets.

Des travaux basés sur des techniques de treillis de Galois ont déjà été entamés par plusieurs équipes (voir [Godin *et al.*, 1995a; Godin *et al.*, 1998; Missaoui *et al.*, 1998] et [Dicky *et al.*, 1994; Dicky *et al.*, 1996]). Toutefois, ces techniques rencontrent des difficultés considérables lorsqu'il s'agit de regrouper des classes co-définies. Les remèdes proposés ne sont pas encore validés [Simon, 1999]. Le rôle des techniques basées sur la proximité sera double. D'une part, elles permettront d'effectuer une étape de pré-traitement pendant laquelle des classes co-définies sont formées selon des critères de similitude. Un treillis de Galois pourra être obtenu plus facilement à partir de l'ensemble de classes initiales. D'autre part, une version de notre mesure de dissimilarité peut être utilisée lors d'une étape de post-traitement à des fins de simplification de la structure globale du treillis.

Une seconde application de nos techniques pourrait être la catégorisation des documents structurés avec une représentation formelle de leur contenu. Un certain nombre de travaux sur la représentation du contenu à l'aide de formalismes structurés laissent prévoir un besoin d'outils d'organisation pour les ensembles de documents [Martin et Eklund, 1999; Fatemi et Mulhem, 1999]. Le développement de la norme XML pour les documents électroniques structurés devrait déboucher vers une forte demande d'outils d'analyse. C'est un autre champ où notre méthode peut constituer une aide précieuse.

Annexes

A

Fonctions utilisées dans les algorithmes

On présente ci-dessous succinctement la liste des pseudo-fonctions utilisées dans le texte du mémoire et dans les algorithmes. Pour celles dont un équivalent dans l'interface fonctionnelle (API) de TROEPS existe, nous avons fourni cet équiuvalent. Le reste des primitives est facilement calculable.

Nom	**Paramètres**	**Résultat**
`obj-classe-attach()`	o un objet, pv un point de vue	c la classe d'attachement de l'objet sous le point de vue
`obj-classes()`	o un objet, pv un point de vue	C l'ensemble de toutes les classes de l'objet sous le point de vue
`obj-concept()`	o un objet	k le concept de l'objet
`cl-superclasse()`	c une classe	c' la super-classe de la classe
`cl-sousclasses()`	c une classe	C l'ensemble de toutes les sous-classes de la classe
`cl-attributs()`	c une classe	A l'ensemble des attributs définis et hérités dans la classe
`cl-objets()`	c une classe	O l'ensemble de tous les objets membres de la classe
`cl-objets-directes()`	c une classe	O l'ensemble de tous les objets directement attachés à la classe
`cl-concept()`	c une classe	k le concept de la classe
`cl-point-de-vue()`	c une classe	pv le point de vue de la classe

TAB. A.1: *Fonctions sur les classes et les objets*

Nom	**Paramètres**	**Résultat**
`pv-concept()`	*pv* un point de vue	le concept du point de vue
`pv-classe-racine()`	*pv* un point de vue	la classe racine du point de vue
`pv-attributs()`	*pv* un point de vue	A l'ensemble des attributs visibles sous le point de vue
`cpt-points-de-vue()`	k un concept	P l'ensemble des points de vue du concept
`cpt-objets()`	k un concept	O l'ensemble de tous les objets instances du concept
`cpt-attributs()`	k un concept	A l'ensemble des attributs du concept

TAB. A.2: *Fonctions sur les points de vue et les concepts*

B

Algorithmes

B.0.1 Calcul des CFC

Comme dans le cas des concepts, il est nécessaire de rechercher les CFC que deux réseaux d'objets ont en commun, $\mathcal{CFC}_{|A}(o)$ et $\mathcal{CFC}_{|A}(o')$. Pour chaque objet o_i dans $\mathcal{CFC}_{|A}(o)$, $CircUni(o_i)$, l'ensemble des relations qui mènent à des objets de la CFC est également calculé. Cette partie de l'approche algorithmique n'est pas exposée ici car elle est relativement classique.

B.0.2 Calcul des couples d'objets

Pour le calcul de $Couples_{|A}(o, o')$, nous utilisons un algorithme récursif (*cf.* Algorithme 13).

Il nécessite la connaissance des ensembles d'objets dans les CFC et, pour chaque objet o_i, l'ensemble de ses attributs qui le lient à d'autres objets dans la CFC, soit $CircUni(o)$. Son résultat, $Couple$, est l'ensemble de tous les couples d'objets dont les dissimilarités sont mutuellement dépendantes. Les membres de $Couples$ vont participer à la formation du système d'équations en donnant chacun suite à une variable.

L'algorithme 13 effectue un parcours en profondeur d'abord des deux CFC en parallèle à la recherche de tous les couples qui se trouvent sur un circuit commun. Il utilise une pile P pour stocker l'ensemble des couples rencontrés au long d'un chemin. Lors de la détection d'un circuit, l'ensemble des couples sur la pile est affecté à $Couples$. La détection d'un circuit utilise deux conditions : l'aboutissement au couple d'origine ou bien à un couple dont on sait déjà qu'il se trouve sur un circuit (donc dans $Couples$). L'arrêt de l'algorithme est assuré en surveillant les couples déjà visités.

L'algorithme principal a pour but d'initialiser les structures globales et d'appeler une procédure récursive qui effectue l'essentiel du traitement(*cf.* Algorithme 14).

La composition du SEL, Algorithme 15, est en réalité le remplissage de la matrice de coefficients, $Coeff[i, j]$. Elle se fait en parcourant l'ensemble $Couples_{|A}(o, o')$. A chaque couple, on remplit la matrice de coefficients de la manière suivante. Le coefficient $Coeff[i, 0]$ est la somme pondérée des écarts sur les attributs qui ne participent pas au calcul récursif. Les coefficients $Coeff[i, j]$ sont les sommes des poids des attributs, conformément à la formule 7.4. En même temps, l'élément du vecteur *Liens* correspondant à une variable est actualisé : celui-ci contient, pour chaque variable, l'ensemble des autres variables dont elle dépend (via un attribut relationnel entre les couples respectifs). Ce vecteur simplifie la résolution du système car il permet de ne prendre en compte que les dépendances effectives entre variables. Une division de tous les coefficients sur une ligne par la somme des poids des attributs boucles est effectuée à la fin. Après la terminaison de l'algorithme, la matrice $Coeff[i, j]$ contient la fonction $\mathbf{F}$ décrite dans

RECHERCHE-COUPLES(o, o', $CircUni()$)

- entrées : o, o' - objets
- : $CircUni()$ - une fonction des objets dans les ensembles d'attributs
- sorties : $Couples$ - un ensemble de couples d'objets

- variables : P - une pile
- : $Visites$ - un ensemble de couples d'objets

$init(P)$
$empiler(P, (o, o'))$
$Couples := \{(o, o')\}$
$Visites := \{(o, o')\}$
pour tout $a \in CircUni(o) \cap CircUni(o')$ **faire**
    RECHERCHE-REC($o.a, o'.a, o, o'$,$CircUni()$)
$depiler(P)$
retourner $Couples$

Algorithme 13: Calcul des couples des objets dans $Couples_{|A}(o, o')$

RECHERCHE-REC(o_1, o_2, o, o',$CircUni()$)

- entrées : o_1, o_2, o, o' - objets
- : $CircUni()$ - une fonction des objets dans les ensembles d'attributs

si ($o_1 = o$ et $o_2 = o'$) ou $(o_1, o_2) \in Couples$ **alors**
    {aboutissement à un circuit, sauvegarde de la pile}
    pour tout (o_i, o_j) sur la pile P **faire**
        $Couples := Couples \cup \{(o_i, o_j)\}$
sinon
    si $(o_1, o_2) \notin Visites$ **alors**
        $Visites := Visites \cup \{(o_1, o_2)\}$
        $empiler(P, (o_1, o_2))$
        pour tout $a \in CircUni(o_1) \cap CircUni(o_2)$ **faire**
            RECHERCHE-REC($o_1.a, o_2.a, o, o'$,$CircUni()$)
        $depiler(P)$

Algorithme 14: Procédure de recherche récursive la section 7.3.4.

```
COMPOSE-SEL(Couples, Attr(), Poids())

- entrées : Couples - un vecteur de couples
-           : Attr() - un vecteur sur les concepts d'ensembles d'attributs pertinents
-           : Poids() - un vecteur de poids d'attributs
- sorties   : Coeff[][] - une matrice n × (n + 1) de réels
-           : Liens[] - un vecteur d'ensembles d'entiers

- variables : Auto - un réel

pour tout (o1, o2) ∈ Couples faire
  soit i l'indice de (o1, o2) dans Couples
  pour j := 0 à n faire
    Coeff[i, j] := 0
    Liens[i] := ∅
  Auto := 1
  pour tout a ∈ Attr(concept(o1)) faire
    si o1.a = o1 et o2.a = o2 alors
      Auto := Auto − Poids(a)
    sinon
      si o1.a = o3 et o2.a = o4 et (o3, o4) ∈ Couples alors
        soit k l'indice de (o3, o4) dans Couples
        Coeff[i, k] := Coeff[i, k] + Poids(a)
        Liens[i] := Liens[i] ∪ {k}
      sinon
        Coeff[i, 0] := Coeff[i, 0] + Poids(a) ∗ DISSIM(o1.a, o2.a)
  pour j := 0 à n faire
    Coeff[i, j] := Coeff[i, j]/Auto
```

Algorithme 15: Composition du SEL

B.0.3 Complexité

Il est intéressant de pouvoir évaluer, bien que de manière très approximative, le coût du traitement des circularités. Pour cela, le calcul des dissimilarités entre deux couples est évalué d'abord. Soient :

- n le nombre des objets du concept C à considérer,
- n_o le nombre moyen d'objets sur une CFC correspondante à un objet de C,
- n_e le nombre des couples effectifs d'objets dans $Couples_{|A}(o, o')$, n_e varie entre n et n^2,
- n_t le nombre total des couples atteinables depuis (o, o'), n_t suit les mêmes variations que n_e,
- n_a le nombre moyen d'attribut dans l'ensemble $Circ(o_1, o_2)$ pour un couple $(o_1, o_2) \in Couples_{|A}(o, o')$; ce nombre ne dépend pas de n, il est plutôt une caractéristique propre au domaine étudié,

– l_c la longueur moyenne d'un circuit commun dans les réseaux l_c varie entre n et n_c.

La complexité de la recherche de $Couples_{|A}(o, o')$ est évaluée comme suit. Il faut d'abord évaluer le nombre d'appels à la routine RECHERCHE-REC qui effectuent réellement un traitement. Ceux-ci sont exactement le nombre de couples n_t car chaque couple ne peut être traité qu'une fois[35]. En revanche, il peut y avoir un plus grand nombre d'appels de RECHERCHE-REC, mais le reste n'auront pas de conséquence (un couple déjà rencontré est soit dans $Couples$ car intéressant, soit $Visites$). Le nombre total d'appels est de l'ordre de $O(n_t * n_a)$. On a donc un nombre n_t d'appels effectifs à RECHERCHE-REC. Au sein d'un appel, le test de présence dans $Couples$ ou dans $Visites$ est *a priori* en $O(n_e)$, mais peut être implanté efficacement en $O(n_o)$[36]. Avec cette même implantation, l'ajout d'un couple à $Couples$ ou à $Visites$ peut être fait en $O(n_o)$. Il reste l'intersection entre ensembles d'attributs qui est en $O((n_a)^2)$. Globalement, la complexité est donc de l'ordre de $O(n_t * n_a * n_o)$ qui, grossièrement surestimé, revient à $O((n_o)^3)$.

De la même manière, la complexité de la composition du SEL est de $O(n_t * |A|)$. Ici nous ne prenons pas en compte le surcoût lié au calcul des écarts qui ne donnent pas suite à des variables (fonction $DISSIM()$). La résolution est faite en $O(n_t * i * n_a)$ où i est le nombre des itérations effectuées.

35. la présence de $Visites$ l'assure

36. à chaque objet de la première CFC on associe deux listes : une des objets de la deuxième CFC avec qui il a déjà formé un couple dans $Couples$ et une autre liste, similaire, pour $Visites$

C

Preuves

C.1 Preuves des propriétés du chapitre 4

Proposition 4.1. *Pour un ensemble de collections* $\{z_1, z_2, .., z_n\}$ *sur un type de base* τ*, si sa généralisation est* $gen_m(\{z_1, z_2, .., z_n\}) = \langle [a_l \; a_h], [g_l \; g_h] \rangle$*, alors* $g_l \; \leq_\tau g_h$.

Preuve. Par définition :

$$g_l = gen_\tau(e_{\min})$$

et

$$g_h = gen_\tau(e_{\max}).$$

L'opérateur de généralisation sur τ est un opérateur sur un type monovalué, il est donc bien défini et monotone vis-à-vis de l'inclusion entre ensemble d'éléments de τ (par définition). Il reste à remarquer que l'ensemble minimal est toujours inclus dans l'ensemble maximal :

$$e_{\min} \subseteq e_{\max}.$$

En conséquence,

$$g_l = gen_\tau(e_{\min}) \; \leq_\tau gen_\tau(e_{\max}) = g_h.$$

□

Proposition 4.2. *Pour deux ensembles* A *et* A' *de collections sur le type de base* τ*,* $A \subseteq A'$ *implique* $gen_m(A) \leq_{\mu(\tau)} gen_m(A')$.

Preuve. Le seul point délicat est la preuve de la condition sur les généralisations :

$$g'_l \; \leq_\tau \; g_l \; \leq_\tau \; g_h \; \leq_\tau \; g'_h$$

Il est tout de même aisé de démontrer que l'inclusion de A dans A' se traduit par une inclusion des ensembles minimaux et maximaux respectifs :

$$e'_{\min} = \bigcap_{z' \in A'} elements(z') \subseteq e_{\min} = \bigcap_{z \in A} elements(z)$$

et de manière duale :

$$e_{\max} = \bigcup_{z \in A} elements(z) \subseteq e'_{\min} = \bigcup_{z' \in A'} elements(z')$$

En conséquence, la propriété recherchée suit directement de la propriété 4.1 et de la définition 4.10 du chapitre 6. □

Proposition 4.3. *Pour un couple d'ensembles d'objets O et O' en provenance d'un concept k, si $O \subseteq O'$, alors $GEN_o(O) \leq_k GEN_o(O')$.*

Preuve. Soit $O = \{o_1, o_2, .., o_n\}$ et $O' = \{o'_1, o'_2, .., o'_m\}$. Pour tout a dans $A = attributs(k)$, les ensembles des valeurs de a dans les objets de O et O' vérifient :

$$\omega_a = \{o_1.a, o_2.a, .., o_n.a\} \subseteq \{o'_1.a, o'_2.a, .., o'_m.a\} = \omega'_a.$$

Soit $\tau(a) = type(a)$. En utilisant la correction des opérateurs sur les attributs, on obtient :

$$gen_a(\omega_a) \leq_{\tau(a)} gen_a(\omega'_a).$$

En composant sur le produit de tous les attributs, on arrive au résultat recherché. Tout d'abord, le produit des ordres préserve l'ordre entre les dimensions :

$$\times_{a \in A} gen_a(\omega_a) \leq_{\tau(a)} \times_{a \in A} gen_a(\omega'_a).$$

Ensuite, les deux parties sont exactement les généralisations des objets dans O et O' respectivement. On en déduit qu'elles sont liées par la relation de sous-typage (et non pas de spécialisation qui doit être explicitement spécifiée par un concepteur) :

$$GEN_o(O) \leq_k GEN_o(O')$$

□

C.2 Preuves des propriétés du chapitre 6

Pour des raisons de simplification, cette section débute par quelques notations utiles pour la suite.

Soit la fonction δ^* une dissimilarité d'ensembles de type de base τ et de domaine $D = dom(\tau)$. La mesure δ^* est définie sur les triplets (ensemble, ensemble, appariement). Formellement,

Définition C.1. *La fonction $\delta^* : 2^D \times 2^D \times 2^{D \times D} \rightarrow [0\ 1]$ est une généralisation de δ^m car elle peut prendre n'importe quel appariement en paramètre au lieu de calculer l'optimum :*

$$\delta^*(s, s', \bar{m}) = \frac{\sum_{(e,e') \in \bar{m}} \delta_D(e, e') + abs(p - q)}{\max(p, q)}$$

où s et s' sont deux ensembles, m est un appariement exclusif quelconque entre s et s' et p et q sont les cardinalités de q et q' respectivement.

Les liens suivants entre les deux mesures sont à noter :

$$\delta^*(s, s', M_{opt}(s, s')) = \delta^m(s, s').$$

De plus, pour tout appariement m exclusif et maximal en cardinalité, les valeurs de δ^* sont au moins égales à celle de δ^m.

$$\delta^m(s, s') \leq \delta^*(s, s', m).$$

Nous allons noter de manière succincte le poids d'un appariement m.

Définition C.2. *Soit m un appariement exclusif, non nécessairement de cardinalité maximale, entre deux collections z et z'. Son poids est noté par :*

$$\sum_m (s, s') = \sum_{(e,e') \in m} \delta_D(e, e').$$

Cela nous permet d'exprimer la mesure δ^m avec la même concision :

$$\delta^m(s,s') = \frac{\sum_{M_{opt}(s_1,s_2)}(s_1,s_2) + |p-q|}{\max(p,q)}.$$

Précisons que les appariements exclusifs que nous utilisons ici définissent des applications injectives entre les ensembles. Nous pouvons donc leur donner un sens et les considérer comme des fonctions totales de l'ensemble de plus petite cardinalité vers l'ensemble de plus grande cardinalité. Ainsi, soient s et s', deux ensembles sur un domaine D de cardinalités p et q respectivement et supposons que $p \leq q$. Soit encore m un appariement exclusif entre s et s'. On pourra noter $m : s \rightarrow s'$. Les notations suivantes sont correctes :

Définition C.3. *Pour le couple s et s' muni de l'appariement m, nous considérons :*

- *le **domaine** de m, noté $m^{-1}(s') = dom(m) \subseteq s$,*
- *le **co-domaine** de m, noté $m(s) = cod(m) \subseteq s'$,*
- *la **restriction** de m à un sous-ensemble s^* de son domaine, soit $m_{|s^*}$.*

Proposition 5.1. *Si la fonction δ_D est une semi-métrique, alors δ^m l'est aussi.*

Preuve. La manière dont la mesure δ^m est obtenue à partir de δ_D pour un couple de collections z et z' est :

$$\delta^m(z,z') = \frac{\sum_{(e,e') \in M_{opt}(z,z')} \delta_D(e,e') + |p-q|}{\max(p,q)} \tag{C.1}$$

Rappelons que le but est de démontrer les propriétés suivantes sur δ^m pour tout couple de collections z et z' :

0)	$\delta^m(z,z')$	$\geq$	0	(non négative)
1)	$\delta^m(z,z)$	$=$	0	(minimalité)
2)	$\delta^m(z,z')$	$=$	$\delta^m(z',z)$	(symétrie)
3)	$\delta^m(z,z')$	$=$	0 implique $z = z'$	(identité)

Tout d'abord, les valeurs de δ^m sont non-négatives, car elles sont des sommes de termes qui sont non-négatifs. La fonction est donc une dissimilarité selon les critères de [van Cutsem, 1994].

Ensuite, sur deux collections identiques, la valeur de la mesure est 0. En effet, le meilleur appariement (il ne peut y en avoir qu'un) consiste à prendre des couples d'éléments identiques qui ont une dissimilarité de 0 car δ_D satisfait 1).

De plus, l'ordre entre les arguments n'intervenant pas, la fonction est *symétrique*.

Finalement, il est aisé de montrer que si la valeur de la fonction est 0 alors les deux collections sont identiques. Remarquons d'abord que la fonction s'annule si le numérateur s'annule. Ainsi, on peut affirmer que les deux collections ont une différence de cardinalité nulle et un poids du meilleur appariement qui est également nul. Elles ont donc des tailles égales et il existe un appariement pour qui tous les couples d'éléments ont une dissimilarité nulle. Or, la mesure δ_D satisfait la propriété 3) ci-dessus. On en déduit donc que les éléments appariés sont identiques. Et puisqu'il n'y a pas d'élément non apparié (à cause de la condition sur les cardinalités qui est nulle), alors on peut conclure que les deux collections sont identiques à leur tour. En conséquence, δ^m satisfait la propriété d'identité.

En somme, la fonction est une *semi-métrique* □

Proposition 5.2. *La valeur de la fonction δ^m décroît lorsque l'on ajoute un élément au plus petit ensemble.*

Preuve. Soient trois ensembles s_1, s_2 et s_3 sur un type de base à domaine D tels que :

1. $\|s_1\| = p$, $\|s_1\| = q$ et $p < q$
2. $s_3 = s_1 \cup \{x\}$ où $x \in D$

Notre but est de montrer que :

$$\delta^m(s_2, s_3) \leq \delta^m(s_1, s_2)$$

et cela, indépendamment de D et du choix de x dans D.

Soient les deux appariements optimaux $m = M_{opt}(s_1, s_2)$ et $m' = M_{opt}(s_2, s_3)$.

L'idée de la preuve est de reconstituer un appariement $\bar{m} = M(s_2, s_3)$ tel que, lorsqu'on calcule une dissimilarité entre s_2 et s_3 selon $\bar{m}$, soit $\delta^*(s_2, s_3, \bar{m})$, alors la valeur obtenue vérifie :

$$\delta^m(s_2, s_3) \leq \delta^*(s_2, s_3, \bar{m}) \leq \delta^m(s_1, s_2). \tag{C.2}$$

Pour cela, on part de m et on lui ajoute un couple (x, e_x), où e_x est un élément dans s_2. En d'autres termes, $\bar{m} = m \cup \{(x, x_e)\}$. Ici le choix de e_x est tel que e_x n'appartient pas à m, c'est-à-dire il n'est pas apparié lors du calcul de $\delta^m(s_1, s_2)$. L'appariement $\bar{m}$ est manifestement un appariement exclusif entre s_2 et s_3. On peut affirmer donc que son poids est supérieur à celui de m' car ce dernier est optimal :

$$\sum_{m'}(s_2, s_3) \leq \sum_{\bar{m}}(s_2, s_3).$$

En conséquence, les dissimilarités prises selon chacun des deux appariements ont le même rapport :

$$\delta^m(s_2, s_3) \leq \delta^*(s_2, s_3, \bar{m}). \tag{C.3}$$

Cela constitue la moitié du résultat recherché. Il reste donc à prouver la partie droite de l'inégalité C.2, soit :

$$\delta^*(s_2, s_3, \bar{m}) \leq \delta^m(s_1, s_2).$$

Pour cela on utilise la décomposition de $\bar{m}$ en termes de m, soit $\bar{m} = m \cup \{(x, x_e)\}$, et de s_3 en s_1 et x. Tout d'abord, on développe la dissimilarité :

$$\delta^*(s_2, s_3, \bar{m}) = \frac{\sum_{\bar{m}}(s_2, s_3) + q - p}{q}. \tag{C.4}$$

Ensuite, on décompose le poids de $\bar{m}$ en poids sur m et la dissimilarité sur le couple supplémentaire, (x, x_e). Lorsqu'on considère le poids de m, il faut rectifier les arguments de la fonction de poids, c'est-à-dire mettre s_1 à la place de s_3 :

$$\sum_{\bar{m}}(s_2, s_3) = \sum_{m}(s_1, s_2) + \delta_D(x, x_e). \tag{C.5}$$

Par ailleurs, la normalisation de la mesure δ_D nous permet d'affirmer que $\delta_D(x, x_e) \leq 1$. En utilisant ce fait, puis en combinant les égalités C.4 et C.5, on obtient l'inégalité suivante :

$$\delta^*(s_2, s_3, \bar{m}) \leq \frac{\sum_m(s_1, s_2) + 1 + q - p}{q}. \tag{C.6}$$

La partie droite est exactement la valeur de $\delta^m(s_1, s_2)$:

$$\delta^m(s_1, s_2) = \frac{\sum_m(s_1, s_2) + q - (p-1)}{q}.$$

Si les inégalités C.3 et C.6 sont prises en compte elles composent le résultat désiré :

$$\delta^m(s_2, s_3) \leq \delta^*(s_2, s_3, \bar{m}) \leq \delta^m(s_1, s_2).$$

Ceci termine la preuve de la monotonie. □

Proposition 5.3. *Soit un domaine D muni d'une mesure de dissimilarité δ_D. Si δ_D est une métrique, alors la fonction sur les ensembles d'éléments dans D, δ^m, est également une métrique.*

Esquisse de preuve. Soient trois ensembles A, B et C sur un type de base à domaine D tels que :

1. $A = \{a_1, ..., a_{l_a}\}$, $\|A\| = l_a$
2. $B = \{b_1, ..., b_{l_b}\}$, $\|B\| = l_b$
3. $C = \{c_1, ..., c_{l_c}\}$, $\|C\| = l_c$
4. et $l_a \leq l_b \leq l_c$

Notre but sera de prouver les trois inégalités suivantes :

$$\left\| \begin{array}{lcl} \delta^m(A,B) & \leq & \delta^m(A,C) + \delta^m(B,C) \\ \delta^m(A,C) & \leq & \delta^m(A,B) + \delta^m(B,C) \\ \delta^m(B,C) & \leq & \delta^m(A,C) + \delta^m(A,B) \end{array} \right.$$

Les trois cas ne sont pas symétriques car la différence de cardinalité est exploitée.
Soient encore les appariements optimaux suivants :

$$\begin{array}{lcl} m_1 & = & M_{opt}(A,B) \\ m_2 & = & M_{opt}(B,C) \\ m_3 & = & M_{opt}(A,C) \end{array}$$

Précisons que les appariements exclusifs que nous utilisons ici définissent des applications injectives entre les ensembles. Nous pouvons donc leur donner un sens et les considérer comme des fonctions totales de l'ensemble de plus petite cardinalité vers l'ensemble de plus grande cardinalité :

$$\begin{array}{lcl} m_1 & : & A \rightarrow B \\ m_2 & : & B \rightarrow C \\ m_3 & : & A \rightarrow C \end{array}$$

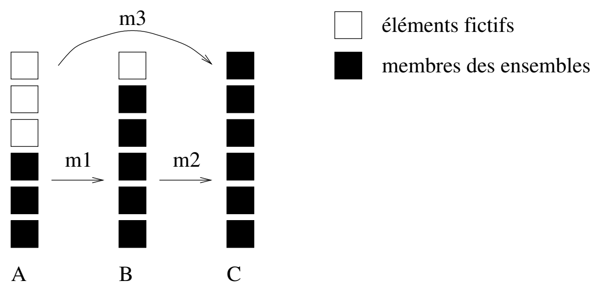


FIG. C.1: *Un élément A_i de A et ces deux images dans C, $m_3(a_i)$ et $m_2 \circ m_1(a_j)$.*

L'idée de la preuve est, en se basant sur le caractère optimal de chaque appariement ci-dessus et sur la propriété de métricité de δ_D, de construire dans chacun des cas un appariement $\bar{m}_i$, $i = 1, 2, 3$. L'appariement $\bar{m}_i$ qui plonge s_i dans s_j est tel que la dissimilarité δ^* sur cet appariement «s'insère» entre la dissimilarité $\delta^m(s_i, s_j)$ et la somme des deux autres dissimilarités. Pour l'obtenir, on utilise la composition de fonctions, l'opérateur $\circ$.

Cas un. On cherche à prouver :

$$\delta^m(A, C) \leq \delta^m(A, B) + \delta^m(B, C). \tag{C.7}$$

L'appariement $\bar{m}_3$ est facilement obtenu en composant m_1 et m_2 :

$$\bar{m}_3 = m_2 \circ m_1$$

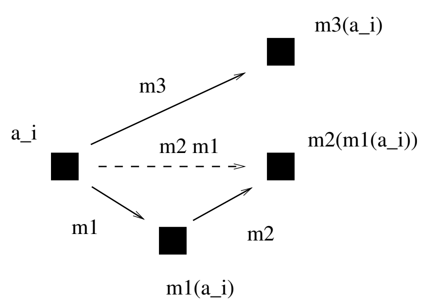


FIG. C.2: *Trois ensembles avec les trois fonctions qui correspondent aux appariements entre ensembles. Les éléments fictifs sont visualisés également.*

Cette définition est correcte car $\bar{m}_3$ est total sur A et injectif sur C (produit de fonctions totales et injectives). On a donc droit de considérer, pour chaque j de 1 à l_a, les trois valeurs :

$$\delta_D(a_j, m_1(a_j))$$
$$\delta_D(a_j, \bar{m}_3(a_j))$$
$$\delta_D(m_1(a_j), m_2 \circ m_1(a_j))$$

en sachant que $m_2 \circ m_1(a_j) = \bar{m}_3(a_j)$. Ces trois valeurs vérifient :

$$\delta_D(a_j, \bar{m}_3(a_j)) \leq \delta_D(a_j, m_1(a_j)) + \delta_D(m_1(a_j), \bar{m}_3(a_j)).$$

car δ_D est une métrique. Si l'on fait la somme sur A, alors on obtient :

$$\sum_{\bar{m}_3}(A,C) \leq \sum_{m_1}(A,B) + \sum_{m_{2|m_1(A)}}(B,C) \tag{C.8}$$

où $m_{2|m_1(A)}$ est la restriction de m_2 sur le co-domaine de m_1.
On note que la dernière somme est une sous-somme du poids total de m_2 :

$$\sum_{m_{2|m_1(A)}}(B,C) = \sum_{m_2}(B,C) - \sum_{m_{2|(B-m_1(A))}}(B,C) \tag{C.9}$$

Notons par Res la somme restante :

$$Res = \sum_{m_{2|(B-m_1(A))}}(B,C)$$

Manifestement, $Res \geq 0$. On développe maintenant les deux dissimilarités à droite de l'inégalité C.7 :

$$\delta^m(A,B) + \delta^m(B,C) = \frac{\sum_{m_1}(A,B) + l_b - l_a}{l_b} + \frac{\sum_{m_2}(B,C) + l_c - l_b}{l_c}. \tag{C.10}$$

D'après les hypothèses, $l_b \leq l_c$. On peut donc diminuer la partie droite de l'égalité ci-dessus en augmentant le dénominateur dans le premier terme. En effectuant la somme :

$$\delta^m(A,B) + \delta^m(B,C) \leq \frac{\sum_{m_1}(A,B) + l_b - l_a}{l_c} + \frac{\sum_{m_2}(B,C) + l_c - l_b}{l_c}. \tag{C.11}$$

En réunissant les deux fractions dans la partie droite de l'inégalité et en simplifiant les différences des cardinalités, on arrive à :

$$\delta^m(A,B) + \delta^m(B,C) \geq \frac{\sum_{m_1}(A,B) + \sum_{m_2}(B,C) + l_c - l_a}{l_c}. \tag{C.12}$$

En substituant au second terme du numérateur à droite ses deux parties vues dans C.9, on obtient :

$$\delta^m(A,B) + \delta^m(B,C) \geq \frac{\sum_{m_1}(A,B) + \sum_{m_{2|\alpha}} + Res + l_c - l_a}{l_c} \tag{C.13}$$

où l'ensemble α remplace la différence $(B - m_1(A))$. En appliquant l'inégalité C.8 qui exprime la propriété de métricité de δ_D, on obtient :

$$\delta^m(A,B) + \delta^m(B,C) \geq \frac{\sum_{\bar{m}_3}(A,C) + l_c - l_a}{l_c}. \tag{C.14}$$

Il reste à remarquer que la dernière expression est exactement la dissimilarité $\delta^*(A,C,\bar{m}_3)$.

$$\delta^m(A,B) + \delta^m(B,C) \geq \delta^*(A,C,\bar{m}_3) \tag{C.15}$$

La dissimilarité à droite est supérieure à $\delta^m(A,B)$ car le dernier est optimal (donc de dissimilarité minimale). En conséquence, on retrouve le résultat recherché :

$$\delta^m(A,B) + \delta^m(B,C) \geq \delta^m(A,C).$$

A partir de là, on obtient facilement l'inégalité désirée. Les deux autres cas peuvent être traités de manière semblable, en se basant sur les mêmes propriétés.

Cas deux. On cherche à prouver :

$$\delta^m(B,C) \leq \delta^m(A,B) + \delta^m(A,C). \tag{C.16}$$

L'appariement $\bar{m}_2$ est obtenu en composant m_1^{-1} et m_3, puis en complétant l'appariement partiel obtenu jusqu'à ce qu'un appariement total et exclusif soit obtenu :

$$\bar{m}_2 = m_3 \circ m_1^{-1} \cup \bar{m}_r = \bar{m}_{1,3} \cup \bar{m}_r \tag{C.17}$$

Ici, $\bar{m}_{1,3}$ est le produit des deux appariements m_3 et m_1^{-1}, alors que $\bar{m}_r$ est un appariement exclusif et total quelconque entre $B - m_1(A)$ et $C - m_3(A)$. Notons que l'appariement $\bar{m}_2$ est correctement défini car il est : (i) total sur B, et (ii) exclusif sur C.

Notre but est de prouver que la dissimilarité entre B $\delta^*(B,C,\bar{m}_2)$ est inférieure à la somme à droite de l'inégalité C.16. Pour cela, d'abord nous utilisons l'inégalité triangulaire :

$$\sum_{\bar{m}_{1,3}}(B,C) \leq \sum_{m_1}(A,B) + \sum_{m_3}(A,C). \tag{C.18}$$

A cet instant, nous utilisons le fait que la fonction δ_D est *normalisée* et que par conséquent toutes les valeurs de δ_D sont inférieures à 1. Ainsi, la somme des dissimilarités sur les couples dans $\bar{m}_{1,3}$ ne dépasse pas leur nombre qui est $l_b - l_a$:

$$\sum_{\bar{m}_r}(B,C) \leq l_b - l_a. \tag{C.19}$$

Il reste à développer les dissimilarités dont on veut évaluer la somme. Comme dans le cas précédent, on simplifie pour arriver à un dénominateur commun :

$$\delta^m(A,B) + \delta^m(A,C) \geq \frac{\sum_{m_1}(A,B) + \sum_{m_3}(A,C) + l_c + l_b - 2 * l_a}{l_c} \tag{C.20}$$

Il ne reste qu'à combiner les trois dernières inégalités en un seul résultat :

$$\delta^m(A,B) + \delta^m(A,C) \geq \frac{\sum_{\bar{m}_{1,3}}(B,C) + \sum_{\bar{m}_r}(B,C) + l_c - l_a}{l_c} \tag{C.21}$$

Compte tenu de l'égalité C.17, la partie droite est exactement $\delta^*(B,C,\bar{m}_2)$:

$$\delta^m(A,B) + \delta^m(A,C) \geq \delta^*(B,C,\bar{m}_2) \geq \delta^m(B,C) \tag{C.22}$$

Ainsi, nous avons terminé la preuve du second cas de la proposition.

Cas trois. Le but est de démontrer :

$$\delta^m(A,B) \leq \delta^m(A,C) + \delta^m(B,C). \tag{C.23}$$

De nouveau, un appariement est construit entre A et B, $\bar{m}_1$, dont on veut démontrer que :

$$\delta^*(A, B, \bar{m}_1) \leq \delta^m(A, C) + \delta^m(B, C).$$

L'appariement $\bar{m}_1$ est le produit des deux autres, complété par un ensemble de couples :

$$\bar{m}_1 = m_2^{-1} \circ m_3 \cup \bar{m}_r$$

C'est un appariement entre A et B, total et exclusif. Il peut donc être utilisé à titre de comparaison dans la preuve.

En appliquant l'inégalité triangulaire sur les valeurs respectives, on obtient l'inégalité suivante :

$$\sum_{\bar{m}_1}(A, B) \leq \sum_{m_{2|m_1(A)}}(B, C) + \sum_{m_3}(A, C) \tag{C.24}$$

En suivant un raisonnement similaire à celui employé dans le **cas deux** de la proposition, nous arrivons à la comparaison des deux sommes ci-dessous :

$$\delta^*(A, B, \bar{m}_1) = \frac{\sum_{\bar{m}_1}(A, B) + l_b - l_a}{l_b} \tag{C.25}$$

et

$$\delta^m(A, C) + \delta^m(B, C) = \frac{\sum_{m_{2|m_1(A)}}(B, C) + \sum_{m_3}(A, C) + l_b - l_a + l_c - l_a}{l_c}. \tag{C.26}$$

Soient α et β les deux sommes dans le numérateur de la fraction ci-dessus. En d'autres termes,

$$\beta = \sum_{m_{2|m_1(A)}}(B, C),$$

et

$$\alpha = \sum_{m_3}(A, C).$$

Le but de la preuve peut être reformulé en utilisant les équations C.24, C.19 et C.20 en des termes utilisant ces deux quantités. En effet, pour achever la preuve, il suffit de démontrer que :

$$\frac{\sum_{\bar{m}_1}(A, B) + l_b - l_a}{l_b} \leq \frac{\alpha + \beta + 2 * l_b - l_a + l_c - l_a}{l_c}. \tag{C.27}$$

En continuant les développements, en particulier, l'égalité C.24, on arrive à l'inégalité suivante :

$$\frac{\alpha + \beta + l_b - l_a}{l_b} \leq \frac{\alpha + \beta + 2l_b - l_a + l_c - l_a}{l_c}. \tag{C.28}$$

Dans la suite, nous ne donnons que les grandes étapes des calculs à effectuer dans la preuve de cette inégalité. Tout d'abord, une multiplication nous permet de nous libérer des dénominateurs des fractions.

$$(\alpha + \beta + l_b - l_a) * l_c) \leq (\alpha + \beta + 2 * l_b - l_a + l_c - l_a) * l_b.$$

Les pas suivants sont une simple poursuite des transformations sur les deux expressions ci-dessus. Elles visent à ramener les calculs à une comparaison avec zéro.

$$(\alpha + \beta) * (l_c - l_b) \leq (2 * l_c - l_b - l_a) * l_b - (l_b - l_a) * l_c$$

$$(2 * l_c - l_b - l_a) * l_b - (l_b - l_a) * l_c = l_b * l_c - l_b^2 - l_a * (l_c - l_b).$$

Finalement, la dernière expression se simplifie en :

$$(l_b + l_a) * (l - c - l_b)$$

Nous pouvons remarquer que α et β sont des sommes de nombres dans [0 1]. Il s'en suit que chacun d'eux ne dépasse pas la cardinalité de la somme sous-jacente. Ainsi, α vérifie $\alpha \leq l_a$. A son tour β est au plus égal à l_b. En conséquence, la somme de ce deux nombres ne dépasse pas la somme de l_b et l_a. Ce fait nous permet de valider l'inégalité 2.1.3 ci-dessous et par là, le principe de la métricité.

Ceci achève la preuve du cas trois et de la proposition entière. □

C.3 Preuves des propriétés du chapitre 7

Dans la suite, nous allons utiliser les notations suivantes :

- x_i pour les inconnues du système
- x_i^r pour les valeurs concrètes, le résultat du r-ème pas du calcul itératif

Le calcul itératif de la solution du système 7.7 repose sur la donnée d'une fonction vectorielle $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$:

$$\mathbf{F} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$$

La fonction opère une transformation sur les vecteurs $\mathbf{X}$ dans $\mathbb{R}^n$:

$$\mathbf{X} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Chaque fonction f_i est une fonction réelle à n inconnues, $f_i : \mathbb{R}^n \to \mathbb{R}$. L'application de $\mathbf{F}$ sur un vecteur $\mathbf{X}$, notée $\mathbf{F(X)}$, est un autre vecteur $\mathbf{Y} \in \mathbb{R}^n$:

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

où pour tout i, $y_i = f_i(x_1, x_2, \ldots, x_n)$ noté également $f_i(\mathbf{X})$. Chaque fonction f_i est une combinaison linéaire de ses arguments :

$$f_i(\mathbf{X}) = a_{i,0} + \sum_{j=1, j \neq i}^{n} a_{i,j} * x_j \tag{C.29}$$

L'expression de chaque fonction f_i est tirée de la matrice $\mathbf{C}$ en se servant de l'équation 7.3. L'idée est d'isoler x_i d'un coté avec un coefficient 1 pour le définir en fonction des autres variables.

Deux cas sont possibles pour un i fixé. Soit d'abord $c_{i,i}$ égal à 1. Dans ce cas, les valeurs des $a_{i,j}$ sont affectées directement. La valeur du membre de degré zéro $a_{i,0}$ est prélevée sur b_i, alors que pour chaque j de 1 à n, sauf $j = i$, la valeur de $a_{i,j}$ est l'inverse de $c_{i,j}$, $a_{i,j} = -c_{i,j}$. A noter que la valeur de $a_{i,i}$ est fixée à zéro. Soit maintenant $c_{i,i}$ inférieur à 1. Dans ce second cas, une division de $c_{i,j}$ par $c_{i,i}$ est nécessaire afin d'arriver à un coefficient 1 devant x_i dans l'équation 7.3. Ainsi,

$$a_{i,0} = \frac{b_i}{c_{i,i}},\ a_{i,j} = -\frac{c_{i,j}}{c_{i,i}} \tag{C.30}$$

Ayant défini les fonctions composantes de $\mathbf{F}$ nous pouvons étudier le comportement de cette dernière fonction. La solution $\overline{\mathbf{X}}$ du système 7.7 est le plus petit point fixe de celle-ci et il peut être obtenu par une suite d'applications successives de $\mathbf{F}$ sur $\mathbf{0}$, l'origine dans $\mathbb{R}^n$ [Golub et van Loan, 1983].

Proposition 7.1. *Il existe un homomorphisme h entre le réseau d'un objet, $R(o)$ et le graphe de dépendances entre concepts $\mathcal{K}$ tel que :*

- $\forall \bar{o} \in R(o), h(\bar{o}) = concept(\bar{o})$,
- *h préserve les étiquettes sur les arcs, c'est-à-dire les noms des attributs.*

Esquisse de preuve. En effet, cet homomorphisme renvoi chaque objet du réseau vers son concept respectif. Il existe un arc entre deux sommets du réseau si et seulement si les objets respectifs sont liés par un attribut dont le nom correspond à l'étiquette de l'arc. L'existence d'un tel attribut entre objets implique l'existence de l'attribut du même nom entre concepts et donc l'existence de l'arc correspondant dans le graphe des dépendances. □

Proposition 7.2. *Si pour tout i, $A_i - Circ(o_i, o'_i) \neq \varnothing$, alors l'inégalité suivante est vérifiée pour tout i aussi :*

$$\forall i,\ \sum_{j=1, j \neq i}^{m} |c_{i,j}| < |c_{i,i}| \tag{C.31}$$

Preuve. Rappelons que les valeurs des coefficients $c_{i,j}$ sont soit zéro, soit des sommes de poids d'attributs λ_t. Le second cas se produit seulement si les couples d'objets dont les dissimilarités sont dénotées par x_i et x_j sont liés par un arc dans la CFC. Plus exactement, soient ces couples (o_i, o'_i) et (o_j, o'_j) respectivement, il doit exister au moins un attribut a_t dans le concept $k_i = concept(o_i)$ avec $o_i.a_t = o_j$ et $o'_i.a_t = o'_j$.

Les coefficients non nuls sur la ligne i hors la diagonale sont donc tous des poids des attributs dans l'expression déterminant x_i :

$$d^o_{C_i}(o_i, o'_i) = \sum_{l=1}^{|A_i|} \lambda * \delta^f_a(o_i.a, o'_i.a) \tag{C.32}$$

D'après la définition de l'ensemble $Circ(o_i, o'_i)$, celui-ci inclut tous les attributs a_t correspondant à des coefficients non nuls. L'inclusion est stricte si l'ensemble des attributs qui bouclent sur les objets o_i, o'_i, $Boucle(o_i, o'_i)$, est non-vide. Sinon, l'ensemble $Circ(o_i, o'_i)$ contient exactement les attributs qui nous intéressent. En d'autres termes, $Circ(o_i, o'_i)$ contient en plus de ces attributs, les attributs qui bouclent sur les objets et donc font dépendre la valeur de $d^o_{k_i}(o_i, o'_i)$ d'elle-même.

D'après la définition de $d^o_{k_i}$, la somme de tous les poids λ dans l'expression C.32 est 1 :

$$\sum_{l=1}^{|A_i|} \lambda = 1 \tag{C.33}$$

D'après notre hypothèse de départ, la somme des poids des attributs dans $Circ(o_i, o'_i)$ est strictement inférieure à 1 :

$$\sum_{a \in Circ(o_i, o'_i)} \lambda = 1 - \sum_{a \in A_i - Circ(o_i, o'_i)} \lambda < 1 \tag{C.34}$$

Pour obtenir la somme des coefficients qui nous intéresse, il suffira de sommer les poids des attributs dans $Circ(o_i, o'_i)$ dont on a soustrait les boucles. En effet, les valeurs absolues sont égales aux poids qui sont toujours non-négatifs. On obtient donc :

$$\sum_{j=1, j \neq i}^{m} |c_{i,j}| = \sum_{a \in Circ(o_i, o'_i) - Boucle(o_i, o'_i)} \lambda_a \tag{C.35}$$

En enlevant la somme des poids des boucles des deux cotés de l'inégalité C.34, on obtient :

$$\sum_{a \in Circ(o_i, o'_i)} \lambda_a - \sum_{a \in Boucle(o_i, o'_i)} \lambda_a < 1 - \sum_{a \in Boucle(o_i, o'_i)} \lambda_a \tag{C.36}$$

Il suffit maintenant de remarquer que la partie droite de l'inégalité ci-dessus est exactement la valeur de $c_{i,i}$. En effet, rappelons que cette valeur s'obtient à partir de l'expression 7.3 par une soustraction des poids des attributs qui bouclent simultanément sur les objets d'origine. □

Proposition 7.3. *Toute fonction d^o calculée à l'aide du système d'équations ci-dessus n'admet que des valeurs non-négatives :*

$$d^o(o, o') \geq 0.$$

Preuve. Notons $\{\mathbf{X}^r\}_{r=0,\infty}$ la suite des vecteurs obtenus par applications de $\mathbf{F}$ sur $\mathbf{0}$. Ainsi, $\mathbf{X}^0 = \mathbf{0}$ et pour tout r, $\mathbf{X}^{r+1} = \mathbf{F}(\mathbf{X}^r)$. Soit encore,

$$\mathbf{X}^r = \begin{pmatrix} x_1^r \\ x_2^r \\ \vdots \\ x_n^r \end{pmatrix}$$

Nous allons montrer que les suites sur les dimensions x_i^r du vecteur sont non-décroissantes, c'est-à-dire que pour tout i, la suite $\{x_i^r\}_{r=0,\infty}$ est non-décroissante.

La preuve se fait par une induction sur k : il suffit de montrer que pour tout l la sous-suite $\{x_i^k\}_{k=0,l}$ est non-décroissante.

Notons d'abord que pour tout i, $x_i^0 = 0$ et $x_i^1 = a_{i,0}$. Tous les membres $a_{i,0}$ sont non-négatifs car combinaison linéaire à coefficients positifs de dissimilarités. On en déduit que $x_i^0 \leq x_i^1$ pour tout i. Notre hypothèse est donc vérifiée pour $l = 1$.

Supposons que pour $l = p > 0$ toutes les sous-suites $\{x_i^k\}_{k=0,l}$ sont non-décroissantes. Prouvons cette affirmation pour $l = p + 1$. Pour cela il suffit de montrer que pour un i arbitraire, $x_i^p \leq x_i^{p+1}$. En exprimant les deux valeurs par la fonction f_i, on obtient :

$$x_i^p = a_{i,0} + \sum_{j=1, j\neq i}^{n} a_{i,j} * x_j^{p-1} \tag{C.37}$$

$$x_i^{p+1} = a_{i,0} + \sum_{j=1, j\neq i}^{n} a_{i,j} * x_j^{p} \tag{C.38}$$

En appliquant l'hypothèse d'induction ($x_i^{p-1} \leq x_i^p$) sur les parties droites des expressions, on obtient

$$\sum_{j=1, j\neq i}^{n} a_{i,j} * x_j^{p-1} \leq \sum_{j=1, j\neq i}^{n} a_{i,j} * x_j^{p} \tag{C.39}$$

et donc $x_i^p \leq x_i^{p+1}$. Par conséquent, la sous-suite $\{x_i^k\}_{k=0,l}$ pour $l = p + 1$ est non-décroissante. Ceci est valable pour tout i. Par induction, quel que soit l, les sous-suites $\{x_i^k\}_{k=0,l}$ sont non-décroissantes ce qui signifie que les suites entières le sont aussi. On en déduit que la solution du système, soit $\mathbf{X}^\infty$, est composée par des valeurs non négatives. □

Proposition 7.4. *Toute fonction d^o calculée à l'aide du système d'équations ci-dessus n'admet que des valeurs qui ne dépassent pas* 1 *:*

$$d^o(o, o') \leq 1$$

Preuve. Il suffit de démontrer que pour tout i, la suite $\{x_i^k\}_{k=0,\infty}$ est bornée par 1. Comme dans le cas précédent nous allons procéder par induction sur la longueur l d'une sous-suite vérifiant la propriété recherchée.

Soit $l = 1$ et considérons, pour un i arbitraire, $\{x_i^1\}$. Par définition, $\{x_i^1\} = a_{i,0}$ et deux cas peuvent se présenter par rapport à la manière dont $a_{i,0}$ est obtenue depuis les expressions initiales des dissimilarités.

Cas **un**, $a_{i,0} = b_i$. Par définition, b_i est une combinaison linéaire de dissimilarités :

$$b_i = \sum_{a \in A_i - Circ(o,o')} \lambda_a * \delta_a^f(o.a, o'.a) \tag{C.40}$$

où (o, o') est le couple d'objets qui correspond à la variable x_i. Par hypothèse, les valeurs des dissimilarités δ_a^f sont toutes normalisées et donc inférieures ou égales à 1. On en déduit :

$$\sum_{a \in A_i - Circ(o,o')} \lambda_a * \delta_a^f(o.a, o'.a) \leq \sum_{a \in A_i - Circ(o,o')} \lambda_a \tag{C.41}$$

La somme en partie droite de l'inégalité ci-dessus est strictement inférieure à 1, car d'une part, la somme totale des poids sur A_i est 1 et d'autre part, l'ensemble $Circ(o, o')$ contient au moins un élément. En effet, dans le cas où $Circ(o,o')$ serait vide, le couple (o,o') ne pourrait pas appartenir à la CFC considérée. Formellement,

$$\sum_{a\in A_i - Circ(o,o')} \lambda_a \;<\; \sum_{a\in A_i} \lambda_a \;=\; 1 \tag{C.42}$$

Cas **deux**, $a_{i,0} = b_i * c_{i,i}^{-1}$. Rappelons la définition de $c_{i,i}$:

$$c_{i,i} = 1 - \sum_{a\in Boucle(o,o')} \lambda \tag{C.43}$$

où (o,o') est le couple d'objets qui correspond à la variable x_i.

$$a_{i,0} = \frac{\sum_{a\in A_i - Circ(o,o')} \lambda_a * \delta_a^f(o.a, o'.a)}{1 - \sum_{a\in Boucle(o,o')} \lambda_a} \tag{C.44}$$

Avec les mêmes contraintes sur les dissimilarités que dans le premier cas, nous obtenons :

$$a_{i,0} \leq \frac{\sum_{a\in A_i - Circ(o,o')} \lambda_a}{1 - \sum_{a\in Boucle(o,o')} \lambda_a} \tag{C.45}$$

On peut donc séparer la somme sur $Circ(o,o')$ en deux sous-sommes :

$$\sum_{a\in A_i - Circ(o,o')} \lambda_a \;=\; 1 - \sum_{a\in Circ(o,o')} \lambda_a \;=\; 1 - (\sum_{a\in Circ(o,o') - Boucle(o,o')} \lambda_a + \sum_{a\in Boucle(o,o')} \lambda_a) \tag{C.46}$$

D'après la définition des ensembles $Circ(o,o')$ et $Boucle(o,o')$ le second est strictement inclus dans le premier :

$$Circ(o,o') - Boucle(o,o') \neq \emptyset$$

On est donc sûr que la somme des poids des attributs dans cet ensemble est positive.

$$1 - (\sum_{a\in Circ(o,o') - Boucle(o,o')} \lambda_a + \sum_{a\in Boucle(o,o')} \lambda_a) \;<\; 1 - \sum_{a\in Boucle(o,o')} \lambda_a \tag{C.47}$$

On en déduit que le numérateur de la fraction dans la partie droite de C.45 ci-dessus est strictement inférieur à son dénominateur :

$$\sum_{a\in A_i - Circ(o,o')} \lambda_a \;<\; 1 - \sum_{a\in Boucle(o,o')} \lambda_a \tag{C.48}$$

Avec une division des inégalités par $1 - \sum_{a\in Boucle(o,o')} \lambda_a$, on obtient :

$$\frac{\sum_{a\in A_i - Circ(o,o')} \lambda_a}{1 - \sum_{a\in Boucle(o,o')} \lambda_a} < 1 \tag{C.49}$$

On a réussi donc à démontrer dans ce second cas que $a_{i,0} < 1$. Il s'en suit que tous les éléments de rang un des séquences,$\{x_i^1\}$ sont strictement inférieurs à un.

Supposons que pour $l = p > 0$ tous les membres des suites $\{x_i^k\}_{k=0,l}$ sont strictement inférieurs à un. Nous allons prouver la propriété pour $l = p + 1$. Prenons un i quelconque et considérons l'expression de x_i^{p+1} pour la démonstration. D'après la définition de $\mathbf{F}$ et $\{x_i^k\}_{k=0,\infty}$:

$$x_i^{p+1} = a_{i,0} + \sum_{j=1,j\neq j}^{n} a_{i,j} * x_j^p \tag{C.50}$$

Selon l'hypothèse d'induction tous les x_j^p sont inférieurs à un. On en déduit :

$$x_i^{p+1} \; < \; a_{i,0} + \sum_{j=1,j\neq j}^{n} a_{i,j} \tag{C.51}$$

Pour évaluer la partie droite de l'inégalité, considérons à nouveau deux cas.

Cas **un**, $a_{i,0} = b_i$ et $a_{i,j} = -c_{i,j}$. Comme dans le pas de base, utilisons la définition de b_i :

$$a_{i,0} = \sum_{a\in A_i - Circ(o,o')} \lambda_a * \delta_a^f(o.a, o'.a) \;\leq\; \sum_{a\in A_i - Circ(o,o')} \lambda_a \tag{C.52}$$

où (o, o') est le couple d'objets qui correspond à la variable x_i. Pour la somme des autres coefficients $a_{i,j}$ on peut remarquer que celle-ci est égale à la somme des poids des attributs a dans $Circ(o, o')$:

$$\sum_{j=1,j\neq j}^{n} a_{i,j} \;=\; \sum_{a\in Circ(o,o')} \lambda_a \tag{C.53}$$

En rassemblant les deux résultats, on obtient :

$$a_{i,0} + \sum_{j=1,j\neq j}^{n} a_{i,j} \;\leq\; \sum_{a\in A_i - Circ(o,o')} + \sum_{a\in Circ(o,o')} \lambda_a \;=\; 1 \tag{C.54}$$

Par conséquent, la valeur de x_i^{p+1} est strictement inférieure à un dans ce premier cas où il n'y a pas de boucles.

Dans le cas **deux**, les coefficients sont obtenus par une division par le complément du poids total des attributs boucles $a_{i,0} = b_i * c_{i,i}^{-1}$ et $a_{i,j} = -c_{i,j} * c_{i,i}^{-1}$. Nous pouvons appliquer le même raisonnement que pour évaluer $a_{i,0}$:

$$a_{i,0} = \;\leq\; \frac{\sum_{a\in A_i - Circ(o,o')} \lambda_a}{1 - \sum_{a\in Boucle(o,o')} \lambda_a} \tag{C.55}$$

De manière identique, la somme des autres coefficients $a_{i,j}$ est inférieure ou égale à la somme des poids des attributs dans $Circ(o, o') - Boucle(o, o')$ divisé par $c_{i,i}$:

$$\sum_{j=1,j\neq j}^{n} a_{i,j} \;\leq\; \frac{\sum_{a\in Circ(o,o') - Boucle(o,o')} \lambda_a}{1 - \sum_{a\in Boucle(o,o')} \lambda_a} \tag{C.56}$$

En mettant tout ensemble :

$$a_{i,0} + \sum_{j=1,j\neq j}^{n} a_{i,j} \;\leq\; \frac{\sum_{a\in A_i - Circ(o,o')} \lambda_a + \sum_{a\in Circ(o,o') - Boucle(o,o')} \lambda_a}{1 - \sum_{a\in Boucle(o,o')} \lambda_a} \tag{C.57}$$

Avec une transformation simple :

$$a_{i,0} + \sum_{j=1, j \neq j}^{n} a_{i,j} \;\leq\; \frac{\sum_{a \in A_i - Boucle(o,o')} \lambda_a}{1 - \sum_{a \in Boucle(o,o')} \lambda_a} \;=\; 1 \tag{C.58}$$

Par conséquent, nous avons réussi a démontrer que

$$x_i^{p+1} \;<\; a_{i,0} + \sum_{j=1, j \neq j}^{n} a_{i,j} \;\leq\; 1 \tag{C.59}$$

Cela termine la preuve de la propriété pour $l = p + 1$. Ainsi, nous avons prouvé que pour un l quelconque et pour tout i, $x_i^l < 1$. Les solutions du système sont donc toutes inférieures ou égales à un. □

Proposition 7.5. *Pour tout* r*, si* $\forall i, i = 1, m$ $x_t^r \leq x_t^{r+1}$ *alors*

$$\sum_{M_{opt}^r} x_t^r \leq \sum_{M_{opt}^{r+1}} x_t^{r+1}$$

Preuve. La preuve sera établie par l'absurde, supposons donc que

$$\sum_{M_{opt}^{r+1}} x_t^{r+1} \;<\; \sum_{M_{opt}^r} x_t^r$$

L'idée sera de mettre en défaut la condition de minimalité sur M_{opt}^r, c'est-à-dire de trouver un appariement M^r à l'étape r tel que

$$\sum_{M^r} x_t^r \;<\; \sum_{M_{opt}^r} x_t^r$$

Pour cela, il suffit de prendre l'appariement M_{opt}^{r+1}. Rappelons que les deux appariements, M_{opt}^r et M_{opt}^{r+1} sont de même cardinalité et considérons la somme

$$\sum_{M_{opt}^{r+1}} x_t^r.$$

Nous pouvons remarquer que l'inégalité suivante est vérifiée

$$\sum_{M_{opt}^{r+1}} x_t^r \;\leq\; \sum_{M_{opt}^{r+1}} x_t^{r+1}.$$

En effet, la somme sur M_{opt}^{r+1} à l'étape r est nécessairement au plus égale à la somme sur le même appariement à l'étape suivante si l'on admet que toutes les suites sont non-décroissantes au moins jusqu'à l'indice $r + 1$.

Finalement, en composant les deux inégalités ci-dessus, la contradiction recherchée est obtenue :

$$\sum_{M_{opt}^{r+1}} x_t^r \;\leq\; \sum_{M_{opt}^r} x_t^r$$

En d'autres termes, nous avons exhibé un exemple d'appariement qui est strictement mieux que l'appariement optimal ce qui est impossible.

En somme, la mannière de calculer l'appariemenet entre objets assure la non-décroissance de la suite des valeurs de la dissimilarité globale sur les appariements éffectués à chaque pas du calcul itératif. □

Proposition 7.6. *Pour tout k la fonction d_k^o calculée à l'aide d'un système d'équations du type ci-dessus est un indice de dissimilarité valide.*

Preuve. Toute valeur de dissimilarité calculée à l'aide du système est non négative. En cela les fonctions d_k^o vérifient la propriété de positivité (voir section 2.3 de chapitre 2). Ensuite, pour un couple d'objets o, o' du concept k, les valeurs de $d_k^o(o, o')$ et de $d_k^o(o', o)$ sont égales. En effet, tous les calculs sont parfaitement symétriques, l'ordre entre les arguments des fonctions n'entre nulle part en considération. Finalement, la valeur de $d_k^o(o, o)$ est zéro. Une démonstration rigoureuse de cette propriété demandera plus de place. Mais intuitivement, on peut facilement s'apercevoir que lorsque les arguments sont identiques, tous les couples dans $Couples(o, o)$ ont la forme (o', o'). Ainsi, tous les écarts de (o', o') calculés sur des attributs qui ne sont pas dans $Circ(o', o')$ sont de zéro. Par conséquent, leur somme est aussi zéro. Comme cette somme est le coefficient libre dans les équations du système. Le système est donc homogène et sa seule solution est le vecteur $\mathbf{0}$. □

Bibliographie

[Abadi et Cardelli, 1996] M. Abadi et L. Cardelli. *A Theory Of Objects*. Monographs in Computer Science. Springer-Verlag, Berlin, 1996.

[Abiteboul *et al.*, 1995] S. Abiteboul, R. Hull et V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.

[Abiteboul, 1997] S. Abiteboul. Querying semi-structured data. Dans *Proceedings of the ICDT'97 Conference*, rédacteurs F. N. Afrati et P. G. Kolaitis, volume 1186 de *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, 1997.

[Aho *et al.*, 1972] A. V. Aho, M. R. Garey et J. D. Ullman. The transitive reduction of a directed graph. *SIAM Journal of Computing*, 1 :131–137, 1972.

[Ahuja *et al.*, 1993] R. K. Ahuja, T. L. Magnanti et J. B. Orlin. *Network Flows : Theory, Algorithms and Applications*. Prentice Hall, 1993.

[Anderberg, 1973] M. Anderberg. *Cluster analysis for applications*. Academic Press, 1973.

[Astudillo, 1997] H. Astudillo. Maximizing object reuse with biological metaphor. *Theory and Practice of Object Systems*, 3(4) :235–251, 1997.

[Augier *et al.*, 1995] S. Augier, G. Venturini et Kodratoff Y. Learning First Order Logic Rules with a Genetic Algorithm. Dans *Proceedings of the 1st Int. Conf. on Knowledge Discovery in DataBases (KDD'95), Montréal, Canada*, pages 21–26. AAAI Press, 1995.

[Barbut et Monjardet, 1970] M. Barbut et B. Monjardet. *Ordre et Classification : Algèbre et combinatoire*. Hachette, 1970.

[Bertrand et Diday, 1990] P. Bertrand et E. Diday. Une généralisation des arbres hiérarchiques : les représentations pyramidales. *Revue de Statistique Appliquée*, XXXVIII(3) :53–78, 1990.

[Bisson, 1992] G. Bisson. Conceptual clustering in a first order logic representation. Dans *Proceedings of the 10th European Conference on Artificial Intelligence, Vienna, Austria*, pages 458–462, 1992.

[Bisson, 1993] G. Bisson. *KBG : induction de bases de connaissances en logique des prédicats*. Thèse de doctorat, Université Paris-Sud, Avril 1993.

[Bisson, 1995] G. Bisson. Why and how to define a similarity measure for object-based representation systems. Dans *Towards Very Large Knowledge Bases*, rédacteur N.J.I. Mars, pages 236–246, Amsterdam, 1995. IOS Press.

[Bisson, 1997] G. Bisson. La similarité : une notion symbolique/numérique. Dans *Induction symbolique/numérique*. 1997.

[Bobrow et Winograd, 1977] D. G. Bobrow et T. Winograd. An Overview of KRL, a Knowledge Representation Language. *Cognitive Science*, 1(1) :3–46, 1977.

[Booch, 1994] G. Booch. *Object-oriented analysis and design with applications*. Benjamin/Cummings Publishing, 1994.

[Bordat, 1986] J.-P. Bordat. Calcul pratique du treillis de Galois d'une correspondance. *Mathématiques et Sciences Humaines*, 96 :31–47, 1986.

[Bournaud et Ganascia, 1995] I. Bournaud et J.-G. Ganascia. Conceptual clustering of complex objects : A generalization space based approach. *Lecture Notes in Computer Science*, 954 :173–187, 1995.

[Bournaud, 1996] I. Bournaud. *Regroupement conceptuel pour l'organisation de connaissances*. Thèse de doctorat, Université Pierre et Marie Curie, Paris, 1996.

[Brachman *et al.*, 1991] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick et A. Borgida. Living with Classic : When and How to Use a KL-ONE Language. Dans *Principles of Semantic Networks*, rédacteur J. F. Sowa, chapitre 14, pages 400–456. Morgan Kaufmann, 1991.

[Brachman *et al.*, 1993] R. Brachman, P. Selfridge, L. Terveen, B. Altman, A. Borgida, F. Halper, T. Kirk, A. Lazar, D. McGuinness et L. Resnick. Integrated support for data archaeology. *International Journal of Intelligent and Cooperative Information Systems*, 2 :159–185, 1993.

[Brachman et Levesque, 1985] rédacteurs R. J. Brachman et H. Levesque. *Readings in Knowledge Representation*. Morgan Kaufman, 1985.

[Brachman et Schmolze, 1985] R. J. Brachman et J. G. Schmolze. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science*, 9(2) :171–216, 1985.

[Brachman, 1977] R.J. Brachman. What's in a concept : Structural foundations for semantic networks. *International Journal of Man-Machine Studies*, 9 :127–152, 1977.

[Brachman, 1985] R. J. Brachman. "I lied about the trees", Or Defaults and Definitions in Knowledge Representation. *AI Magazine*, 6(3) :80–93, 1985.

[Brito, 1991] P. Brito. *Analyse de données symboliques. Pyramides d'héritage*. Thèse de doctorat, Université Paris IX Dauphine, 1991.

[Buneman *et al.*, 1997] P. Buneman, S. Davidson, M. Fernandez et D. Suciu. Adding structure to unstructured data. Dans *Proceedings of the ICDT'97 Conference*, rédacteurs F. N. Afrati et P. G. Kolaitis, volume 1186 de *Lecture Notes in Computer Science*, pages 336–350. Springer-Verlag, 1997.

[Capponi *et al.*, 1995] C. Capponi, J. Euzenat et J. Gensel. Objects, types and constraints as classification schemes. Dans *International Conference on Knowledge Re-use, Storage and Efficiency (KRUSE'95), Santa Cruz (CA), USA*, rédacteurs G. Ellis, R.A. Levinson, A. Fall et V. Dahl, pages 69–73, 1995.

[Capponi, 1995] C. Capponi. *Identification et exploitation des types dans un modèle de connaissances à objets*. Thèse d'informatique, Université Joseph Fourier, Grenoble (FR), 1995.

[Capponi, 1997] C. Capponi. Design and implementation of type system for a knowledge representation system. Research Report 3096, INRIA, January 1997.

[Carpineto et Romano, 1996] C. Carpineto et G. Romano. Information Retrieval through Hybrid Navigation of Lattice Representations. *International Journal of Human-Computer Studies*, 45(5) :553–578, 1996.

[Carpinetto et Romano, 1996] C. Carpinetto et G. Romano. A Lattice Conceptual Clustering System and its Application to Browsing retrieval. *Machine Learning*, 24(2) :95–122, 1996.

[Carré *et al.*, 1995] B. Carré, R. Ducournau, J. Euzenat, A. Napoli et F. Rechenmann. Classification et objets : programmation ou représentation ? Dans *Actes des 5ème journées nationales du PRC-GDR « Intelligence Artificielle »*, pages 213–237, Nancy (France), Février 1995.

[Carré, 1989] B. Carré. *Méthodologie orientée objet pour la représentation des connaissances. Concepts de points de vue, de représentation multiple et évolutive d'objet.* Thèse d'Informatique, Université des Sciences et Technologies de Lille, 1989.

[Caseau, 1993] Y. Caseau. Efficient handling of multiple inheritance hierarchies. Dans *Proceedings of the ACM OOPSLA Conference, Washington, DC, USA*, rédacteur A. Paepcke, pages 271–287. ACM Press, 1993.

[Causse et Nédellec, 1992] K. Causse et C. Nédellec. Knowledge refinement using knowledge acquisition and machine learning methods. Dans *European Knowledge Acquisition Workshop, Heidelberg and Kaiserslautern (EKAW'92), LNAI 599*, pages 171–190. Springer-Verlag, 1992.

[Celeux *et al.*, 1989] G. Celeux, E. Diday, G. Govaert, Y. Lechevallier et H. Ralambondrainy. *Classification automatique des données*. Dunod, 1989.

[Cheeseman *et al.*, 1988] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor et D. Freeman. Autoclass : A bayesian classification system. Dans *Proceedings of the 5th Internatinal Conference on Machine Learning, Ann Arbor (MI)*, pages 54–56, 1988.

[Chein et Mugnier, 1992] M. Chein et M.-L. Mugnier. Conceptual graphs : Fundamental notions. *Revue d'Intelligence Artificielle*, 6(2) :365–406, 1992.

[Chein, 1969] M. Chein. Algorithme de recherche des sous-matrices premières d'une matrice. *Bull. Math. de la soc. Sci. de la R.S. de Roumanie*, 13, 1969.

[Chouvet *et al.*, 1996] M.-P. Chouvet, F. Le Ber, J. Lieber, L. Mangelinck, A. Napoli et A. Simon. Analyse des besoins en représentation et raisonnement dans une représentation à objets — L'exemple de Y3. Dans *Actes du Colloque Langages et Modèles à Objets (LMO'96), Leysin, Suisse*, rédacteur Y. Dennebouy, pages 150–169. École Polytechnique Fédérale de Lausanne, 1996.

[Cohen, 1996] W. Cohen. Learning Trees and Rules with Set-valued Features. Dans *Proceedings of the 13th AAAI*, 1996.

[Cook, 1992] W. R. Cook. Interfaces and specifications for the Smalltalk-80 Collection classes. Dans *Proceedings of OOPSLA'92*, rédacteur A. Paepcke, ACM SIGPLAN, pages 1–15, Vancouver, (CA), 1992. ACM Press.

[Crampé et Euzenat, 1996] Y. Crampé et J. Euzenat. Révision interactive dans une base de connaissan ces à objets. Dans *10ème congrès Reconnaissances des Formes e t Intelligence Artificielle*, pages 615–623, Rennes (France), Janvier 1996. AFCET-AFIA.

[Crampé, 1997] I. Crampé. *Révision interactive dans une base de connaissance à objets.* Thèse d'informatique, Université Joseph Fourier, Grenoble (FR), 1997.

[Daniel-Vatonne, 1993] M.-C. Daniel-Vatonne. *Les termes : un modèle de représentation et structuration de données symboliques.* Thèse de doctorat, Université Montpellier II, 1993.

[Davey et Priestley, 1992] B. A. Davey et H. A. Priestley. *Introduction to lattices and order.* Cambridge University Press, 1992.

[Davis, 1987] H. E. Davis. VIEWS : Multiple Perspectives and Structured Objects in a Knowledge Representation Language. Bachelor and master of scinece thesis, MIT, Cambridge, MA, 1987.

[Day et Edelsbrunner, 1984] W. H. E. Day et H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1 :7–24, 1984.

[Dayhoff *et al.*, 1978] M. O. Dayhoff, R. M. Schwartz et B. C. Orcutt. A model for evolutionary change in proteins. Dans *Atlas of Protein Sequence and Structure*, rédacteur M. O. Dayhoff, volume 5, pages 345–352. National Biochemical Research Foundation, Washington DC, 1978.

[de A. T. de Carvalho, 1998] F. de A. T. de Carvalho. Statistical proximity functions of boolean symbolic objects based on histograms. Dans *Advances in Data Science and Classification*, rédacteurs A. Rizzi, M. Vichi et H.-H. Bock, pages 391–396. Springer-Verlag, Berlin, 1998.

[De Giacomo et Lenzerini, 1997] G. De Giacomo et M. Lenzerini. A uniform framework for concept definitions in description logics. *Journal of Artificial Intelligence Research*, 6 :87–110, 1997.

[Decastaecker, 1993] C. Decastaecker. Apprentissage et outils statistiques en classification conceptuelle incrémentale. *Revue d'Intelligence Artificielle*, 7(1) :33–77, 1993.

[Delobel *et al.*, 1991] C. Delobel, C. Lécluse et P. Richard. *Bases de données : des systèmes relationnels aux systèmes à objets.* InterEditions, 1991.

[Dicky *et al.*, 1994] H. Dicky, C. Dony, M. Huchard et T. Libourel. Un algorithme d'insertion avec restructuration dans les hiérarchies de classes. Dans *Proceedings of Langages et Modèles à Objets, Grenoble, France*, pages 125–136, 1994.

[Dicky *et al.*, 1996] H. Dicky, C. Dony, M. Huchard et T. Libourel. On automatic class insertion with overloading. Dans *Proceedings of OPPSLA'96*, pages 251–267, 1996.

[Diday *et al.*, 1982] E. Diday, J. Lemaire, J. Pouget et F. Testu. *Elements d'analyse de données.* Dunod, 1982.

[Diday, 1998] E. Diday. Symbolic data analysis : a mathematical framework and tool for data mining. Dans *Advances in Data Science and Classification*, rédacteurs A. Rizzi, M. Vichi et H.-H. Bock, pages 409–416. Springer-Verlag, Berlin, 1998.

[Ducournau *et al.*, 1995] R. Ducournau, M. Habib, M. Huchard, M.-L. Mugnier et A. Napoli. Le point sur l'héritage multiple. *Technique et Science Informatiques*, 14(3) :309–345, 1995.

[Ducournau *et al.*, 1999] R. Ducournau, M. Huchard, T. Libourel et A. Napoli. Aspects classifcatoires des systemes à objets. Dans *Actes des Septièmes Journées de la SFC, Nancy*, rédacteurs F. Le Ber, J.-F. Mari, A. Napoli et A. Simon, pages 45–52. INRIA Lorraine, 1999.

[Ducournau, 1988] R. Ducournau. *YAFOOL version 3.22*. SEMA METRA, Montrouge (France), 1988.

[Ducournau, 1998] R. Ducournau. La logique des objets : application à la classification incertaine. Dans *Langages et modèles à objets : état et perspectives de la recherche*, rédacteurs R. Ducournau, J. Euzenat, G. Masini et A. Napoli, numéro 19 dans Didactique, pages 363–391. INRIA, 1998.

[Duda et Hart, 1973] R. O. Duda et P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons Inc., New York, 1973.

[Dugerdil, 1988] P. Dugerdil. *Contribution à l'étude de la représentation des connaissances fondée sur les objets. Le langage OBJLOG*. Thèse d'Informatique, Université d'Aix-Marseille II, 1988.

[Emde et Wettschereck, 1996] W. Emde et D. Wettschereck. Relational instance based learning. Dans *Machine Learning - Proceedings 13th International Conference on Machine Learning*, rédacteur L. Saitta, pages 122–130. Morgan Kaufmann Publishers, 1996.

[Euzenat, 1993a] J. Euzenat. Brief overview of t-tree : the tropes taxonomy building tool. Dans *Proceedings of the 4th ASIS SIG/CR classification research workshop, Columbus (OH US)*, pages 69–87, 1993.

[Euzenat, 1993b] J. Euzenat. On a purely taxonomic and descriptive meaning for classes. Dans *13th IJCAI Workshop on Object-Based Representation Systems*, pages 81–92, Chambéry, France, 1993.

[Euzenat, 1994] J. Euzenat. Classification dans les représentations par ob jets : produits de systèmes classificatoires. Dans *9ème RFIA*, rédacteur AFCET, volume 2, pages 185–196, Paris (France), Janvier 1994.

[Euzenat, 1998] J. Euzenat. Représentation de connaissances par objets. Dans *Langages et modèles à objets : état et perspectives de la recherche*, rédacteurs R. Ducournau, J. Euzenat, G. Masini et A. Napoli, numéro 19 dans Didactique, pages 293–319. INRIA, 1998.

[Euzenat, 1999] J. Euzenat. Sémantique des représentations de connaissance. Notes de cours de DEA d'Informatique, UJF, Grenoble, 1999.

[Faid *et al.*, 1999] M. Faid, R. Missaoui et R. Godin. Knowledge discovery in complex objects. *Computational Intelligence*, 15(1) :28–49, 1999.

[Fatemi et Mulhem, 1999] N. Fatemi et P. Mulhem. A conceptual graph approach for video data representation and retrieval. Dans *Proceedings of the 3rd International Symposium on Intelligent Data Analysis.*, rédacteurs D. Hand, J. Kok et M. Berthold, volume 1642 de *Lecture Notes in Computer Science*, pages 525–536, 1999.

[Fisher et Pazzani, 1991] D. Fisher et M. Pazzani. *Knowledge and experience in unsupervised learning*, chapitre Computational Models of Concept Learning, pages 307–322. Morgan Kaufman, San Mateo (CA US), 1991.

[Fisher, 1987] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2 :139–172, 1987.

[Fisher, 1996] D. H. Fisher. Iterative optimization and simplification of hierarchical clusterings. *Journal of Artificial Intelligence Research*, 4 :147–180, 1996.

[Forgy, 1965] E. W. Forgy. Cluster analysis of multivariate data : efficiency versus interpretability of classifications. *Biometrics*, 21, 1965.

[Ganter et Wille, 1989] B. Ganter et R. Wille. *Applications of combinatorics and graph theory to the biological and social sciences*, volume 17 de *The IMA volumes in Mathematics and its applications*, chapitre Conceptual Scaling, pages 139–167. Springer-Verlag, New York, 1989.

[Ganter, 1984] B. Ganter. Two basic algorithms in concept analysis. preprint 831, Technische Hochschule, Darmstadt, 1984.

[Garey et Johnson, 1979] M. Garey et D. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 1979.

[Gennari *et al.*, 1989] J. H. Gennari, P. Langley et D. H. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40,(1-3) :11–61, 1989.

[Gentner, 1988] D. Gentner. Structure-mapping : A theoretical framework for analogy. Dans *Readings in Cognitive Science*, pages 303–310. Morgan Kaufman, 1988.

[Girard, 1997] R. Girard. *Classification Conceptuelle sur des Données Arborescentes et Imprécises*. Thèse de doctorat, Université de la Réunion, 1997.

[Gluck et Corter, 1985] M. Gluck et J. Corter. Information uncertainty and the utility of categories. Dans *Proceedings of the 7th Annual Conference of the Cognitive Science Society*, pages 283–287, 1985.

[Godin *et al.*, 1991] R. Godin, R. Missaoui et H. Alaoui. Learning algorithms using galois lattices. Dans *Proceedings of the IEEE International Conference on Tools for Artificial Intelligence, San Jose, CA, USA*, pages 22–29, 1991.

[Godin *et al.*, 1995a] R. Godin, G. W. Mineau, R. Missaoui, M. St-Germain et N. Faraj. Applying concept formation methods to software reuse. *Journal of Knowledge Engineering and Software Engineering*, 5(1) :119–142, 1995.

[Godin *et al.*, 1995b] R. Godin, R. Missaoui et H. Alaoui. Incremental concept formation algorithms based on galois (concept) lattices. *Computational Intelligence*, 11(2) :246–267, 1995.

[Godin *et al.*, 1998] R. Godin, H. Mili, G. W. Mineau, R. Missaoui, A. Arfi et T. T. Chau. Design of class hierarchies based on concept (galois) lattices. *Theory and Application of Object Systems*, 4(2) :117–134, 1998.

[Goldberg et Robson, 1989] A. Goldberg et D. Robson. *Smalltalk-80 : The Language*. Addison-Wesley Series in Computer Science. Addison-Wesley, 1989.

[Golub et van Loan, 1983] G. Golub et C. van Loan. *Matrix Computations*. John Hopkins University Press, 1983.

[Gondran et Minoux, 1984] M. Gondran et M. Minoux. *Graphs and Algorithms.* John Wiley and Sons, 1984.

[Gordon, 1999] A. D. Gordon. *Classification.* Chapman & Hall/CRC, 1999.

[Guénoche, 1990] A. Guénoche. Construction du treillis de galois d'une relation binaire. *Mathématiques et Sciences Humaines*, 109 :41–53, 1990.

[Gusfield, 1997] D. Gusfield. *Algorithms on Strings, Trees, and Sequences : Computer Science and Computational Biology.* Cambridge University Press, 1997.

[Hadzikadic et Yun, 1989] M. Hadzikadic et D. Y. Y. Yun. Concept formation by incremental conceptual clustering. Dans *Proceedings of IJCAI'89, Detroit (MI), USA*, pages 831–836, 1989.

[Han *et al.*, 1994] J. Han, S. Nishio et H. Kawano. Knowledge discovery in object-oriented and active databases. Dans *Knowledge Building and Knowledge Sharing*, rédacteurs F. Fuchi et T. Yokoi, pages 221–230. Ohmsha, Ltd. and IOS Press, 1994.

[Han *et al.*, 1998] J. Han, S. Nishio, H. Kawano et W. Wang. Generalization-based data mining in object-oriented databases using an object-cube model. *Data and Knowledge Engineering*, 25(1-2) :55–97, 1998.

[Hand, 1997] D. Hand. Intelligent data analysis : Issues and opportunities. Dans *Proceedings of the 2nd Symposium on Intelligent Data Analysis*, rédacteurs P.Coen X. Liu et M. Berthold, volume 1280 de *Lecture Notes in Computer Science*, pages 1–14. Springer-Verlag, 1997.

[Hanson et Bauer, 1989] S. J. Hanson et M. Bauer. Conceptual clustering, categorization and polymorphy. *Machine Learning*, 3 :343–372, 1989.

[Haton *et al.*, 1991] J.-P. Haton, N. Bouzid, F. Charpillet, M.-C. Haton, B. Lâasri, H. Lâasri, P. Marquis, T. Mondot et A. Napoli. *Le raisonnement en intelligence artificielle.* InterEditions, Paris, France, 1991.

[Hausdorff, 1962] Hausdorff. *Set Theory.* Chelsea, 1962.

[Haussler, 1989] D. Haussler. Learning conjunctive concepts in structural domains. *Machine Learning*, 4(1) :7–40, 1989.

[Hillier et Lieberman, 1995] F. S. Hillier et G. J. Lieberman. *Introduction to Operational Research.* McGraw-Hill, New York, 6th édition, 1995.

[Huttenlocher *et al.*, 1993] D. P. Huttenlocher, G. A. Klanderman et W. J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9) :850–863, 1993.

[Ichino, 1994] M. Ichino. Feature selection for symbolic data classification. Dans *New Approaches in Classification and Data Analysis*, rédacteurs E. Diday, Y. Lechevallier, M. Schader, P. Bertrand et B. Burtschy, pages 423–429. Springer-Verlag, Berlin, 1994.

[Jain et Dubes, 1988] A. Jain et R. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[Ji *et al.*, 1997] P. Ji, W. B. Lee et H. Li. A New Algorithm for the Assignment problem : an Alternative to the Hungarian Method. *Computers and Operations Research*, 24(11) :1017–1023, 1997.

[Ketterlin *et al.*, 1995] A. Ketterlin, P. Gancarski et J. J. Korczak. Hierarchical clustering of composite objects with variable number of components. Dans *Proceedings of the 5th International Workshop on Artificial Intelligence and Statistics, Fort Lauerdale, FL, USA*, rédacteurs D. H. Fisher et P. Lenz, 1995.

[Ketterlin et Korczak, 1994] A. Ketterlin et J. J. Korczak. Concept formation in complex domains. Dans *Proceedings of the ECML-94, European Conference on Machine Learning, Catania, Italy*, rédacteurs F. Bergadano et L. De Raedt, volume 784 de *Lecture Notes in Computer Science*, pages 371–374, 1994.

[Ketterlin, 1995] A. Ketterlin. *Découverte de Concepts Structurés dans les Bases de Données.* Thèse de doctorat, Université Louis Pasteur, Strasbourg, 1995.

[Kietz et Morik, 1994] J.-U. Kietz et K. Morik. A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14(2) :193–217, 1994.

[Kodratoff, 1997] Y. Kodratoff. L'extraction de connaissances à partir de données : un nouveau sujet pour la recherche scientifique. Dans *Actes des Journées Ingénierie des Connaissances et Apprentissage Automatique, Roscoff, France*, rédacteur J. Charlet, pages 539–566. INRIA Rennes, 1997.

[Kosters *et al.*, 1999] W. A. Kosters, E. Marchiori et A. A. J. Oerlemans. Mining clusters with association rules. Dans *Proceedings of the 3rd International Symposium on Intelligent Data Analysis.*, rédacteurs D. Hand, J. Kok et M. Berthold, volume 1642 de *Lecture Notes in Computer Science*, pages 39–50, 1999.

[Kuhn, 1955] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics*, 2 :83–97, 1955.

[Lance et Williams, 1967a] G. N. Lance et W. T. Williams. A general theory of classificatory sorting strategies : I. Hierarchical Systems. *Computer Journal*, 9 :373–380, 1967.

[Lance et Williams, 1967b] G. N. Lance et W. T. Williams. A general theory of classificatory sorting strategies : II. Clustering Systems. *Computer Journal*, 10 :271–277, 1967.

[Lavrac et Dzeroski, 1994] N. Lavrac et S. Dzeroski. *Inductive Logic Programming.* Ellis Horwood, Chichester/New York, 1994.

[Leclerc, 1998] B. Leclerc. Consensus of classifications : the case of trees. Dans *Advances in data science and classification*, rédacteurs A. Rizzi, M. Vichi et H.-H. Bock, pages 81–90, Heildelberg (DE), 1998. Springer-Verlag.

[Levesque et Brachman, 1987] H. J. Levesque et R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3 :78–93, 1987.

[Levinson et Ellis, 1992] R. Levinson et G. Ellis. Multi-level hierarchical retrieval. *Knowledge Based Systems*, 5(3) :233–244, 1992.

[Liquiere et Sallantin, 1998] M. Liquiere et J. Sallantin. Structural machine learning with galois lattice and graphs. Dans *Proceedings of the 15th International Conference on Machine Learning*, pages ? ?–? ? Morgan Kaufmann Publishers, 1998.

[MacQueen, 1967] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. Dans *Proceedings of the 5th Berkeley Symposium*, volume 2, pages 281–297, 1967.

[Margush et McMorris, 1981] T. Margush et F. R. McMorris. Consensus n-trees. *Bull. Math. Biol*, 43 :239–244, 1981.

[Mariño *et al.*, 1990] O. Mariño, F. Rechenmann et P. Uvietta. Multiple perspectives and classification mechanism in object-oriented representation. Dans *9th European Conference on Artificial Intelligence*, pages 425–430, Stockholm, Suède, Août 1990.

[Martin et Billman, 1994] J. D. Martin et D. O. Billman. Acquiring and combining overlapping concepts. *Machine Learning*, 16 :121–155, 1994.

[Martin et Eklund, 1999] P. Martin et P. Eklund. Embedding Knowledge in Web Documents : CGs versus XML-Based Metadata. Dans *Conceptual structures : Standards and Practices*, rédacteurs W. M. Tepfenhart et W. Cyre, volume 1640 de *Lecture Notes in Computer Science*, pages 230–246. Springer-Verlag, Berlin, 1999.

[Masini *et al.*, 1989] G. Masini, A. Napoli, D. Colnet, D. Léonard et K. Tombre. *Les langages à objets*. InterEditions, Paris, 1989.

[McKusick et Langley, 1991] K. B. McKusick et P. Langley. Constraints on tree structure in concept formation. Dans *Proc. of the 12 th IJCAI*, pages 810–816, Sidney, Australie, 1991.

[Michalski et Stepp, 1983] R. S. Michalski et R. Stepp. Learning from observation : conceptual clustering. Dans *Machine learning : an Artificial Intelligence approach*, rédacteurs R. S. Michalski, J. G. Carbonell et T. M. Mitchell, volume I, pages 331–363. Tioga publishing company, Palo Alto (CA US), 1983.

[Michalski, 1983] R. S. Michalski. A theory and methodology of inductive learning. Dans *Machine Learning : An Artificial Intelligence Approach*, rédacteurs R. S. Michalski, J. G. Carbonell et T. M. Mitchell, volume I, pages 83–134, Palo Alto, CA, 1983. Tioga.

[Mineau, 1990] G. W. Mineau. *Structuration de Bases de connaissance*. PhD thesis, Université de Montréal, 1990.

[Minsky, 1975] M. Minsky. A framework for representing knowledge. Dans *The psychology of computer vision*, rédacteur P. Winston. McGraw-Hill, 1975.

[Mirkin, 1997] B. Mirkin. *Mathematical Clustering*. Springer-Verlag, 1997.

[Missaoui *et al.*, 1998] R. Missaoui, R. Godin et H.A. Sahraoui. Migrating to an object-oriented database using semantic clustering and metamodelling techniques. *Data and Knowledge Engineering*, 27(1) :97–113, 1998.

[Missikoff et Scholl, 1989] M. Missikoff et M. Scholl. An algorithm for insertion into a lattice : Application to type classification. Dans *Proceedings of the 3rd International FODO Conference*, numéro 367 dans Lecture Notes in Computer Science, 1989.

[Mitchell, 1990a] T. M. Mitchell. Generalisation as Search. Dans *Readings in Machine Learning*, pages 96–107. Morgan Kaufmann, 1990.

[Mitchell, 1990b] T. M. Mitchell. The Need of Biases in Learning Generalisations. Dans *Readings in Machine Learning*, rédacteurs J. Shavlik et T. Dietterich, pages 184–191. Morgan Kaufmann, 1990.

[Mitchell, 1997] T. M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[Montjardet, 1981] B. Montjardet. Metrics on partially ordered sets – a survey. *Discrete Mathematics*, 35(1) :173–184, 1981.

[Morik *et al.*, 1993] K. Morik, S. Wrobel, J.-U. Kietz et W. Emde. *Knowledge Acquisition and Machine Learning - Theory, Methods, and Applications.* Academic Press, 1993.

[Morik et Brockhausen, 1997] K. Morik et P. Brockhausen. A multistrategy approach to relational knowledge discovery in databases. *Machine Learning*, 27(3) :287–312, 1997.

[Muggleton et Buntine, 1988] S. Muggleton et W. Buntine. Machine invention of first order predicates by inverting resolution. Dans *Proceedings of the 5th International Conference on Machine Learning, University of Michigan, Ann Arbor (MI)*, rédacteur J. Laird, pages 339–352. Morgan Kaufmann, 1988.

[Muggleton et Feng, 1992] S. Muggleton et C. Feng. *Inductive Logic Programming*, chapitre Efficient Induction of Logic Programs. Academic Press, 1992.

[Mugnier et Chein, 1996] M.-L. Mugnier et M. Chein. Représenter des connaissances et raisonner avec des graphes. *Revue d'Intelligence Artificielle*, 10(1) :7–56, 1996.

[Napoli, 1992] A. Napoli. *Représentation à objets et raisonnement par classification en Intelligence Artificielle.* Doctorat d'état ès sciences mathématiques, Université de Nancy I, Janvier 1992.

[Napoli, 1997] A. Napoli. Une introduction aux logiques de descriptions. Rapport de Recherche RR-3314, INRIA, 1997.

[Nebel, 1990] B. Nebel. *Reasoning and revision in hybrid representation systems*, volume 422 de *Lecture Notes in Artificial Intelligence.* Springer-Verlag, Berlin (DE), 1990.

[Nebel, 1991] B. Nebel. Terminological Cycles : Semantics and Computational Properties. Dans *Principles of Semantic Networks : Explorations in the Representation of Knowledge*, rédacteur J. F. Sowa, pages 331–361. Morgan Kaufmann Publishers, San Mateo (CA), USA, 1991.

[Nédellec *et al.*, 1995] C. Nédellec, J.-L. Ferreira, J. Correia et E. Costa. Machine learning goes to the bank. *Applied Artificial Intelligence*, 8(4) :593–615, 1995. Special Issue on Machine Learning.

[Needleman et Wunsch, 1970] S. Needleman et C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48 :443–453, 1970.

[Norris, 1978] E. M. Norris. An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées*, 23(2) :243–250, 1978.

[Papadimitriou et Steiglitz, 1998] C. H. Papadimitriou et K. Steiglitz. *Combinatorial Optimization : Algorithms and Complexity.* Prentice-Hall, 1998.

[Piatetsky-Shapiro et Frawley, 1991] G. Piatetsky-Shapiro et W. Frawley. *Knowledge discovery in databases*. AAAI Press, 1991.

[Polaillon, 1998] G. Polaillon. *Organisation et interprétation par les treillis de Galois de données de type multivalué, intervalle ou histogramme*. Thèse de doctorat, Université Paris IX Dauphine, 1998.

[Puerta *et al.*, 1992] A. R. Puerta, J. W. Egar, S. W. Tu et M. A. Musen. A multiple-method knowledge acquisition shell for the automatic generation of knowledge acquisition tools. *Knowledge Acquisition*, 4 :171–196, 1992.

[Quillian, 1968] M. R. Quillian. Semantic memory. Dans *Semantic Information Processing*, rédacteur M. Minsky, pages 227–270. MIT Press, Cambridge, MA, 1968.

[Quinlan et Cameron-Jones, 1993] J. R. Quinlan et R. M. Cameron-Jones. FOIL : A midterm report. Dans *Proceedings of the European Conference on Machine Learning (ECML-93)*, rédacteur P. B. Brazdil, volume 667 de *LNAI*, pages 3–20, Vienna, Austria, Avril 1993. Springer-Verlag.

[Quinlan, 1990] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5 :239–266, 1990.

[Rechenmann *et al.*, 1990] F. Rechenmann, P. Fontanille et P. Uvietta. *SHIRKA : Système de gestion de bases de connaiss ances centrées-objets*. INRIA et laboratoire Artémis / IMAG, Grenoble, France, Octobre 1990.

[Rieu *et al.*, 1992] D. Rieu, G. T. Nguyen et J. Escamilla. Shood : An object model for engineering design. Dans *Actes de l'AVIGNON'92*, pages 295–303, Avignon, France, 1992. EC2.

[Roberts et Goldstein, 1977] R. B. Roberts et I. P. Goldstein. *The FRL Manual*. AI Lab, MIT, Cambridge, MA, 1977.

[Rosch et Mervis, 1975] E. Rosch et C. Mervis. Family resemblances : Studies in the internal structure of categories. *Cognitive Psychology*, 7 :573–605, 1975.

[Salotti, 1992] S. Salotti. Fuzzy filtering on objects for case-based reasoning : the floran system. Dans *Applications Des Ensembles Flous*, pages 335–342. EC2, Nimes, France, 1992.

[Saporta, 1990] G. Saporta. *Probabilités, Analyse des Données et Statistique*. Technip, 1990.

[Schreiber *et al.*, 1994] A. Schreiber, B. Wielinga, R. de Hoog, H. Akkermans et W. van de Velde. CommonKADS : A comprehensive methodology for KBS development. *IEEE Expert*, December, 1994.

[SHERPA, 1995] Projet SHERPA. *Tropes 1.0 reference manual*. INRIA Rhône-Alpes, Grenoble (FR), 1995.

[Simon et Napoli, 1999] A. Simon et A. Napoli. Un algorithme de fouille dans une représentation des données par objets : une application au domaine médical. Dans *Ingénierie des Connaissances, évolutions récentes et nouveaux défis*, rédacteurs J. Charlet, M. Zacklad, G. Kassel et D. Bourigaul. Eyrolles, 1999.

[Simon, 1999] A. Simon. Relations et treillis de Galois. Dans *Actes des 7èmes journées de la SFC, Nancy (FR)*, rédacteurs F. Le Ber, J.-F. Mari, A. Napoli et A. Simon, pages 299–305, Nancy, 1999. INRIA Lorraine, Unité de recherche INRIA Lorraine.

[Sowa, 1984] J. Sowa. *Conceptual Structures : Information Processing in Mind and Machine*. Addison Wesley, MA, 1984.

[Stepp et Michalski, 1986] R. Stepp et R. S. Michalski. Conceptual clustering : Inventing goal-oriented classifications of structured objects. Dans *Machine learning : an Artificial Intelligence approach*, rédacteurs R. S. Michalski, J. G. Carbonell et T. M. Mitchell, volume II, pages 471–498. Morgan Kaufman, San Mateo (CA US), 1986.

[Tarjan, 1988] R. E. Tarjan. *Data structures and network algorithms*. Numéro 44 dans CBMS-NSF regional conference series in applied mathematics. SIAM, 1988.

[Teccuci et Kodratoff, 1989] G. Teccuci et Y. Kodratoff. *Machine Learning : An Artificial Intelligence Approach*, volume III, chapitre Apprenticeship Learning in Nonhomogeneous Domain Theories. Morgan Kaufmann Publishers, 1989.

[Thompson et Langley, 1991] K. Thompson et P. Langley. Concept formation in structured domains. Dans *Concept Formation : knowledge and experience in unsupervised learning*, rédacteurs M. Pazzani D. Fisher et P. Langley, pages 127–161. Morgan Kaufman, San Mateo (CA US), 1991.

[Valtchev et Euzenat, 1996] P. Valtchev et J. Euzenat. Classification of concepts through products of concepts and abstract data types. Dans *Ordinal and symbolic data analysis*, rédacteurs Y. Lechevallier E. Diday et Dr O. Opitz, pages 3–12, Heildelberg (DE), 1996. Springer-Verlag.

[Valtchev, 1997] P. Valtchev. Automatic taxonomy building within an advanced object formalism. Dans *Proceedings of the 2nd Indo-French Workshop on Symbolic Data Analysis and its Applications*, rédacteurs M. Csernel, C. Pardoux et E. Perinel, volume II, pages 200–211. Université Paris IX - Dauphine, 1997.

[Valtchev, 1999] P. Valtchev. An algorithm for minimal insertion in a type lattice. *Computational Intelligence*, 15(1) :63–78, 1999.

[van Cutsem, 1994] B. van Cutsem. *Classification and dissimilarity analysis*, volume 93 de *Lecture notes in statistics*. Springer Verlag, New York, 1994.

[Vasco *et al.*, 1996] J. J. F. Vasco, C. Faucher et E. Chouraqui. A knowledge acquisition tool for multi-perspective concept formation. Dans *Proceedings of the 9th European Knowledge Acquisition Workshop, EKAW'96*, rédacteur Sprin ger Verlag, pages 227–244, 1996.

[Webb *et al.*, 1999] G. I. Webb, J. Wells et Z. Zheng. An experimental evaluation of integrating machine learning with knowledge acquisition. *Machine Learning*, 35(1) :5–23, 1999.

[Wille, 1992] R. Wille. Concept lattices and conceptual knowledge systems. *Computers and Mathematics with Applications*, 23 :493–515, 1992.

[Winston *et al.*, 1987] M. E. Winston, R. Chaffin et D. Herrmann. A taxonomy of part-whole relations. *Cognitive Science*, 11 :417–444, 1987.

[Woods et Schmolze, 1992] W. A. Woods et J. G. Schmolze. The KL-ONE family. *Computers & Mathematics with Applications*, 23(2-5) :133–177, 1992.

[Woods, 1975] W. A. Woods. What's in a link ? : Foundations for semantic networks. Dans *Representation and Understanding : Studies in Cognitive Science*, rédacteurs D. G. Bobrow et A. Collins, pages 35–82. Academic Press, New York, 1975.

[Zucker et Ganascia, 1996] J.-D. Zucker et J.-G. Ganascia. Changes of Representation for Efficient Learning in Structural Domains. Dans *Proceedings of the 13th ICML, Bari (IT)*. Morgan Kaufmann, 1996.

Résumé

La taxonomie de classes constitue le squelette d'une base de connaissances à objets, sa conception est une tâche délicate pour laquelle aucune méthodologie universelle n'existe. La thèse défendue est que le concepteur d'une taxonomie peut être aidé par des outils automatiques de construction de classes à condition que ceux-ci puissent travailler directement sur les structures de connaissance, qu'ils prennent en compte de manière intégrale les descriptions de ces structures et qu'ils démeurent suffisamment flexibles.

Afin de satisfaire ces exigences, une approche par proximité a été choisie parmi les techniques de regroupement conceptuel existantes. Elle a été adaptée à un modèle d'objets avec des relations inter-objets multivaluées et/ou circulaires où multiples taxonomies peuvent coexister. L'approche définie s'appuie sur le découpage de la base en familles d'objets. Une stratégie permet d'organiser ces familles en des ensembles indissociables pour la construction de classes puis d'appliquer à chaque ensemble une méthode de construction tout en suivant les relations entre familles. La méthode proposée fonctionne, contrairement à l'accoutumée, en deux phases successives, regroupement et caractérisation.

Lors de la première phase, les objets sont regroupés selon une mesure de dissimilarité qui s'applique de manière uniforme à tous leurs d'attributs : propriétés ou relations, mono- ou multivalués. L'exploration totale des relations n'étant pas toujours souhaitable, cette fonction permet de tirer parti des taxonomies existantes si nécessaire. Sur les couples d'objets co-définis, la dissimilarité s'exprime comme le point-fixe d'une fonction vectorielle composée par les co-définitions. Lors de la seconde étape, les groupes d'objets sont transformés en classes et munis d'une description en intention produite par généralisation à partir des descriptions des objets membres.

Les travaux présentés ont été implantés au sein du système à objets TROEPS sous forme d'un module de construction automatique de taxonomies, T-TREE.

Mots-clés: représentation de connaissances par objets, types, relations, collections, circularité, conception de taxonomies de classes, dissimilarité entre objets, résolution de systèmes d'équations.

Abstract

The class taxonomy is the back-bone of any object knowledge base, there is, however no consensual procedure for taxonomy design. Our thesis is that automatic classification tools could successfully assist the design of a knowledge base. For that purpose, the tools should directly apply on knowledge constructs, process the construct descriptions thoroughly and remain flexible.

The proximity-based approach has been chosen among other class-building techniques for its highest satisfaction of the above requirements. It fas been extended to fit into an object model with circular and/or multi-valued inter-object relations where multiple taxonomies may coexist. Our approach relies on the division of a knowledge base into object families. Families are further separated into sets which are indistinguishable for class construction. A strategy allows a subsequent applications of the class building method on every such set whereby inter-family relations are respected. The method itself proceeds by first clustering the objects and then turning the obtained clusters into intentionally described classes.

During the clustering stage, objects are compared by means of a dissimilarity measure which universally applies on their attributes : properties or relationships, single- or multi-valued ones. To avoid unnecessary searches through the relational structure of the objects, the dissimilarity relies on already existing taxonomies. When applied on mutually-dependent object couples, it becomes the fixed point of a composite function derived from the dissimilarity definitions. During the characterization stage, a generalization procedure infers a description for each cluster from the description of its members.

All the presented tools have been implemented within a taxonomy building environment, T-TREE, associated with the TROEPS object knowledge manager.

Keywords: object-based knowledge representation, types, relations, collections, circularity, class taxonomy design, object dissimilarity, solution of linear equation systems.