Aspects actuels des représentations de connaissances par objets et de la classification

Amedeo Napoli, Isabelle Crampé, Roland Ducournau, Jérôme Euzenat, Michel Leclère, Philippe Vismara

▶ **To cite this version:**

Amedeo Napoli, Isabelle Crampé, Roland Ducournau, Jérôme Euzenat, Michel Leclère, et al.. Aspects actuels des représentations de connaissances par objets et de la classification. 6e journées nationales PRC-GDR intelligence artificielle, Mar 1997, Grenoble, France. pp.289-314. hal-01401182

HAL Id: hal-01401182
https://hal.science/hal-01401182

Submitted on 23 Nov 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Aspects actuels des représentations de connaissances par objets et de la classification

Collectif d'auteurs
Rédaction supervisée par Amedeo Napoli
CRIN CNRS – INRIA Lorraine, Nancy
(napoli@loria.fr)

Résumé

Cet article présente certains thèmes de recherches étudiés par les membres du groupe « Objets et classification » du PRC IA. Ces thèmes concernent essentiellement la théorie des systèmes de représentation de connaissances par objets (RCPO), la révision d'une base de connaissances dans les systèmes de RCPO, la classification de classes et d'instances, et enfin la mise en œuvre d'applications, illustrée ici par le système RESYN. Les travaux présentés montrent une certaine continuité avec les préoccupations des membres du groupe depuis qu'il existe. L'article se termine par la présentation d'éléments de définition d'un système de RCPO, et de perspectives de recherches découlant des thèmes explicités dans l'article.

1 Introduction

Cet article présente un certain nombre de thèmes de recherches qui sont actuellement étudiés par les membres du groupes « Objets et classification » du GDR – PRC IA. Trois thèmes principaux se détachent dans ce qui suit. Le premier thème est théorique et traite d'une part des *systèmes classificatoires*, qui constituent, au moins à l'heure actuelle, les premiers éléments d'une théorie des systèmes de représentation de connaissances par objets — « systèmes de RCPO » dans la suite — et d'autre part de la *révision* dans une base d'objets. Le second thème est en relation avec la *classification* sous ses deux formes standards en RCPO : (1) conception de hiérarchies (de classes) et traitement de l'évolution de ces hiérarchies, ou la classification vue comme un outil de construction de hiérarchies, (2) le raisonnement par classification dans une

RCPO, ou la classification vue comme un mode de raisonnement ; il est question aussi de classification dans le modèle des graphes conceptuels, classification qui se ramène aux deux formes précédentes. Le troisième thème abordé dans l'article est la *conception d'applications*, qui constitue une des préoccupations les plus importantes des membres du groupe. Théorie et pratique se rencontrent et s'alimentent mutuellement et de façon continue, au moins pour ce qui concerne les études et réalisations effectuées dans l'univers des RCPO. C'est par l'intermédiaire du système d'aide à la planification de synthèses en chimie organique appelé RESYN que sont illustrées la pratique de la théorie et la théorie de la pratique de la classification et des objets. En forme de bilan de ce qui précède, des éléments de définition d'un système de RCPO sont proposés, ainsi que certaines directions de recherches qui en découlent.

Cette présentation donne également en substance le plan de l'article. Pour terminer, voici la liste des auteurs/contributeurs dont les écrits ont été collectés et fusionnés pour confectionner cet article : Isabelle Crampé (INRIA Rhône-Alpes), Roland Ducournau (LIRMM Montpellier), Jérôme Euzenat (INRIA Rhône-Alpes), Olivier Guinaldo (IUT de Clermont-Ferrand à Aubière, et LIRMM Montpellier), Marianne Huchard (LIRMM, Montpellier), Michel Leclère (IRIN – IUT de Nantes), Amedeo Napoli (CRIN CNRS – INRIA Lorraine), Philippe Vismara (ENSA-M et LIRMM Montpellier).

2 Vers une théorie des objets

2.1 Les systèmes classificatoires : introduction

Le succès des objets est justifié habituellement par les qualités que recherche et promeut le génie logiciel : modularité, extensibilité ou réutilisabilité sont des références obligées. Pourtant, une raison moins reconnue est sans doute aussi importante : les objets (informatiques) ont une capacité «naturelle» de représentation (des objets) du «monde». Cette capacité est à l'origine de la représentation des connaissances par objets en intelligence artificielle, autant que des méthodes d'analyse et de conception en génie logiciel. En fait, on peut conjecturer que ces bonnes qualités de représentation, par leur «déclarativité» [Win 75; Car 95], sont aussi à l'origine des qualités revendiquées par le génie logiciel. L'étude de la sémantique de représentation des diverses approches objet et leur comparaison dans un cadre commun est ainsi une étape nécessaire. Or, parallèlement aux développements de l'approche objet dans le courant principal du génie logiciel, un courant important de la représentation des connaissances et plus particulièrement des *réseaux sémantiques* a conduit à la définition d'une famille de langages, les *logiques de descriptions* [Neb 90], qui s'est trouvée confrontée, dès sa naissance, à ces questions de sémantique et les a résolues d'une manière tout à fait satisfaisante. Il était donc tentant d'appliquer cette recette éprouvée aux systèmes à objets plus usuels. Le cadre des *systèmes classificatoires* est une abstraction de ces diverses approches objet qui constitue une première proposition en ce sens [Euz 93] [Duc 96a] [Duc 96b].

Un système classificatoire est constitué, d'une part d'une *hiérarchie* de *classes*, décrites ou définies en *intension* par un ensemble de *propriétés* et munies d'une interprétation *extensionnelle*, d'autre part d'un ensemble d'assertions formant une description, incomplète, du monde [Duc 96a]. L'ensemble peut être considéré comme une logique, avec une sémantique directe en théorie des modèles, comme celle des logiques de descriptions [Neb 90], et un système de déduction constitué d'un ensemble de règles d'inférences ou d'algorithmes spécialisés. La justification de «classificatoire» est double : ces systèmes offrent une «classification des espèces», au sens de la hiérarchie de classes, et ils se prêtent à une «classification des individus», au sens du mécanisme d'inférence qui reconnaît qu'un individu appartient à une classe.

Cette approche classificatoire a été reconnue et mise en œuvre à la charnière des années 70 et 80 dans les langages dits de *frames* qui ont donné naissance à deux courants principaux. L'un des critères discriminant ces deux courants est le caractère *définitionnel* ou *descriptif* des classes. Les logiques de descriptions ont introduit la distinction entre classe[1] *définie* ou *primitive* [Neb 90] : pour une classe définie, la description de la classe constitue une *condition nécessaire et suffisante* d'appartenance d'un objet à l'extension de la classe. En revanche, pour une classe primitive, la description n'est qu'un ensemble de conditions nécessaires. Des classes définies autorisent la classification automatique des instances que pratiquent les logiques de descriptions sous le nom d'*instanciation* (ou encore *réalisation*). Du côté des systèmes de RCPO, la sémantique des classes est purement descriptive : aussi, pour remédier à l'impossibilité d'une classification automatique, certains de ces systèmes pratiquent, depuis SHIRKA [Rec 88] et ses successeurs TROPES [Mar 93; Euz 95] ou FROME [Dek 94], une classification que l'on peut qualifier d'*incertaine* puisqu'elle attribue à la relation classe-instance une modalité parmi *sûre*, *possible* et *impossible*.

Chacune de ces approches présente une limitation importante : pour les logiques de descriptions, les classes primitives constituent une barrière quasiment infranchissable, dans la mesure où l'appartenance à une classe primitive ne peut en général pas se déduire, alors que, de son côté, la RCPO ne peut effectuer presqu'aucune inférence automatique valide.

Le problème de la classification se pose dans un cadre de représentation par objets, au sens large, qui consiste en une *hiérarchie* de *classes* décrites en *intension* et interprétées en *extension*. Cette hiérarchie constitue une première composante, *intensionnelle*, ce que les logiques de descriptions nomment *TBox* ; il lui correspond une seconde composante, *extensionnelle*, *ABox* dans la définition (restreinte) de [Neb 90] ou de [Buc 93]. L'ensemble constitue un *système classificatoire*, expression abrégée en SC dans la suite.

1. Les logiques de descriptions utilisent en fait le terme de *concept*, mais nous utiliserons uniformément celui de *classe*.

2.2 Intensions et extensions dans une hiérarchie d'objets

La composante intensionnelle des SC est constituée d'une hiérarchie de classes décrites en intension par des *propriétés* qui s'expriment au travers d'*attributs* ; s'y ajoutent diverses relations entre ces classes.

$\mathcal{X}$ est un ensemble fini d'*entités génériques* ou *classes*, ordonné par une relation de *spécialisation* $\prec$ qui est un ordre partiel strict (et $\preceq$ est l'ordre partiel associé). Comme c'est l'habitude dans les approches objet, on parlera de *sous-classe* ou de *super-classe* en référence à cette relation. Cette hiérarchie s'interprète dans un cadre extensionnel dont nous présentons d'abord un aperçu intuitif et relativement informel : Ω est un ensemble d'*entités individuelles* ou *objets*, qui peuvent être associés aux classes par l'application :

$Ext : \mathcal{X} \longrightarrow 2^{\Omega}$ est telle que $\alpha \preceq \beta \Longrightarrow Ext(\alpha) \subseteq Ext(\beta)$.

$Ext(\alpha)$ est appelé l'*extension* de la classe α : c'est l'ensemble des objets qui «tombent sous le concept» α. Lorsque $o \in Ext(\alpha)$, o est une *instance de* α, et α est *une* (et non pas *la*) *classe de* o. Inversement, l'application $Isa : \Omega \longrightarrow 2^{\mathcal{X}}$ associe à tout objet o les classes dont il est instance, *ses* classes.

La relation de *subsomption extensionnelle* est un *préordre* sur $\mathcal{X}$, noté $\trianglelefteq$, induit par l'inclusion des extensions : $\alpha \trianglelefteq \beta \stackrel{\text{def}}{\Longleftrightarrow} Ext(\alpha) \subseteq Ext(\beta)$. Enfin, deux éléments de $\mathcal{X}$ sont distingués : un plus grand élément, noté $\top$, d'extension Ω, et un plus petit, noté $\bot$, d'extension vide.

Il faut enfin ajouter à ce cadre général une relation d'incompatibilité, notée $\alpha \nabla \beta$, qui peut se définir extensionnellement par $Ext(\alpha) \cap Ext(\beta) = Ext(\bot)$. La relation d'incompatibilité est héritable : si deux classes sont incompatibles, les sous-classes de l'une sont incompatibles avec l'autre.

Chaque classe est accompagnée d'une «description» *en intension*, par une collection de *propriétés* qui s'expriment par l'appartenance d'un *attribut* à un *domaine*. $\mathcal{P}$ est un ensemble d'attributs et, $\forall \alpha \in \mathcal{X}$, $\mathcal{P}_\alpha$ représente l'ensemble des attributs de α. L'*héritage de propriétés* consiste d'abord en une inclusion des ensembles d'attributs associés à des classes qui se spécialisent : $\alpha \prec \beta \Longrightarrow \mathcal{P}_\alpha \supseteq \mathcal{P}_\beta$. Une application $Dom : \mathcal{X} \times \mathcal{P} \longrightarrow 2^{\mathcal{D}}$ (où $\mathcal{D}$ est un domaine quelconque contenant au moins Ω ou 2^{Ω}) associe à chaque attribut d'une classe un *domaine*.

Le cadre intensionnel est relié au cadre extensionnel comme suit. D'abord, chaque objet o possède l'ensemble des attributs décrits dans ses classes :

$Attr(o) \stackrel{\text{def}}{=} \bigcup_{\alpha \in Isa(o)} \mathcal{P}_\alpha$.

Dans un objet o, chaque attribut P ($\in Attr(o)$) désigne une *valeur*, notée $P(o)$, dont $Dom(\alpha, P)$ représente les valeurs admissibles. Si $o \in Ext(\alpha)$ et $P \in \mathcal{P}_\alpha$, alors $P(o) \in Dom(\alpha, P)$. On notera aussi Dom_P (resp. Dom_α) la fonction obtenue en fixant P (resp. α). L'*héritage de propriétés* n'est pas seulement l'héritage des attributs ; c'est aussi l'héritage de leur domaine et une contrainte d'inclusion en cas de *redéfinition* de ces domaines : si $\alpha \prec \beta$ et $P \in \mathcal{P}_\beta$, alors $Dom_P(\alpha) \subseteq Dom_P(\beta)$.

À ce stade, on peut introduire la notion d'*intension* d'une classe, notée $Int(\alpha)$ et constituée par le couple $(\mathcal{P}_\alpha, Dom_\alpha)$. Les intensions sont munies

d'une relation d'ordre $\sqsupseteq$ similaire à l'inclusion comme suit :

$$Int(\alpha) \sqsupseteq Int(\beta) \stackrel{\text{def}}{\Longleftrightarrow} \mathcal{P}_\alpha \supseteq \mathcal{P}_\beta \ \& \ \forall P \in \mathcal{P}_\beta : Dom_P(\alpha) \subseteq Dom_P(\beta)$$

De façon analogue à la subsomption extensionnelle, la relation de *subsomption intensionnelle* est un *préordre* sur $\mathcal{X}$, noté $\trianglerighteq$, induit par l'inclusion des intensions : $\alpha \trianglerighteq \beta \stackrel{\text{def}}{\Longleftrightarrow} Int(\alpha) \sqsupseteq Int(\beta)$.

Les opérations ensemblistes usuelles s'étendent aux intensions en introduisant les opérateurs $\sqcap$ et $\sqcup$: $Int(\alpha) \sqcup Int(\beta) \stackrel{\text{def}}{=} (\mathcal{P}_\alpha \cup \mathcal{P}_\beta, Dom_{\alpha\cap\beta})$, avec

$$Dom_{\alpha\cap\beta}(P) = \begin{cases} Dom_\alpha(P) \cap Dom_\beta(P) & \text{si } P \in \mathcal{P}_\alpha \cap \mathcal{P}_\beta \\ Dom_\alpha(P) & \text{si } P \in \mathcal{P}_\alpha - \mathcal{P}_\beta \\ Dom_\beta(P) & \text{si } P \in \mathcal{P}_\beta - \mathcal{P}_\alpha \end{cases}$$

Dans cette définition, si jamais l'un des $Dom_\alpha(P) \cap Dom_\beta(P)$ est vide (si P n'a aucune valeur admissible), on pose alors que $Int(\alpha) \sqcup Int(\beta) = (\mathcal{P}, \emptyset) = Int(\bot)$. $\sqcap$ se définit de manière duale.

À ce qui précède, il faut ajouter une *composante extensionnelle*, qui est constituée par la description de toutes les entités individuelles, instances *effectives* des classes introduites dans la composante intensionnelle. La composante extensionnelle constitue une *description du monde* notée $\mathcal{W}$. C'est une réalisation particulière de l'interprétation extensionnelle du SC qui doit donc respecter toutes les contraintes formulées dans l'exposé de cette interprétation. La description du monde peut être vue comme un ensemble fini d'assertions, analogue à une base de données relationnelle, ces assertions pouvant être de deux sortes :

- $(\alpha\ o) \in \mathcal{X} \times \Omega$ affirme l'existence de l'*objet* $o \in Ext(\alpha)$;
- $(P\ o\ o') \in \mathcal{P} \times \Omega \times \Omega$ associe à l'attribut P de l'objet o la valeur o'.

2.3 Les objectifs d'un système classificatoire

La construction de la hiérarchie elle-même mise à part, un système classificatoire peut être utilisé, au niveau extensionnel, de diverses façons :

l'héritage permet de connaître *déductivement* les attributs d'un objet — pas leurs valeurs, au moins dans un premier temps, mais leur présence et leur « intension » —, étant donné ses classes ;

la classification permet de reconnaître *inductivement* un objet comme instance d'une classe, compte-tenu de ses attributs connus ; des assertions $\{(\alpha\ o)\ (P\ o\ o')\}$, on peut déduire que o' a pour type celui qui est associé à P dans α, mais on peut aussi induire une classe β, éventuellement sous-classe de α, telle que o' soit dans l'extension de β et ait pour type celui qui est associé à P dans β.

le filtrage détermine les instances d'une classe ou, plus généralement, les objets qui vérifient certaines propriétés qui peuvent s'exprimer par l'intension d'une classe : dans les logiques de descriptions, comme dans FROME [Dek 93], toute requête est considérée comme une classe dont il ne reste plus qu'à collecter les instances, une fois qu'elle a été correctement insérée dans la hiérarchie.

la construction d'instance consiste à valuer les attributs d'un objet, éventuellement en en créant d'autres qu'il faut récursivement construire et classer [Gir 95].

Mais la distinction de ces différents problèmes est artificielle : il s'agit d'abord d'un problème uniforme de déduction. Ce n'est qu'ensuite que l'on peut introduire des spécificités comme une classification ou un filtrage incertains.

2.4 Sémantique et déduction dans les SC : éléments

La *théorie des modèles* offre un cadre général pour établir une sémantique formelle ensembliste : le but est en particulier de définir la notion de conséquence *valide*, c'est-à-dire valable pour tous les *modèles*, d'un ensemble de descriptions. Par ailleurs, l'étude de la déduction dans les SC (en fait, essentiellement dans les logiques de descriptions) s'appuie, pour les approches les plus récentes, sur des techniques relevant de la théorie de la déduction.

De façon abstraite, la déduction dans un SC va consister à définir un opérateur de dérivation $\vdash$, analogue méta-syntaxique de $\models$, mais opérant au plan syntaxique et non plus sémantique. Cet opérateur peut être défini aussi bien implicitement — au travers d'un algorithme [Neb 90; Bor 94], de systèmes de contraintes [SS 91; Buc 93; Don 94] ou de recherche systématique d'un modèle fini [Par 96] —, qu'explicitement, par des règles d'inférence [Bor 92].

Les rapports entre déduction et sémantique s'appréhendent classiquement en termes de *correction* — $\mathcal{H};\mathcal{W} \vdash d \Longrightarrow \mathcal{H};\mathcal{W} \models d$ — et de *complétude* (sémantique) — $\mathcal{H};\mathcal{W} \models d \Longrightarrow \mathcal{H};\mathcal{W} \vdash d$.

La littérature sur les logiques de descriptions abonde en résultats portant sur la décidabilité ou semi-décidabilité de la subsomption dans ces systèmes [PS 89], ou sur sa complexité lorsqu'elle est décidable, ce qui amène à envisager un compromis *expressivité-complétude-complexité* [Lev 87; Att 91]. Deux conséquences générales peuvent être alors tirées de ces études :

- un même système classificatoire peut présenter une certaine variabilité de la déduction, ce qui fait perdre toute unicité aux notions dont la définition est basée sur la déduction ;
- une sémantique opérationnelle peut être nécessaire pour expliciter ce que calcule réellement le SC lorsqu'il n'est pas complet : c'est l'objectif affiché des règles d'inférence de [Bor 92].

Les paragraphes précédents ne donnent qu'un bref aperçu des systèmes classificatoires, dont l'étude est en cours de développement. Le lecteur plus intéressé pourra consulter [Duc 96a] et [Duc 96b] pour une vue plus précise et plus complète sur les travaux concernant les systèmes classificatoires

2.5 La révision dans une RCPO

La construction d'une base de connaissances est un processus qui n'est pas ponctuel et qui peut nécessiter le travail de plusieurs personnes, ce qui peut conduire à l'apparition d'*inconsistances* après l'ajout de nouvelles connaissances dans la base, lorsqu'elles sont contradictoires avec les connaissances déjà présentes dans la base. La problématique de la *révision* consiste à trouver alors la ou les modifications à apporter à la base initiale afin qu'elle puisse accueillir les nouvelles connaissances.

Le travail présenté dans [Cra 96] a pour thème la révision dans le cadre des systèmes de RCPO. L'objectif est de proposer l'ensemble des modifications possibles d'une base de connaissances afin que celle-ci soit *consistante* avec les dernières modifications qui lui ont été apportées. En effet, trouver la meilleure base révisée *a priori* est difficile à envisager, car la connaissance nécessaire pour décider quelles sont les connaissances qui sont les plus importantes et celles qui peuvent être supprimées n'est pas forcément disponible. Cependant, parmi l'ensemble des bases révisées possibles, seules les bases qui modifient de façon minimale la base initiale sont intéressantes, c'est-à-dire celles qui permettent de préserver le maximum de connaissances déductibles. La révision a été étudiée longuement dans le domaine de la logique [Alc 85] [Neb 92], et ces travaux peuvent être appliqués aux objets en traduisant les représentations par objets dans un formalisme logique. Mais, l'opération de révision reste problématique, car deux types de problèmes se posent lors de sa mise en œuvre : des problèmes de *complexité* (en particulier pour décider la consistance) et de *complétude* (les critères définis syntaxiquement ne correspondent pas à ceux définis sémantiquement). Et surtout, la traduction d'une représentation par objets vers un formalisme logique entraîne la perte de la structure (hiérarchique) d'une RCPO, structure qui, grâce aux entités présentes (classes, objets, attributs) et à des règles d'inférence adaptées, permet d'orienter la recherche des bases révisées.

Le système de RCPO sur lequel porte cette étude de la révision est une version simplifiée de Tropes [Tro 95]. Les entités de base du domaine à modéliser sont représentées par des objets (instances), qui possèdent des attributs ayant une valeur relevant d'un type prédéfini ou bien d'une classe (la valeur est alors une instance). Les objets sont regroupés dans des *classes*, auxquelles sont associés des ensembles de contraintes portant sur les valeurs des attributs. Ces contraintes sont des conditions *nécessaires* (et non suffisantes) d'appartenance d'un objet dans la classe (sémantique descriptive). De plus, les classes sont organisées en une hiérarchie par la relation de *spécialisation* ; tous les objets rattachés à une classe appartiennent également à toutes ses super-classes. Le mécanisme d'héritage sur les contraintes associées aux classes opère comme

suit : les contraintes d'une classe sont héritées par toutes ses sous-classes.

Une base de connaissances dans un système de RCPO est alors composée d'*assertions* ayant quatre formes possibles : l'appartenance d'un objet à une classe, la valeur d'un attribut d'un objet, la relation de spécialisation entre deux classes et les restrictions attachées à un attribut dans une classe.

Dans une telle base, les inconsistances pouvant apparaître sont de trois types :

- Deux valeurs différentes sont définies pour un attribut d'un objet particulier.
- Une valeur définie pour un attribut d'un objet ne satisfait pas les contraintes associées à une des classes d'appartenance de l'objet.
- Un objet est rattaché à une classe, et aucune valeur n'est possible pour l'un des attributs de l'objet, vu les contraintes associées à la classe.

Pour résoudre les problèmes d'inconsistance, il faut supprimer des connaissances de la base initiale. Les bases révisées sont donc des bases qui sont contenues dans la base initiale, telles que la dernière assertion ajoutée est consistante dans la base révisée. Comme il est souhaitable de conserver le maximum de connaissances présentes dans la base initiale, seules les bases révisées vérifiant la propriété suivante sont minimales : il n'est pas possible de trouver une base contenant plus de connaissances provenant de la base initiale qui soit également consistante avec la dernière assertion. L'ordre reposant sur la relation de spécialisation entre classes permet d'obtenir ces bases révisées minimales.

Afin de pouvoir établir formellement la correction du mécanisme de révision, le langage de RCPO étudié a été doté d'une sémantique et d'un système déductif correct et complet [Cra 96]. Les critères de minimalités syntaxique et sémantique d'une base révisée ont été définis. De plus, les postulats donnés dans [Alc 85] sont satisfaits. Au delà du résultat formel, les notions introduites, et les expérimentations en cours, permettent d'envisager des techniques de mise en œuvre efficace de la révision dans un système de RCPO.

3 Classification et conception de hiérarchies

3.1 La classification

Construire et raisonner

Le processus de classification opère sur une hiérarchie $\mathcal{H}$ et cherche à mettre en évidence les dépendances implicites qui existent entre les objets dans $\mathcal{H}$, les dépendances classes – classes (« la classification des espèces »), et les dépendances classes – instances (« classification des individus »). Ainsi, le processus de classification présente deux aspects qui nous intéressent dans la suite [2]. Tout

2. Sur les multiples aspects de la classification, voir par exemple [Mar 96].

d'abord l'aspect *constructif* se caractérise par l'insertion d'un nouvel objet dans la hiérarchie $\mathcal{H}$, en accord avec les informations présentes dans $\mathcal{H}$: le processus de classification peut donc servir de guide lors de la construction d'une hiérarchie, à l'image d'un processus de *classification* — ou *catégorisation* — *conceptuelle*. La classification conceptuelle consiste à regrouper des objets en *classes* qui matérialisent des *concepts* du domaine étudié. Les objets individuels sont discriminés sur la base de propriétés qu'ils partagent, mais aussi sur la base des connaissances disponibles sur le domaine [Nap 94a].

Le second aspect de la classification est l'aspect *déductif*, qui est à la base du *raisonnement par classification*. Le processus de classification s'appréhende comme une procédure de déduction qui exploite les propriétés d'un univers où les connaissances sont représentées par des classes organisées en une hiérarchie [Att 91]. Les hiérarchies d'héritage, les hiérarchies de concepts des logiques de descriptions [Neb 90], et les hiérarchies de composants dans la représentation d'objets composites [Nap 94b], sont des exemples de tels univers.

Dans ce qui suit, ces deux aspects de la classification sont abordés, que ce soit pour concevoir des hiérarchies de classes (dans le cas d'un langage à objets ou d'un système de RCPO, ainsi que dans le modèle des graphes conceptuels pour concevoir les taxinomies de types), ou encore pour mener à bien des raisonnements portant sur des objets organisés en une hiérarchie (RCPO ou graphes conceptuels). Avant de détailler le raisonnement par classification proprement dit, nous présentons quelques éléments de l'inférence par héritage, qui est couramment utilisée dans les systèmes RCPO.

L'inférence par héritage

Le rôle du mécanisme d'héritage est de contrôler le partage de propriétés et la réutilisation de code, en s'appuyant sur un algorithme de linéarisation qui établit l'ordre des ascendants d'une classe dans $\mathcal{H}$, ordre dans lequel sont héritées les propriétés partagées de la classe [Duc 95]. En particulier, bien programmer nécessite une bonne connaissance de $\mathcal{H}$ et du mécanisme d'héritage.

Si une classe `A` spécialise une classe `B`, alors les ensembles de propriétés — attributs et méthodes — $\mathtt{P_A}$ de `A` et $\mathtt{P_B}$ de `B` vérifient : $\mathtt{P_B} \subseteq \mathtt{P_A}$; l'*intension* de `A` contient l'intension de `B`. L'inférence par héritage repose essentiellement sur la *transitivité* de la relation de spécialisation : si Socrate est un philosophe, que les philosophes sont des hommes, et que les hommes sont mortels, alors Socrate hérite la propriété d'être mortel ; bien que la spécialisation porte sur les classes, l'héritage concerne *in fine* les instances.

Lorsque l'héritage est *standard*, les domaines spécialisants sont inclus dans les domaines spécialisés (domaines emboîtés). Un domaine de valeurs est alors considéré comme une disjonction de valeurs. Par exemple, si le domaine d'un attribut `att` est `[1,2]` dans la classe `C` et `[2,3]` dans la classe `D`, ce domaine est égal à `[2]` ($[1,2] \cap [2,3]$) dans la classe `E`, sous-classe de `C` et `D` : $[1] \longrightarrow [1,2]$ et $[2] \longrightarrow [1,2]$, où la flèche $\longrightarrow$ correspond à la relation de spécialisation et où $[1] \longrightarrow [1,2]$ s'interprète comme `[1]` *implique* `[1 ou 2]` ou `[1]` *spécialise*

`[1,2]`.

Il est possible de faire une analogie entre l'interprétation qui vient d'être donnée pour les domaines et le constructeur `one-of` des logiques de descriptions, qui définit un domaine en extension [Bra 91] : l'expression `one-of(a,b,c)` *subsume* l'expression `one-of(a,b)`, au sens où l'ensemble `{a,b,c}` contient l'ensemble `{a,b}` et il est plus général que ce dernier : `{a,b,c}` s'interprète comme `{a ou b ou c}`.

L'héritage peut aussi être *cumulatif* : la valeur d'un attribut `att` dans une classe `C` donnée peut être définie à partir de la collection des valeurs de `att` dans les super-classes de `C`. Dans le cas cumulatif, les domaines spécialisants contiennent les domaines spécialisés, et un domaine de valeurs est considéré comme une conjonction de valeurs. Par exemple : $[1,2] \cup [2,3] \longrightarrow [1,2]$ et $[1,2] \cup [2,3] \longrightarrow [2,3]$, et cette fois, l'interprétation est `[1 et 2 et 3]` *implique* `[1 et 2]` ou `[1 et 2 et 3]` *spécialise* `[1,2]`.

Ce qui vient d'être présenté à propos des domaines de valeurs suppose que le masquage des types, domaines et intervalles, est monotone. Si ce n'était pas le cas, l'analyse ci-dessus serait incorrecte et aucune des « implications » ne pourrait être garantie.

Du point de vue des inférences, l'héritage ne permet de déduire ou de combiner des valeurs que dans des cas relativement simples. Ainsi, si l'attribut `att` a au plus `2` valeurs élémentaires dans la classe `C` et au moins `2` valeurs élémentaires dans la classe `D`, alors aucun mécanisme ne permet de déduire que `att` doit avoir exactement `2` valeurs dans la sous-classe `E` de `C` et `D`. De même, une personne dont tous les enfants sont musiciens, et dont tous les enfants musiciens sont peintres, doit forcément être une personne dont tous les enfants sont peintres [PS 90]. Une telle inférence est de l'ordre du *modus ponens* : `musicien`, $\texttt{musicien} \longrightarrow \texttt{peintre} \vdash \texttt{peintre}$. Le mécanisme d'héritage n'est pas en mesure d'effectuer une telle inférence, mais ce n'est pas là son rôle. En particulier, les mécanismes de filtrage et de classification complètent le mécanisme d'héritage du point de vue des inférences.

Ainsi, le filtrage permet de retrouver des objets s'appariant avec un *filtre*, qui n'est autre qu'une classe virtuelle définie par un ensemble d'attributs, les *contraintes de filtrage*, qui doivent être vérifiées par les objets recherchés. Le filtrage et la classification sont complémentaires, et les principes du raisonnement par classification sont explicités ci-dessous.

Le raisonnement par classification

Le raisonnement par classification s'appuie sur un cycle *réification – classification – exploitation* permettant de tirer profit d'une hiérarchie $\mathcal{H}$ en vue de résoudre des problèmes, mais aussi de fournir une aide à la conception de hiérarchies. Ainsi, la conception d'un plan de synthèse de molécule en chimie organique, dont certains éléments sont détaillés dans le paragraphe 4.1, montre un bon exemple du double rôle du mécanisme de classification : savoir construire une structure moléculaire `M` revient à résoudre un problème de synthèse orga-

nique et consiste à classifier M dans une hiérarchie de structures moléculaires, en vue de reconnaître les caractéristiques de M. Le même mécanisme de classification est utilisé pour construire dynamiquement la hiérarchie de structures moléculaires, qui est l'univers de résolution du problème de synthèse.

Le raisonnement par classification s'appuie sur un *cycle de classification*, qui consiste à établir la position d'un objet o, classe ou instance, dans la hiérarchie $\mathcal{H}$. La mise en œuvre du raisonnement par classification repose sur un *cycle* comprenant trois étapes principales :

- *Réification* : création du nouvel objet o en cours d'étude (classe ou instance).
- *Classification* : parcours de la hiérarchie $\mathcal{H}$ et mise en place de o dans $\mathcal{H}$, en accord avec l'ordre associé à $\mathcal{H}$. Cette opération comprend trois phases, la recherche des *ascendants les plus spécifiques* ou APS de o, la recherche des *descendants les plus généraux* ou DPG de o (cette recherche est inutile si o est une instance), et enfin l'insertion de o dans $\mathcal{H}$ [Baa 94].
- *Exploitation* : la mise en place de o dans $\mathcal{H}$ déclenche des opérations de mise à jour d'objets interdépendants et/ou la production de nouveaux objets, ce qui ramène le cycle à sa première étape. Le cycle continue tant que les informations disponibles permettent de produire de nouveaux objets à classifier et que le but associé au problème à résoudre n'est pas atteint.

Par extension, il est envisageable d'appliquer le cycle de classification sur $\mathcal{H}$ avec plusieurs relations d'ordre, correspondant à autant de *points de vue globaux*, qui peuvent chacun constituer une *dimension* dans l'ensemble des classes considéré, donnant ainsi naissance à un processus de *classification multi-dimensionnelle*.

3.2 Conception et re-conception de hiérarchies d'objets

Motivations

Les travaux concernant la production d'outils visant à automatiser certains aspects de l'élaboration des hiérarchies de classes se multiplient. C'est sans doute le signe que la nécessité de rationaliser leur construction et leur évolution apparaît de plus en plus clairement à la communauté objet. Les mêmes constats se retrouvent sous la plume de la plupart des auteurs de ces travaux :

- de nombreuses hiérarchies ont été construites et modifiées à la va-vite, et sans souci d'uniformité,
- il est très difficile de concevoir dans un premier jet une hiérarchie de classes satisfaisante : une succession de phases d'essai puis de retour sur la conception est nécessaire, et c'est d'ailleurs le propre des approches objets de permettre un prototypage facile,

- même lorsqu'un domaine est modélisé de façon satisfaisante, rien ne garantit que l'on a trouvé les classes possédant un degré de généralité permettant de les réutiliser,
- la taille de certaines hiérarchies les fait échapper à un contrôle critique de leur(s) concepteur(s),
- l'évolution des hiérarchies (ajout ou retrait de classes, de propriétés) doit être contrôlable, que ce soit pour l'utilisateur ou pour les programmes opérant sur la hiérarchie,
- certaines classes sont le produit de calcul (comme les *classes vues* dans les bases de données) et rien n'est plus naturel que de les gérer par d'autres calculs.

Du neuf avec du vieux

La problématique de la conception re-conception de hiérarchies d'objets n'est pas nouvelle en tant que telle, et elle s'apparente fortement à celle de la *classification conceptuelle* : entendre par là un mécanisme d'apprentissage symbolique qui cherche à former des concepts décrits en intension à partir de l'observation d'un domaine dont on souhaite produire une représentation. Le domaine est généralement perçu par l'intermédiaire d'objets (dépendant de classes) décrits par des attributs. Quelques points précisent les particularités de l'organisation des classes dans les langages à objets (et dans les systèmes de RCPO) :

- c'est plutôt en organisant des concepts, ou des classes, que l'on en découvre d'autres, même si certains auteurs déclarent partir d'*individus* qui sont décrits de la même manière que les classes (la distinction n'est pas nette puisque ces « individus » ou une classe qui les décrit apparaîtront finalement dans la hiérarchie [God 93]) ; de même, toujours dans le cadre de la factorisation, il est possible de rechercher des propriétés communes plutôt que des valeurs communes pour certaines propriétés,
- l'idée de *similarité* que l'on trouve dans certains travaux de classification conceptuelle disparaît dans le cas des langages à objets au profit de l'idée d'*exactitude* de la représentation : toutes les propriétés doivent être représentées dans les classes, et rien de plus.
- l'optique dans laquelle se fait l'organisation des classes est une optique de représentation dans le cas des langages à objets, alors que les systèmes classiques de regroupement conceptuel se préoccupent essentiellement de *prédiction* [Bou 96].

Ces quelques différences entre le problème tel qu'il se pose en classification conceptuelle et dans le cas de l'organisation de classes dans les langages à objets font que certaines techniques d'apprentissage pourraient être envisagées,

moyennant adaptation, pour construire des hiérarchies de classes, à condition de ne pas se limiter à la production d'un arbre comme c'est assez souvent le cas dans les méthodes de classification conceptuelle. Il semblerait que pour l'instant seules les techniques s'appuyant sur les *treillis de Galois* aient été utilisées (ou ré-inventées) par la communauté se préoccupant d'objets (et celle des bases de données par objets).

État des lieux et discussion

L'essentiel des techniques développées pour l'instant s'appuie sur des procédés de *refactorisation* des propriétés, et plus précisément cherchent à assurer une factorisation maximale des propriétés, ce qui ne garantit pas un résultat unique. Parmi les factorisations maximales possibles, l'une introduit un nombre minimum de classes, c'est celle qui s'appuie sur la construction du treillis de Galois de la relation binaire liant classes et propriétés. Il n'est donc pas fortuit que ce fameux treillis, ou plutôt un de ses sous-ordres caractéristiques, apparaisse implicitement ou explicitement en toile de fond de nombreux travaux.

Les propriétés sont le plus souvent décrites par un type [Dic 96] [Cap 95] ou par un objet composite ayant un statut de classe, et pour les méthodes, par une signature et/ou un code [Dic 96]. Un modèle plus général consiste à utiliser un ordre partiel sur les différentes occurrences d'une propriété [God 93; Dic 95].

Reste à préciser ce qui est réellement pris en compte dans la comparaison, dans la possibilité de surcharger et de masquer des propriétés et dans le calcul d'une généralisation de deux propriétés. Les modèles basés sur des treillis de concepts peuvent prendre en compte le cas le plus général où les ordres partiels sur les différentes occurrences d'une propriété permettent d'envisager des profondeurs de masquage non limitées. Cependant, pris au pied de la lettre, ils demandent que soient stockées (dans les classes ou dans la table de la relation binaire classe – propriété) toutes les occurrences de propriétés. Une alternative consiste à intégrer un calcul de la plus petite généralisation commune de deux propriétés (super-type de deux types, généralisation de deux signatures, généralisation de deux codes) [Dic 96].

Notons aussi que les graphes d'héritage sur lesquels s'appliquent ces travaux sont plus ou moins contraints (graphes sans circuit, treillis), et que les stratégies algorithmiques mises en œuvre pour les parcourir et les modifier sont assez variées : opérations locales de modification, algorithmes globaux et incrémentaux [Dic 94; Dic 95; Dic 96].

Doit-on espérer beaucoup de ces techniques? Les expériences déjà menées tendent à faire penser (ou à faire croire?) que l'on peut tirer beaucoup d'informations des résultats d'une (ré-)organisation automatique, mais il ne faut sans doute pas en attendre une hiérarchie immédiatement utilisable. Les difficultés sont, comme dans le cadre de l'apprentissage, de décrire assez finement les classes et de préciser ce que l'on attend d'une *bonne hiérarchie* ; il s'agit alors de s'entendre, par exemple, sur le choix du sous-typage ou du partage

d'implémentation, sur le poids à accorder à la refactorisation. Le champ est donc ouvert à de nombreux développements.

3.3 La classification dans les graphes conceptuels

Le modèle des graphes conceptuels est un modèle général de représentation des connaissances fondé sur la description de concepts et de liens entre ces concepts [Sow 84]. Ce modèle associe une approche graphique — la connaissance est modélisée par des graphes et les raisonnements sont des opérations sur ces graphes — à la logique du premier ordre, qui permet de doter le modèle d'une sémantique formelle (voir [Che 92] et [Mug 96] pour la théorie des graphes conceptuels).

Les connaissances exprimées dans le modèle des graphes conceptuels se structurent en deux niveaux principaux : un *niveau terminologique* composé d'une taxinomie de types de concepts et d'une « taxinomie » de types de relations, et un *niveau assertionnel* où sont décrits des faits par des graphes construits avec les éléments du niveau terminologique. Dans ce qui suit est présentée l'utilisation du raisonnement par classification à ces deux niveaux.

Construction des taxinomies

Les taxinomies des types permettent de représenter les relations de spécialisation qui existent entre les différents types. Les types de concepts modélisent des catégories d'entités, d'attributs, d'états ou d'événements et les types de relations représentent les diverses catégories de liens entre instances de catégories d'entités. Les deux ensembles de types sont disjoints et chacun d'eux est structuré par une relation de spécialisation.

Les types composant ces taxinomies peuvent être *atomiques* ou *définis* [3]. Les types atomiques sont de simples étiquettes qui permettent de désigner des catégories générales. La position de ces types dans la taxinomie est directement donnée par le concepteur de la taxinomie. Les types définis comprennent en plus de leur étiquette une *structure* représentant un ensemble de conditions nécessaires et suffisantes d'appartenance au type (il existe aussi des types *partiellement définis* pour lesquels seules des conditions nécessaires existent).

La structure associée à un type défini permet de le positionner de manière automatique, par classification, dans la taxinomie des types, selon le schéma général de classification présenté au paragraphe 3.1. Lors de la classification, la comparaison des types définis s'appuie sur l'opération de *projection*. Des propriétés particulières étant imposées aux taxinomies, comme par exemple une structure de treillis pour celle des types de concepts, il faut donc vérifier après chaque insertion la conservation de ces propriétés. Ainsi, un système d'aide à la construction du niveau terminologique du modèle des graphes conceptuels est proposé dans [Lec 94; Lec 95; Lec 96]. L'opération de projection correspond à

3. Il est possible de faire un parallèle avec les classes primitives et définies des systèmes classificatoires ou des logiques de descriptions.

un morphisme de graphes étiquetés préservant le type des sommets (un sommet peut avoir comme image un sommet ayant une étiquette plus spécifique ; il est possible de faire un parallèle avec la relation de co-subsomption utilisée dans le système RESYN, paragraphe 4.1). Pour tenir compte des classes d'équivalence de la relation de spécialisation (qui est un préordre), les projections injectives sont prises en compte ou bien la classification est réalisée sur l'ordre quotient.

Structuration d'une base de faits

L'opération de projection est à la base de tous les raisonnements effectués dans le modèle des graphes conceptuels (comme suggéré au paragraphe 2.3, c'est un problème uniforme de déduction). Ces raisonnements passent, pour la plupart, par une recherche de graphes spécialisant ou généralisant un graphe conceptuel donné. De fait, il semble naturel qu'un système de gestion de bases de graphes conceptuels maintienne une structure hiérarchique représentant cette relation de spécialisation. La construction et l'exploitation de cette structure est réalisée à l'aide des techniques de classification : l'adjonction d'un graphe est réalisée en recherchant sa position par classification ; de même la recherche des spécialisations ou des généralisations d'un graphe donné est faite par l'intermédiaire du calcul de ses ascendants et descendants dans les taxinomies concernées.

La nature des éléments traités ici conduit aux remarques suivantes :

- la relation de spécialisation est un préordre [Che 92], et la classification opère ainsi sur l'ordre quotient ;
- un graphe fait peut contenir plusieurs centaines de sommets ;
- une base de faits peut contenir un très grand nombre de faits (plusieurs milliers).

La plate-forme logicielle CoGITo [Hae 95], permettant le développement d'applications où l'unité de base est le graphe conceptuel, a été enrichie d'un module de gestion d'ensembles de graphes conceptuels stockés en mémoire secondaire, et d'un module d'interrogation [Gui 96b]. Ces deux modules s'appuient d'une part sur des algorithmes de classification (où sont utilisés des graphes conceptuels spéciaux jouant le rôle d'*index* [Gui 95]), et d'autre part sur des recherches de graphes par leur structure [Gui 96a]. L'adjonction de ces deux modules représente une avancée réelle pour les outils manipulant des graphes conceptuels.

4 Applications

4.1 Le système RESYN

RESYN est un système d'aide à la conception de plans de synthèse en chimie organique [Vis 92; Vis 95; Vis 96]. Dans ce qui suit, l'appariement de structures

moléculaires, sur lequel s'appuie le raisonnement par classification dans RESYN, est détaillé.

La connaissance manipulée dans RESYN est représentée sous la forme de structures moléculaires, appelées *graphes moléculaires*, qui sont décrites par des graphes étiquetés où sommets et arêtes sont respectivement associés aux atomes et aux liaisons de la structure moléculaire. Le système RESYN a pour but d'aider un chimiste à construire un plan de synthèse pour une molécule donnée, appelée *molécule cible* (par exemple un nouveau médicament). Suivant un raisonnement *analytique*, on cherche à appliquer au graphe moléculaire décrivant la molécule cible une série de *transformations*, jusqu'à obtenir un ensemble de réactifs disponibles (un exemple de résolution de problème en chaînage arrière, cf. figure 1).

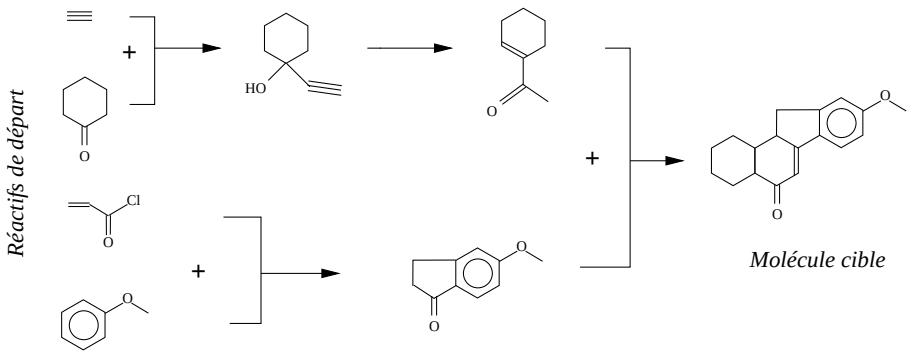

FIG. 1: *Exemple de plan de synthèse décrivant les enchaînements de réactions permettant d'obtenir la molécule cible à partir d'un ensemble de réactifs.*

Une *transformation* modifie un ensemble de sommets et d'arêtes du graphe moléculaire. Le graphe induit par cet ensemble constitue l'*objectif* de la transformation (cf. figure 2). L'application d'une transformation obéit à des contraintes d'ordre chimique qui définissent les environnements favorables ou défavorables à l'utilisation de la transformation. Nous décrivons ces environnements sous la forme de graphes appelés *contextes*, qui étendent l'objectif de la transformation aux sommets et arêtes conditionnant son application. L'ensemble des contextes, qui peut être très vaste, est représenté sous la forme d'un réseau organisé suivant la relation de *co-subsomption* (qui est explicitée ci-dessous). La recherche d'un contexte favorable à l'application d'une transformation s'apparente alors à une classification du graphe moléculaire dans la hiérarchie de graphes définie par ce réseau (cf. figure 2).

Un graphe moléculaire `G1` *co-subsume* un graphe moléculaire `G2` s'il existe un sous-graphe $\mathtt{G}_2'$ de `G2` isomorphe à `G1`, où l'isomorphisme préserve le type des atomes et des liaisons, c'est-à-dire que tout atome, respectivement toute liaison, dans `G1` a un type plus général que l'atome, respectivement la liaison, correspondant dans $\mathtt{G}_2'$ [Nap 93] [Vis 95].

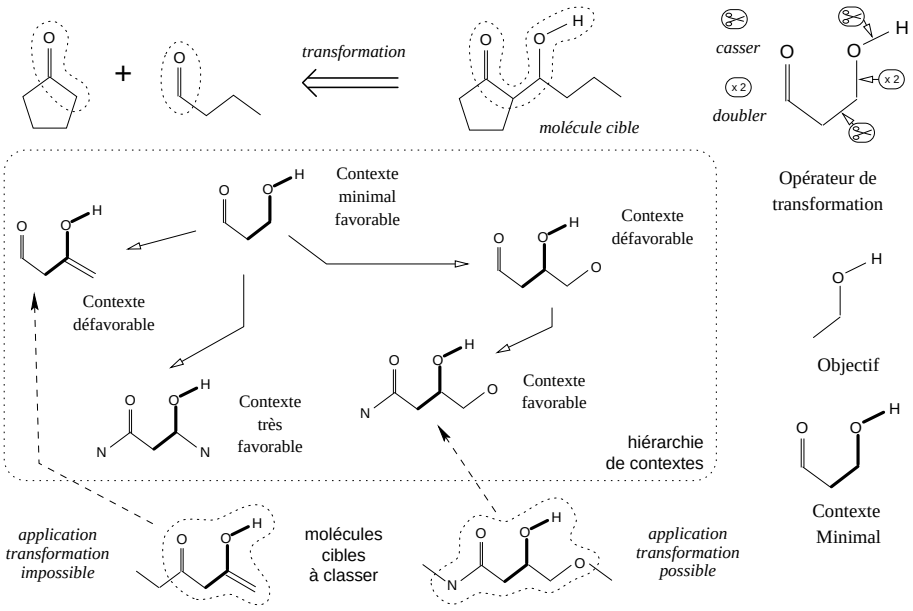

FIG. 2: *Transformation, objectif et hiérarchie de contextes.*

4.2 Subsomption d'appariements

Un simple raisonnement par classification, basé sur la relation de co-subsomption, n'est cependant pas satisfaisant dans le cadre de RESYN. En effet, un opérateur de transformation peut souvent être appliqué dans plusieurs endroits de la molécule. Suivant la partie concernée, cette application pourra être possible ou impossible, du fait de la présence à cet endroit d'un contexte favorable ou défavorable.

Lorsqu'on cherche à appliquer une transformation à un endroit précis de la molécule, on dispose d'un appariement f_O du graphe G_O décrivant l'objectif de la transformation vers le graphe moléculaire G_{mol}. Dans l'exemple de la figure 3, la transformation associée à l'objectif G_O peut être appliquée à deux endroits de la molécule G_{mol}, qui correspondent aux appariements f_O et f'_O.

Pour savoir si la transformation est applicable, on étudie l'ensemble des contextes qui sont à la fois plus spécifiques que l'objectif G_O et plus généraux que le graphe G_{mol}. Sur la figure 3, l'appariement f_O de l'objectif G_O est « recouvert » par l'appariement f_i du contexte *favorable* G_i. On en déduit que la transformation est applicable à cet endroit de la molécule. Par contre, le second appariement f'_O de l'objectif est « recouvert » non seulement par un appariement f'_i de G_i mais aussi par l'appariement f_j du contexte *défavorable* G_j. Il en résulte que la transformation n'est pas applicable dans la partie de la molécule désignée par l'appariement f'_O. Le problème de l'application d'une transformation ne peut donc pas se résoudre par un simple raisonnement par classification

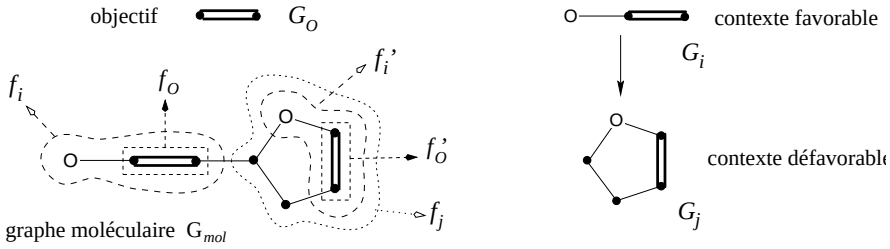

FIG. 3: *Appariements d'un objectif et de contextes dans un graphe moléculaire.*

s'appuyant sur la relation de co-subsomption définie entre graphes. Il ne faut pas rechercher tous les contextes qui co-subsument G_{mol} mais uniquement ceux qui peuvent s'apparier dans G_{mol}, en intégrant l'appariement f_O de l'objectif. Il paraît ainsi nécessaire d'introduire une nouvelle relation de subsomption, qui est définie non pas sur l'ensemble des contextes mais plutôt sur celui de leurs appariements dans le graphe moléculaire.

Ainsi, l'appariement f_i du graphe G_i dans G_{mol} *subsume* (par définition) l'appariement f_j de G_j dans G_{mol}, si et seulement si l'image de G_i par f_i, notée $f_i(G_i)$, est un sous-graphe partiel de $f_j(G_j)$ (c.-à-d. $f_i(G_i)$ est «contenu» dans $f_j(G_j)$).

Le problème consiste alors à rechercher les appariements de contextes dans le graphe G_{mol} qui sont subsumés par l'appariement f_O de l'objectif. Pour optimiser cette recherche, il faut se limiter aux contextes qui sont co-subsumés par le graphe G_O de l'objectif et qui co-subsument le graphe moléculaire G_{mol}. La détermination de ces contextes nous ramène au raisonnement par classification initial, qui consiste à « descendre » dans la hiérarchie de contextes à partir du contexte minimal co-subsumé par l'objectif (cf. figure 2). Mais les seuls contextes pris en compte pour l'application de la transformation seront ceux qui admettent au moins un appariement dans G_{mol} qui est subsumé par f_O.

Cette méthode, qui suppose de chercher tous les appariements de chaque contexte co-subsumant G_{mol}, n'est cependant pas satisfaisante. En effet, si le graphe G_{mol} étudié est relativement grand, un contexte donné peut s'apparier dans de nombreuses parties de G_{mol}. Or parmi tous ces appariements, seuls ont de l'intérêt ceux qui sont subsumés par f_O. Il est donc intéressant de focaliser la recherche des appariements de contextes «autour» du sous-graphe partiel de G_{mol} défini par $f_O(G_O)$. Cette idée est à la base de la notion d'*appariements dirigés* qui a été élaborée dans le cadre de REsyn [Vis 95; Vis 96].

Cette méthode repose sur la remarque suivante : pour que l'appariement f_i d'un contexte G_i soit subsumé par f_O, il faut qu'une partie de G_i soit associée au sous-graphe partiel de G_{mol} défini par $f_O(G_O)$. En effet, puisque G_O subsume G_i, tout appariement de G_i dans G_{mol} doit «contenir» un appariement de G_O dans G_{mol}. Le principe des « appariements dirigés » consiste alors à utiliser f_O pour construire f_i. En effet, si on détermine les relations structurelles

d'inclusion qui existent entre les graphes G_O et G_i, on peut considérer f_O comme la «solution partielle» d'un appariement f_i de G_i dans G_{mol}.

De la même manière, considérons deux contextes G_i et G_j tels que G_i co-subsume G_j et supposons que l'on connaisse les appariements (valides) de G_i dans le graphe G_{mol} (figure 4). La méthode des «appariements dirigés» va nous permettre de calculer les appariements de G_j dans G_{mol} à partir de ceux de G_i dans G_{mol}.

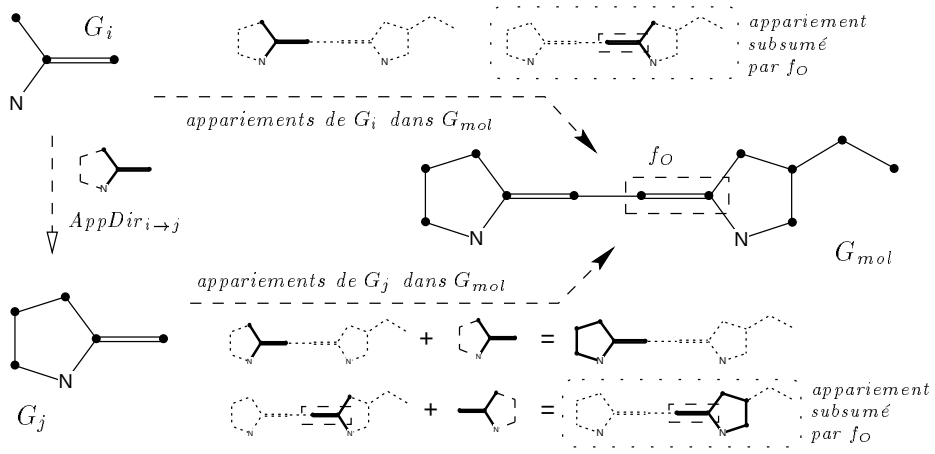

FIG. 4: *Principe des appariements dirigés.*

Pour ce faire, on dispose de l'ensemble, noté $AppDir_{i\to j}$, des *appariements dirigés de G_i dans G_j*. Cet ensemble contient tous les appariements de G_i dans G_j qui ne sont pas équivalents à une symétrie de G_j près[4]. L'ensemble $AppDir_{i\to j}$ peut être calculé à l'avance et stocké dans la hiérarchie de graphes au niveau du lien existant entre G_i et G_j. Le calcul des appariements de G_j dans G_{mol} s'effectue alors de la manière suivante : pour chaque appariement f_i de G_i dans G_{mol}, on construit un ou plusieurs appariements f_j de G_j dans G_{mol}, en utilisant les appariements stockés dans l'ensemble $AppDir_{i\to j}$ pour étendre f_i au graphe G_j. Cette méthode est illustrée par la figure 4. Pour chacun des deux appariements de G_i dans G_{mol}, on obtient un appariement de G_j dans G_{mol} en utilisant l'appariement de G_i dans G_j qui est stocké dans l'ensemble $AppDir_{i\to j}$. Il est important de noter ici que si parmi les appariements de G_i dans G_{mol}, un seul f_i est subsumé par l'appariement f_O de l'objectif, on ne construira qu'un seul appariement f_j de G_j dans G_{mol} à partir de f_i.

D'un point de vue algorithmique, la méthode des appariements dirigés permet de focaliser la recherche des appariements des contextes en se limitant à

4. On peut montrer que la suppression des appariements symétriques relativement à G_j ne remet pas en cause la validité de la méthode, tout en réduisant la taille des ensembles d'appariements dirigés. Il faut également préciser que d'autres contraintes interviennent dans la définition de ces ensembles.

la partie du graphe cible G_{mol} entourant l'appariement f_O initial. Lorsque le graphe cible est relativement grand, cette méthode permet d'éviter des calculs d'appariements inutiles.

Par ailleurs, la notion d'appariements dirigés offre un grand intérêt pour la représentation de certaines connaissances liées au domaine d'application. Dans RESYN, les graphes de la hiérarchie permettent de définir des environnements favorables ou défavorables à l'application d'une transformation caractérisée par son graphe objectif. Comme le montre la figure 5, certains contextes intègrent plusieurs appariements de l'objectif qui sont favorables ou défavorables, ou pour lesquels on ne dispose d'aucune information. La transmission de ce dernier type d'appariements est alors inutile et alourdit le processus de classification. La notion d'appariements dirigés permet ainsi de ne considérer que les appariements qui apportent une réelle information. Ce mécanisme correspond au raisonnement de l'expert qui élimine directement les appariements entre structures moléculaires n'ayant aucun sens chimique.

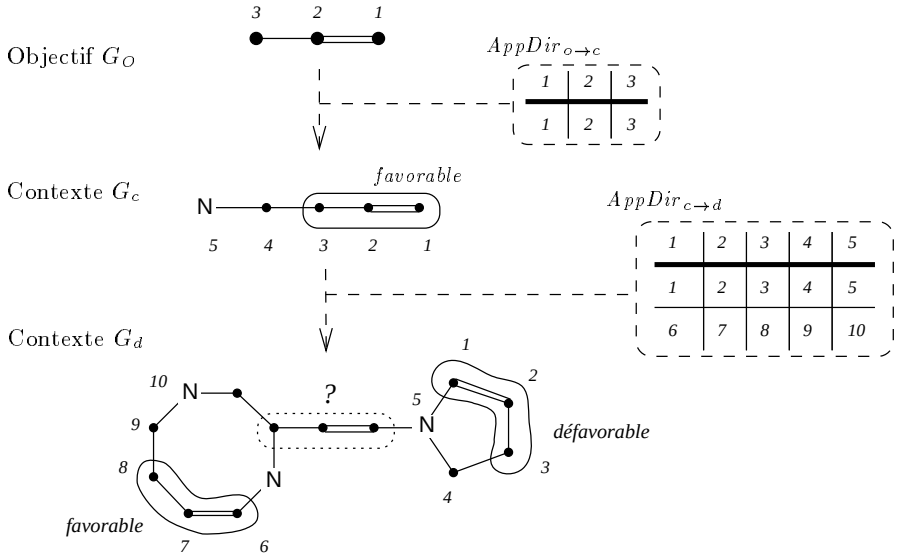

FIG. 5: *Les* appariements dirigés *permettent de ne pas transmettre les appariements pour lesquels on ne dispose d'aucune information.*

Le système RESYN est développé dans le cadre du GDR CNRS TICCO (Traitement Informatique de la Connaissance en Chimie Organique). Ce GDR interdisciplinaire, dans lequel informaticiens et chimistes collaborent, regroupe une dizaine de laboratoires de recherche du secteur public et différentes sociétés industrielles : FRAMENTEC COGNITECH, ROUSSEL UCLAF, SANOFI CHIMIE, SEMA GROUP et SERVIER. Les développements pratiques et théoriques effectués autour de RESYN montrent bien comment une collaboration interdisciplinaire peut donner naissance à un système informatique original et de qualité,

mais aussi comment la pratique peut se nourrir et faire avancer la théorie, dans l'univers des systèmes de RCPO.

4.3 Les besoins pour la conception de systèmes de RPCO

Plusieurs besoins en matière de représentation et de raisonnement dans les systèmes de RCPO peuvent être répertoriés [Cho 96]. La conception d'un système d'intelligence artificielle dans un environnement à base d'objets nécessite de construire un modèle du domaine et de l'implanter sous la forme d'une hiérarchie de classes. Les classes sont munies d'attributs qui peuvent dénoter des relations — liens entre classes — et les classes peuvent aussi avoir le statut particulier d'objet composite. De plus, une classe peut avoir un comportement spécifique qu'il faut pouvoir représenter, comportement qui peut dépendre d'une ou de plusieurs autres classes.

Un système d'intelligence artificielle repose sur une base de connaissances et met en œuvre un mécanisme de raisonnement. Le raisonnement peut revêtir plusieurs formes dans le cadre des systèmes de RCPO. Il peut reposer sur la circulation d'informations le long de la hiérarchie (héritage), mais aussi sur diverses fonctions associées aux classes. Dans ce cadre, le raisonnement par classification apparaît comme un mécanisme naturel pour gérer et contrôler les informations présentes dans une hiérarchie et compléter les mécanismes de filtrage, d'héritage, et le raisonnement procédural. L'évolution et la mise à jour d'une base de connaissances dans un système de RCPO nécessite également de s'intéresser aux problèmes de révision.

Un système de RCPO peut alors se définir comme un environnement de représentation (et éventuellement de programmation), qui permet de résoudre des problèmes posés sur un domaine donné, et qui possède un certain nombre des caractéristiques suivantes :

- Les classes sont munies de propriétés structurelles et comportementales — attributs et méthodes — et sont instanciables. Si le système est réflexif, toute entité peut alors être manipulée comme une instance, y compris les relations et les méthodes.

- Les classes sont organisées en une hiérarchie d'héritage par l'intermédiaire de la relation de spécialisation, qui reflète le niveau d'abstraction des classes. La gestion de la hiérarchie et le partage de propriétés sont à la charge du mécanisme d'héritage. Les classes s'organisent aussi autour de relations pour lesquelles il doit être possible de définir un partage de propriétés spécifique.

- Les opérations principales liées au raisonnement sont l'inférence par héritage, le filtrage et la classification (opérations auxquelles il faut ajouter la *satisfaction de contraintes*, non abordée ici). La gestion de la cohérence et de l'évolution d'une base d'objets nécessite de mettre en place un système de révision des connaissances.

- Des assertions permettent de décrire des faits dans lesquels sont engagées les classes et les instances. Elles sont quantifiées ou non, et se ramènent à des instanciations de classes ou de relations entre classes, ou encore, lorsque l'instanciation s'avère insuffisante, à des envois de messages ou des expressions fonctionnelles.

- Lorsqu'elle est possible (voire souhaitée), la programmation se fait sur la base de l'envoi de messages et de l'accès aux attributs. Les problèmes de *sémantique* revêtent alors une très grande importance. En particulier, même en cas de programmation disciplinée, aucun mécanisme ne permet de contrôler les effets de bords des fonctions et procédures employées dans le code.

La liste de caractéristiques qui vient d'être donnée n'est qu'une première approximation. En particulier, elle se place sur le plan des fonctionnalités disponibles ou à prévoir. Toutefois, il faut encore continuer à travailler sur le problème de la sémantique des systèmes de RCPO, et délimiter soigneusement ce qui relève de la représentation (avec sa sémantique) et ce qui relève de la programmation (avec sa sémantique), pour concevoir un système de RCPO où la représentation, le raisonnement et la programmation reposent sur une sémantique adéquate.

5 Conclusion

Certains des thèmes de recherches étudiés dans le groupe «Objets et classification» du PRC IA ont été présentés dans cet article. La tendance actuelle va plutôt vers la formalisation des études antérieures, et a permis de faire émerger les études sur les systèmes classificatoires et sur la révision dans les bases d'objets. La classification, sous ses deux formes standards, conception de hiérarchie et identification, reste un mode de raisonnement privilégié dans les systèmes de RCPO. Par ailleurs, cet article a également voulu montrer l'importance des applications pour illustrer les concepts théoriques associés aux systèmes de RCPO, mais aussi pour initier des développements de cette théorie. Un autre point important concerne la méthodologie de conception de systèmes de RCPO, sur lesquels de nombreux travaux restent à faire.

Les perspectives de recherches associés aux thèmes étudiés par les membres de groupe «Objets et classification» du PRC IA, en dehors des développements déjà mentionnés sur les points précédents, sont nombreuses et diverses. Elles peuvent s'orienter vers l'intégration des contraintes dans les systèmes de RCPO, l'analyse des liens existant entre systèmes classificatoires, logiques de descriptions et graphes conceptuels, les passerelles vers les bases de données (dans le contexte de la fouille de données par exemple), l'adéquation des systèmes de RCPO à des applications industrielles importantes ou dans la réalisation de systèmes d'intelligence artificielle distribuée.

Références

[Alc 85] C. Alchourròn, E. Gärdenfors, and P. Makinson. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. *Journal of Symbolic Logic*, 50(2), 1985.

[Att 91] G. Attardi. An Analysis of Taxonomic Reasoning. In M. Lenzerini, D. Nardi, and M. Simi, editors, *Inheritance Hierarchies in Knowledge Representation and Programming Languages*, pages 29–49. John Wiley & Sons Ltd, Chichester, West Sussex, 1991.

[Baa 94] F. Baader, B. Hollunder, B. Nebel, H.J. Profitlich, and E. Franconi. An Empirical Analysis of Optimization Techniques for Terminological Representation Systems. *Journal of Applied Intelligence*, 4(2): 109–132, 1994. Also published in the Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, Massachusetts, pages 270–281, 1992.

[Bor 92] A. Borgida. From Type Systems to Knowledge Representation: Natural Semantics Specifications for Description Logics. *International Journal of Intelligent and Cooperative Information Systems*, 1(1): 93–126, 1992.

[Bor 94] A. Borgida and P.F. Patel-Schneider. A semantics and complete algorithm for subsumption in the Classic description logic. *Journal of Artificial Intelligence Research*, 1: 277–308, 1994.

[Bou 96] I. Bournaud. *Regroupement conceptuel pour l'organisation de connaissances*. Thèse de l'Université Pierre et Marie Curie, Paris 6, 1996.

[Bra 91] R.J. Brachman, D.L. McGuinness, P.F. Patel-Schneider, L.A. Resnick, and A. Borgida. Living with CLASSIC: When and How to Use a KL-ONE Language. In J. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 401–456. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1991.

[Buc 93] M. Buchheit, F.M. Donini, and A. Schaerf. Decidable Reasoning in Terminological Knowledge Representation Systems. *Journal of Artificial Intelligence Research*, 1: 109–138, 1993.

[Cap 95] C. Capponi. *Identification et exploitation des types dans un modèle de connaisances à objets*. Thèse de l'Université Joseph Fourier, Grenoble, 1995.

[Car 95] B. Carré, R. Ducournau, J. Euzenat, A. Napoli, and F. Rechenmann. Classification et objets : programmation ou représentation ? In *Actes des Cinquièmes Journées Nationales PRC-GDR Intelligence Artificielle, Nancy*, pages 213–237, Toulouse, 1995. Teknea.

[Che 92] M. Chein and M.L. Mugnier. Conceptual graphs : fundamental notions. *Revue d'intelligence artificielle*, 6(4): 365–406, 1992.

[Cho 96] M.-P. Chouvet, F. Le Ber, J. Lieber, L. Mangelinck, A. Napoli, and A. Simon. Analyse des besoins en représentation et raisonnement dans une représentation à objets — L'exemple de Y3. In Y. Dennebouy, editor, *Actes du Colloque Langages et Modèles à Objets (LMO'96), Leysin, Suisse*, pages 150–169. École Polytechnique Fédérale de Lausanne, 1996.

[Cra 96] I. Crampé and J. Euzenat. Fondements de la révision dans un langage d'objets simple. In Y. Dennebouy, editor, *Actes du Colloque Langages et Modèles à Objets (LMO'96), Leysin, Suisse*, pages 134–149. École Polytechnique Fédérale de Lausanne, 1996.

[Dek 93] L. Dekker. La réification de filtres en FROME. In M. Habib and M. Oussalah, editors, *Actes de la Conférence Représentations Par Objets, La Grande Motte*, pages 23–35, 1993.

[Dek 94] L. Dekker. FROME *: représentation multiple et classification d'objets avec points de vue*. Thèse de l'Université des Sciences et Technologies de Lille, 1994.

[Dic 94] H. Dicky, C. Dony, M. Huchard, and T. Libourel. ARES, un algorithme d'Ajout avec REStructuration dans les hiérarchies de classes. In F. Rechenmann, editor, *Actes du Colloque Langages et Modèles à Objets (LMO'94), Grenoble*. IMAG, Grenoble, 1994.

[Dic 95] H. Dicky, C. Dony, M. Huchard, and T. Libourel. ARES, Adding a class and REStructuring Inheritance Hierarchies. In *11ièmes journées Bases de Données Avancées, Nancy*, 1995.

[Dic 96] H. Dicky, C. Dony, M. Huchard, and T. Libourel. On Automatic Class Insertion with Overloading. In *Proceedings of 11th OOPSLA, San Jose, California, Special Issue of ACM SIGPLAN Notices (31)10*, pages 251–267, 1996.

[Don 94] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in Concept Languages: From Subsumption to Instance Checking. *Journal of Logic and Computation*, 4: 1–30, 1994.

[Duc 95] R. Ducournau, M. Habib, M. Huchard, M.L. Mugnier, and A. Napoli. Le point sur l'héritage multiple. *Technique et science informatiques*, 14(3): 309–345, 1995.

[Duc 96a] R. Ducournau. Des langages à objets aux logiques terminologiques : les systèmes classificatoires. Rapport de Recherche 96-030, LIRMM, Montpellier, 1996. ftp.lirmm.fr/pub/LIRMM/papers/1996/RRI-Ducournau-96.ps.

[Duc 96b] R. Ducournau. Les incertitudes de la classification incertaine. In Y. Dennebouy, editor, *Actes du Colloque Langages et Modèles à Objets (LMO'96), Leysin, Suisse*, pages 183–200. École Polytechnique Fédérale de Lausanne, 1996.

[Euz 93] J. Euzenat. Définition abstraite de la classification et son application aux taxonomies d'objets. In M. Habib and M. Oussalah, editors, *Actes de la Conférence Représentations Par Objets, La Grande Motte*, pages 235–246, 1993.

[Euz 95] J. Euzenat and F. Rechenmann. SHIRKA, 10 ans, c'est TROPES ? In A. Napoli, editor, *Actes du Colloque Langages et Modèles à Objets (LMO'95), Nancy*, pages 13–34. Unité de Recherche Inria Lorraine, Nancy, 1995.

[Gir 95] P. Girard. *Construction hypothétique d'objets complexes*. Thèse de l'Université Joseph Fourier, Grenoble, 1995.

[God 93] R. Godin and H. Mili. Building and Maintaining Analysis-Level Class Hierarchies Using Galois Lattices. In *Proceedings of 8th OOPSLA, Washington, DC, Special Issue of ACM SIGPLAN Notices (28)10*, pages 394–410, 1993.

[Gui 95] O. Guinaldo. Techniques d'indexation pour aider à la classification dans le modèle des graphes conceptuels. In A. Napoli, editor, *Actes du Colloque Langages et Modèles à Objets (LMO'95), Nancy*, pages 53–66. Unité de Recherche Inria Lorraine, Nancy, 1995.

[Gui 96a] O. Guinaldo. Conceptual graphs isomorphism – algorithm and use. In *Proceedings of the 4^{th} International Conference on Conceptual Structures (ICCS'96)*, Lecture Notes in Artificial Intelligence 1115, pages 160–174, Sydney, Australia, 1996. Springer-Verlag.

[Gui 96b] O. Guinaldo. *Étude d'un gestionnaire d'ensembles de graphes conceptuels*. Thèse de l'Université des Sciences et Techniques du Languedoc, Montpellier, 1996.

[Hae 95] O. Haemmerlé. *CoGITo : une plate-forme de développement de logiciels sur les graphes conceptuels conceptuels*. Thèse de l'Université des Sciences et Techniques du Languedoc, Montpellier, 1995.

[Lec 94] M. Leclère. Diverses problématiques de classification dans le modèle des graphes conceptuels. In F. Rechenmann, editor, *Actes du Colloque Langages et Modèles à Objets (LMO'94), Grenoble*, pages 81–94. IMAG, Grenoble, 1994.

[Lec 95] M. Leclère. *Le niveau terminologique du modèle des graphes conceptuels : construction et exploitation*. Thèse de l'Université des Sciences et Techniques du Languedoc, Montpellier, 1995.

[Lec 96] M. Leclère. C-chic : construction coopérative de hiérarchies de catégories. *Revue d'intelligence artificielle*, 10(1): 57–100, 1996.

[Lev 87] H.J. Levesque and R.J. Brachman. Expressiveness and Tractability in Knowledge Representation and Reasoning. *Computational Intelligence*, 3(2): 78–93, 1987.

[Mar 93] O. Mariño. *Raisonnement classificatoire dans une représentation à objets multi-points de vue*. Thèse de l'Université Joseph Fourier, Grenoble, 1993.

[Mar 96] J.F. Mari and A. Napoli, editors. *Actes du séminaire Aspects de la classification, Nancy*, 1996. Rapport de Recherche INRIA RR-2909.

[Mug 96] M.-L. Mugnier and M. Chein. Représenter des connaissances et raisonner avec des graphes. *Revue d'intelligence artificielle*, 10(1): 7–56, 1996.

[Nap 93] A. Napoli and C. Laurenço. Représentations à objets et classification – Conception d'un système d'aide à la planification de synthèses organiques. *Revue d'intelligence artificielle*, 7(2): 175–221, 1993.

[Nap 94a] A. Napoli. Catégorisation, raisonnement par classification et raisonnement à partir de cas. In *Actes des Journées Acquisition – Validation – Apprentissage (JAVA'94), Strasbourg*, pages E1–E14, 1994.

[Nap 94b] A. Napoli, C. Laurenço, and R. Ducournau. An Object-Based Representation System for Organic Synthesis Planning. *International Journal of Human-Computer Studies*, 41(1/2): 5–32, 1994.

[Neb 90] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems.* Lecture Notes in Artificial Intelligence 422. Springer-Verlag, Berlin, 1990.

[Neb 92] B. Nebel. Syntax-Based Approaches to Belief Revision. In P. Gärdenfors, editor, *Belief Revision*, Cambridge Tracts in Theoretical Computer Science 29, pages 52–88. Cambridge University Press, Cambridge, 1992.

[Par 96] M. Paramasivam and D.A. Plaisted. Automated Deduction Techniques for Classification in Concept languages. Technical Report TR96-015, CS Department, University of North Carolina, Chapel Hill, 1996. Available by anonymous ftp at ftp.cs.unc.edu in pub/technical-reports.

[PS 89] P.F. Patel-Schneider. Undecidability of Subsumption in NIKL. *Artificial Intelligence*, 39: 263–272, 1989.

[PS 90] P.F. Patel-Schneider. Practical, Object-Based Knowledge Representation for Knowledge-Based Systems. *Information Systems*, 15(1): 9–19, 1990.

[Rec 88] F. Rechenmann. SHIRKA *: système de gestion de bases de connaissances centrées-objet. Manuel de référence.* INRIA/ARTEMIS, Grenoble, 1988. ftp://ftp.inrialpes.fr/pub/sherpa/rapports/manuel-shirka.ps.gz.

[Sow 84] J.F. Sowa. *Conceptual Structures - Information Processing in Mind and Machine.* Addison-Wesley, Reading, Massachusetts, 1984.

[SS 91] M. Schmidt-Schauss and G. Smolka. Attributive Concept Descriptions With Complements. *Artificial Intelligence*, 48(1): 1–26, 1991.

[Tro 95] Projet Sherpa, Tropes 1.0: Reference Manual. Rapport interne, INRIA Rhône-Alpes, Grenoble, France, 1995. ftp://ftp.inrialpes.fr/pub/sherpa/rapports/tropes-manual.ps.gz.

[Vis 92] P. Vismara, J.C. Régin, J. Quinqueton, M. Py, C. Laurenço, and L. Lapied. RESYN – Un système d'aide à la conception de plans de synthèse en chimie organique. In *Actes des 12^e Journeés Internationales Intelligence Artificielle, Systèmes Experts, Langage Naturel, Avignon*, pages 305–318, 1992.

[Vis 95] P. Vismara. *Reconnaissance et représentation d'éléments structuraux pour la description d'objets complexes. Application à l'élaboration de stratégies de synthèse en chimie organique.* Thèse de l'Université des Sciences et Techniques du Languedoc, Montpellier, 1995.

[Vis 96] P. Vismara. Appariements dirigés pour le raisonnement par classification sur des hiérarchies de graphes. In Y. Dennebouy, editor, *Actes du Colloque Langages et Modèles à Objets (LMO'96), Leysin, Suisse*, pages 201–214. École Polytechnique Fédérale de Lausanne, 1996.

[Win 75] T. Winograd. Frame Representation and the Declarative/Procedural Controversy. In D.G. Bobrow and A. Collins, editors, *Representation and Understanding: Studies in Cognitive Science*, pages 185–210. Academic Press, New York, 1975.