Petits riens[1]

Dictionnaire informatique par intermittence

Roland Ducournau — LIRMM[2]

1994-95

[1]Autres titres possibles : “Chosités”, “Objectités” ou “Petits choses”. C’est à peu (de chose) près le sens littéral de “Smalltalk”. Les “riens” font bien sûr référence à la “Glissade sémantique” de Quine : voir POLYMORPHISME. L’auteur est profondément redevable aux membres du groupe *objets – classification* du PRC-IA, en particulier à Bernard Carré, Jérôme Euzenat, Amedeo Napoli et François Rechenmann, pour les nombreuses discussions préalables à la rédaction de ce document ainsi que pour leurs commentaires (im)pertinents des premières versions.

[2]Ce travail a été commencé alors que l’auteur était à Sema Group.

Résumé

Ce travail est né d'une discussion et d'une lecture. La discussion a eu lieu dans le cadre du groupe *objet – classification* du PRC-IA et a abouti à une présentation aux journées nationales du PRC-IA [Carré *et al.*, 1995]. La lecture a été celle du philosophe W.V. Quine, en particulier de "Quiddités" [Quine, 1993], auquel le titre et la forme du dictionnaire ont été empruntés.

Le groupe *objets – classification* du PRC-IA, s'est donné comme but l'étude du point commun à toutes les manifestations des *objets*, à savoir le regroupement des objets en classes décrites par un ensemble de propriétés, l'organisation taxinomique de ces classes en une structure de classification dont un mécanisme spécifique — la *classification* justement — permet d'assister la construction ou l'évolution. Ce document est un dictionnaire sur les objets vus du point de vue de la RCO, destiné avant tout à la communauté objet elle-même. Les notions de base sont supposées connues, mais le vocabulaire en sera discuté, avec quelques prétentions normatives.

Préface à la réédition (sic) électronique (2008). Ce document dormait sagement dans mon système de fichiers depuis le 28 juin 1995. C'est un reflet personnel des discussions au sein du groupe *Objets et Classification* du PRC-IA entre 1992 et 1995, et un préambule à mon travail sur les *Systèmes classificatoires* [Ducournau, 1996] qui s'est poursuivi jusqu'en 2000. Cela n'a plus grand chose à voir avec ce que je fais depuis. Encore que ...

Bien entendu, à part quelques modifications syntaxiques concernant LaTeXet BibTeX, tout le texte est d'origine : à une douzaine d'années de distance, je est un autre.

Introduction

Ce travail est dédié à la mémoire de Lenneke Dekker,
disparue accidentellement le 1^e juillet 1994.

La notion d'objet a pris une importance considérable dans l'informatique d'aujourd'hui [Masini *et al.*, 1989], que ce soit en informatique classique avec les langages de programmation à objets (LPO), les méthodologies de conception objets (MCO) ou les bases de données orientées objet (BDO), ou en intelligence artificielle (IA) avec les systèmes de représentation des connaissances à objets (RCO), auxquels on peut assimiler, dans une certaine mesure, les logiques terminologiques (LT) [Woods et Schmolze, 1992].

Dans ce contexte diversifié, le groupe *objets – classification* du PRC-IA, s'est donné comme but l'étude du point commun à toutes ces manifestations des *objets*, à savoir le regroupement des objets en classes décrites par un ensemble de propriétés, l'organisation taxinomique de ces classes en une structure de classification dont un mécanisme spécifique — la *classification* justement — permet d'assister la construction ou l'évolution [Carré *et al.*, 1995]. Au-delà des effets de mode, le succès de l'approche objet tient certainement à l'étroite correspondance entre ce tronc commun et une approche *ontologique* de la réalité [Wand, 1989 ; Bunge, 1977 ; Bunge, 1979].

Par delà les point communs à toutes ces manifestations des objets, le groupe s'intéresse à la mise en évidence — tant au niveau des principes qu'au niveau des réalités — des différences entre LPO et RCO dont la proximité apparente ne doit pas cacher l'irréductibilité radicale. Irréductibilité n'étant pas incompatibilité, le dernier but du groupe est l'étude des possibilités de cohabitation, voire d'intégration.

La problématique du groupe reproduit ainsi la structure argumentaire[1] ternaire de nos humanités :

thèse dans toutes ses manifestations, l'approche objet repose sur un modèle sous-jacent commun et "naturel" ;

antithèse mais ses deux branches LPO et RCO sont irréductibles l'une à l'autre [Euzenat, 1994b] ;

synthèse comment peut-on faire cohabiter LPO et RCO ? L'intégration de la seconde dans le premier a-t-elle un sens ?

Ni le groupe, ni les auteurs n'ont une vision définitive et consensuelle sur tous ces points. Cet article s'essaie donc à rendre compte de l'état des réflexions et des discussions voire des désaccords du groupe en totalité, mais il reflète d'abord les points de vue de ses auteurs. Ces derniers peuvent ainsi se positionner sur plusieurs axes socio-psychologiques ou politico-religieux :

- les *intégristes* s'opposent aux *œcuménistes* dans leur vision de la cohabitation entre RCO et LPO ;
- les *autoritaires* (ou *sécuritaires* ?) s'opposent aux *libertaires* dans la liberté qu'ils sont prêts à accorder aux utilisateurs des systèmes qu'ils conçoivent ;
- les *optimistes* s'opposent aux *pessimistes* quant à la possibilité d'atteindre le paradis (de la représentation des connaissances ou des langages de programmation [Aït-Kaci, 1991]).

Les différents sujets d'opposition sont, comme souvent, de l'ordre du verre à moitié vide ou du verre à moitié plein. Plus généralement, les thèmes débattus dans le groupe se retrouvent au cœur

[1] Une *trope* en quelque sorte... Voir MARMITE.

des discussions des conférences *Représentation Par Objets* (RPO) de La Grande Motte [Habib et Oussalah, 1992 ; Habib et Oussalah, 1993] et *Langages et Modèles à Objets* (LMO) de Grenoble [Rechenmann, 1994] ou des workshops *Object-Based Representation Systems* de IJCAI'93 [Napoli, 1993] ou *Integrating Object-Orientation and Knowledge Representation* de ECAI'94 [Cunis *et al.*, 1994].

Ce travail est né de cette discussion dans le cadre du PRC-IA, et d'une lecture, celle du philosophe W.V. Quine, en particulier de "Quiddités" [Quine, 1993], auquel le titre et la forme du dictionnaire ont été empruntés, en respectueux hommage. Sa préface à [Quine, 1993] montre bien quelle liberté — pour le lecteur autant que pour l'auteur — procure l'ordre lexicographique qui autorise tous les désordres et tous les coq-à-l'âne. L'article "Liberté" du même ouvrage montre tout aussi bien le prix à payer — par le lecteur du moins — pour cette liberté. Par ailleurs, le même [Quine, 1993] a des préoccupations de vocabulaire voisines des nôtres, mais dans un sens subtilement différent[2]. Il s'agit donc ici d'un dictionnaire sur les objets vus du point de vue de la RCO, destiné avant tout à la communauté objet elle-même. Les notions de base sont supposées connues, mais le vocabulaire en sera discuté, avec quelques prétentions normatives. Un dictionnaire représente l'une des formes les plus courantes de méta-langage, lequel sert, comme chacun sait, à parler d'un *langage-objet*. Pour aider le lecteur à distinguer l'usage de la mention — voir "Usage *versus* mention" in [Quine, 1993] — nous avons utilisé, autant que possible, des "guillemets", malheureusement pas toujours dans le même sens. Les termes en PETITES CAPITALES renvoient à des points d'entrée du dictionnaire, sauf rares exceptions.

Ultime avertissement : en partant d'une discussion d'apparence limitée, beaucoup de grands débats de l'Intelligence Artificielle, voire de la philosophie, peuvent surgir au détour de la phrase. Que le lecteur soit indulgent devant le nombre de portes ouvertes que nous aurons enfoncées ...

> Cet ouvrage fait partie d'une série décousue, de livres décousus qui s'inspirent du *Dictionnaire philosophique* de Voltaire. Ce que disait Voltaire du Saint Empire romain s'applique ici *mutatis mutandis* : son livre n'était pas un dictionnaire et il n'était pas philosophique[3].
>
> Willard Van Orman Quine, Préface à *Quiddités*, 1993.

[2] Si ce pensum-ci sait donner envie de lire ce Quine-là, il n'aura pas complètement démérité.

[3] Voltaire aurait-il donc lu *Le parapluie de l'escouade* d'Alphonse Allais, dont le titre s'imposait, d'après l'auteur lui-même, par le fait qu'il n'y était question ni d'escouade ni de parapluie ?

A-D

acquisition (des connaissances)

[Kaindl, 1994] note l'analogie entre les MCO et l'acquisition des connaissances, cette dernière discipline se préoccuppant de plus en plus de modélisation. Il faudrait mieux analyser la séquence 1) RÉALITÉ, 2) MODÉLISATION, 3) REPRÉSENTATION, 4) IMPLÉMENTATION — en oubliai-je ? — en la mettant en rapport avec les notions d'ANALYSE et de CONCEPTION (génie logiciel) et d'acquisition des connaissances (intelligence artificielle).

acteur

La RCO ne se désintéresse pas des acteurs et autres objets concurrents, mais elle attend qu'ils soient mûrs, au niveau LPO, pour que l'on puisse commencer à exiger d'eux les critères de la RCO. Dit autrement : on ne va pas compliquer le débat en rajoutant en plus la concurrence, qui est une notion orthogonale, sans incidence sur la RCO.

amorçage

> [...] je ne connais pas de formalisme qui permette d'exprimer clairement ce qui se passe lors d'un amorçage ; cela traduit bien le manque d'intérêt actuel vis-à-vis de ce mécanisme qui devrait être plus sérieusement étudié par les théoriciens.
>
> J. Pitrat, *Métaconnaissance*, 1990

Voir BOOTSTRAP.

analyse

Quelle différence y a-t-il entre l'analyse et la CONCEPTION ? S'il faut en croire [Girod, 1991], l'analyse s'occupe de la mise en relation de la RÉALITÉ avec le modèle de conception. "Analyse" et MODÉLISATION seraient donc plus ou moins synonymes, et une méthode d'analyse ne serait ni plus ni moins qu'une méthode d'ACQUISITION des connaissances.

antithèse

Le deuxième temps de la dialectique (hegelienne) : mais les deux principales tendances de l'approche objet, LPO et RCO, sont irréductibles l'une à l'autre [Euzenat, 1994b]. Voir THÈSE, SYNTHÈSE, DÉCLARATIF, PROCÉDURAL, FAIRE.

attachement procédural

Le terme est trop général puisque l'on peut très bien l'appliquer aux méthodes. RÉFLEXE, dû semble-t-il à Ferber [Ferber, 1983], est bien préférable.

attribut

Le terme désigne les propriétés des objets qui décrivent leur état. On ne discutera pas de ses nombreux synonymes "champ", "variable", *"slot"* etc. Les attributs sont valués, ce qui différencie cet usage de l'usage philosophique classique de "propriété" ou "attribut" qui désignent plutôt des prédicats. Le domaine de valeur des attributs constitue l'essentiel de la description qui en est faite dans les classes (l'INTENSION de la classe). Ils peuvent être en particulier MONO-VALUÉS ou MULTI-VALUÉS, être une RELATION prenant ses valeurs dans l'ensemble des objets, ou une valeur "simple". Voir RÔLE.

Les attributs décrivent en général des conditions NÉCESSAIRES : toute instance d'une classe doit les posséder en en vérifiant les contraintes. Dans certain cas, ces conditions peuvent aussi être SUFFISANTES : tout objet les vérifiant est, de plein droit, une instance de la classe.

autoritaire

Tout ce qui n'est pas interdit est autorisé. Qui n'est pas contre moi est avec moi. L'informatique et les langages formels sont volontiers autoritaires, souvent TOTALITAIRES, plus rarement LIBERTAIRES.

BDO (base de données objet)

Une hérésie courante consiste à implémenter un modèle objet au-dessus d'un modèle relationnel, qu'il s'agisse de bases de données ou d'un LPO/RCO au-dessus de PROLOG comme OBJLOG. En matière de performance, l'un des intérêts du modèle objet est l'efficacité avec lequel il implémente les RELATIONS BINAIRES [Caseau, 1991] : en gros, suivre une relation (mono-valuée) se fait en temps constant. Passer par l'intermédiaire du modèle relationnel fait perdre immédiatement cet avantage. Reste, bien sûr et quand même, le modèle lui-même.

bootstrap

> Car, pour forger, il faut un marteau, et pour avoir un marteau, il faut le fabriquer. Ce pourquoi on a besoin d'un autre marteau et d'autres outils, et pour les posséder, il faut encore d'autres instruments, et ainsi infiniment.
>
> Spinoza, *De la réforme de l'entendement*[4], 1661

Le tour de passe-passe qui donne lieu à la création d'une classe qui est sa propre instance, la première MÉTA-CLASSE : littéralement transcendant. C'est la seule exception au principe de Bunge-Lavoisier de la sémantique ONTOLOGIQUE. Cela semble mystérieux, mais ce n'est qu'une mystification[5] : le programmeur est Dieu. On peut voir aussi le *bootstrap* comme une mutation, une transmutation ou une métamorphose : le passage de la chrysalide au papillon, de l'entité à l'OBJET.

[Pitrat, 1990] utilise plus judicieusement, mais dans un sens élargi, le terme d'AMORÇAGE.

Le problème du *bootstrap* est qu'il ne peut pas se comprendre dans les termes du système résultant qu'il transcende absolument, à l'instar du *big bang*. Ainsi, le premier bootstrap qui donne naissance à un système à objets réflexif ne peut pas s'interpréter en termes d'*objets* mais seulement en termes d'*objets de première classe* du langage d'implémentation, LISP par exemple. En revanche, un second bootstrap qui donnerait naissance à une nouvelle méta-classe instance d'elle-même, sous-classe de la première, peut se comprendre purement en termes d'objets, la méta-classe devenant sa propre instance par un mécanisme de migration résultant par exemple d'une classification. ce n'est donc plus un bootstrap.

catégorie

Informellement, c'est un synonyme de CONCEPT, CLASSE ou TYPE, plus particulièrement utilisé en sciences cognitives. Comme c'est le seul parmi tous ces termes à ne pas être trop connoté informatiquement — mais il l'est mathématiquement —, on peut le prendre pour désigner informellement les entités génériques.

catégorisation

Terme proposé pour désigner le processus de construction d'une hiérarchie de catégories, pour ne pas confondre avec CLASSIFICATION qui est déjà pris pour l'insertion d'une nouvelle catégorie ou d'un individu. La catégorisation correspond donc à la "classification automatique" de l'analyse de données.

Lorsque la construction de la hiérarchie n'est pas assurée par un dispositif automatique comme le *classifieur* des LT, le problème se pose naturellement de fournir des aides à la réorganisation d'une hiérarchie produite par une catégorisation manuelle [Dicky *et al.*, 1994 ; Casais, 1991].

classe

Désigne les entités génériques dans le modèle classe-instance qui distingue entités générique et entités individuelles. Une classe représente donc un concept abstrait, et décrit, par un

[4] Cité par [Gochet et Gribomont, 1991], page 18.

[5] Roland Barthes distinguait le mystère (personne ne connaît la réponse) de la mystification (quelqu'un connaît la réponse) : si Dieu n'existe pas, la création est un mystère, mais, s'il existe, quelle mystification !

ensemble de PROPRIÉTÉS, (son INTENSION) l'ensemble de ses INSTANCES (son EXTENSION). Une classe a des *super-classes* qu'elle spécialise, et des *sous-classes* qu'elle généralise (ou subsume). Voir HÉRITAGE, CLASSIFICATION, SUBSOMPTION.

Classiquement (*sic*), "classe" et "ensemble" sont quasiment synonymes, même si la théorie axiomatique moderne des ensembles (Zermelo-Fraenkel) définit deux sortes de collections, les classes et les ensembles justement, pour résoudre les paradoxes de la théorie naïve : voir par exemple "classes *versus* ensembles" dans [Quine, 1993], [Gochet et Gribomont, 1994] pour un exposé de la théorie des ensembles, et RUSSEL. Du point de vue informatique qui nous intéresse ici, "classe" se confond plutôt, soit avec TYPE (tendance plus spécifiquement LPO), soit avec CONCEPT ou CATÉGORIE (tendance RCO). Voir PLATON.

classification

> Comment pourrait-on classer les verbes qui suivent : cataloguer, classer, classifier, découper, énumérer, grouper, hiérarchiser, lister, numéroter, ordonnancer, ordonner, ranger, regrouper, répartir ?
>
> Georges Pérec, *Penser/Classer*, 1982[6].

Ce mot décrit à la fois un *être*, la structure hiérarchique des classes aussi appelée taxinomie, et un *faire*, le mécanisme qui permet de rajouter de nouvelles classes dans cette hiérarchie, ou d'y placer (ou reconnaître) des instances. Ce double sens est caractéristique de la RCO. On utilisera HIÉRARCHIE pour le premier usage et on réservera "classification" pour le second. En toute généralité, le mécanisme de classification peut s'interpréter comme un cas particulier de FILTRAGE, l'objet à classer étant considéré comme un filtre.

La possibilité du mécanisme de classification repose sur une sémantique très stricte de la hiérarchie de classes : pour bien faire — c'est-à-dire pour faire simplement — il faut que l'héritage soit monotone, sans exception [Brachman, 1985][7], et que les propriétés soient vues comme des conditions NÉCESSAIRES et SUFFISANTES (CNS). C'est le cas dans les LT, où la classification a été étudiée intensivement [Nebel, 1990]. Le cadre de la RCO pose deux catégories de problèmes par rapport à ce cadre idéal des LT : les objets à classer sont souvent incomplets (des attributs ne sont pas valués), et les classes n'expriment en général que des conditions nécessaires (CN) [Euzenat, 1993b]. Le problème a été traité en SHIRKA, TROPES ou (F)ROME, par une classification tri-valuée où la relation entre un objet et une classe peut être qualifiée de *sûre*, *possible* ou *impossible*, suivant que l'objet vérifie toutes les contraintes de la classe (*sûre*), n'en viole aucune *possible*) ou en viole au moins une (*impossible*). Une classe *possible* traduit ainsi le caractère incomplet de l'instance. La terminologie employée prête à contresens : une classe *sûre* n'autorise en fait une classification automatique que si la classe exprime une CNS. Comme ce n'est pas le cas, *a priori*, en RCO, une classe sûre n'exprime en fait qu'une autre forme de possibilité (cf. [Ducournau, 1996]).

L'un des termes du débat concerne ainsi la possibilité d'avoir une sémantique correcte de la classification en RCO sans pour autant tomber dans les LT [Dekker, 1994 ; Mariño, 1993 ; Napoli, 1992 ; Napoli, 1991]. On peut voir la classification comme le versant *inductif* de l'HÉRITAGE — qui lui est déductif [Ducournau et Habib, 1989 ; Ducournau *et al.*, 1995]. Dans le cas des LT, c'est une induction logiquement correcte donc finalement déductive. En revanche, la classification de la RCO serait purement inductive, donc logiquement non fondée si elle était automatique.

[Coupey et Fouqueré, 1994] propose aussi une classification FLOUe dans les LT : dans son cas, c'est l'instance qui peut être qualifiée de *sûre* ou *probable* et de *typique* ou *exceptionnelle*. Il faut noter que dans ce cas, une instance *probable* traduit d'abord le caractère non nécessaire de la classe. Voir SÉMANTIQUE, TYPICALITÉ.

L'analyse de données désigne sous le nom de "classification" (automatique) le processus de construction de classes à partir des individus, que nous appelons ici CATÉGORISATION.

[6] Dans la citation de Pérec, remplacer le 4^{e} mot par l'un des "verbes suivants". Les essayer tous. Lire à haute voix.

[7] Bien que l'on cherche de plus en plus à se libérer de cette exigence de monotonie [Padgham et Zhang, 1993 ; Padgham et Nebel, 1993 ; Coupey et Fouqueré, 1994].

classificatoire (système)

Ou comment dépasser les clivages traditionnels RCO / LT pour élargir la problématique [Euzenat, 1993a ; Euzenat, 1994a ; Ducournau, 1996]. La philosophie n'est d'ailleurs pas en reste [Tort, 1989].

CLOS

CLOS (*Common Lisp Object System*) est le noyau objet de COMMON LISP, le successeur des FLAVORS, de LOOPS et COMMON LOOPS et de quelques autres. *Meta-object protocol*, héritage multiple par LINÉARISATION, fonctions génériques (multi-méthodes) et combinaison des méthodes sont ses caractéristiques principales. Un LPO pur sucre [Steele, 1990 ; Keene, 1989 ; Kiczales *et al.*, 1991], particulièrement LIBERTAIRE pour beaucoup [Baker, 1991].

complétude

En logique classique, les rapports entre la théorie des modèles et la théorie de la démonstration s'appréhendent en termes de *correction* et de *complétude*. Tout ce que l'on démontre est-il vrai ? Tout ce qui est vrai est-il démontrable ? Les sémantiques OPÉRATIONNELLE et DÉNOTATIONNELLE se prêtent aux mêmes interrogations.

Les LT offrent une variation sur ce thème, avec leurs sémantiques *intensionnelle* ou *extensionnelle* [Woods, 1991] : si la sémantique extensionnelle est clairement dénotationnelle, la sémantique intensionnelle n'est pas exactement opérationnelle (elle est quand même plus déclarative que cela), mais elle relève néanmoins de la théorie de la démonstration. La littérature propose de nombreux résultats de complétude ou d'incomplétude : voir [Borgida et Patel-Schneider, 1994] par exemple. [Dionne *et al.*, 1993 ; Coupey et Fouqueré, 1994] proposent une sémantique dénotationnelle où le domaine est construit comme une algèbre libre (ou universelle ?). On suspecte qu'il pourrait s'agir d'un tour de passe-passe pour résoudre, par construction, le problème de complétude : en fait, les auteurs semblent démontrer une équivalence entre un langage et une algèbre, comme [Caseau, 1991]. Mais ne s'agit-il pas là d'une complétude partielle, restreinte à une certaine classe de domaines ? N'y aurait-il pas quelque "méprisme" [Quine, 1993][8] ?

Dans cette discussion sur la complétude, on présuppose la *décidabilité* du problème, qui n'a rien d'obligatoire : dans les LT, la SUBSOMPTION est souvent indécidable [Schild, 1988 ; Patel-Schneider, 1989 ; Schmidt-Schauß, 1989 ; Borgida et Patel-Schneider, 1994].

Jusqu'ici, la complétude était *sémantique* dans la mesure où elle concernait l'adéquation du système de déduction avec sa sémantique. Il y a aussi une incomplétude *syntaxique*, dont l'archétype est le deuxième théorème de Gödel, et qui concerne la capacité du système d'axiomes à couvrir la "théorie" considérée. Formellement, la représentation des connaissances paraît condamnée à cette incomplétude, la puissance expressive des systèmes étant forcément très inférieure aux besoins.

La représentation des connaissances est aussi, dans son ensemble, sujette à des soucis de complétude : un paradigme de représentation des connaissances donné permet-il de représenter la totalité des connaissances représentables ? Tout dépend bien sûr de ce que l'on entend par connaissances : si l'on remplace connaissances par "monde réel", on se convainc aisément qu'un système de représentation complet permettrait de définir la réalité ! De simples questions de calculabilité, ou le théorème de Gödel, excluent toute complétude.

La physique elle-même n'échappe pas aux questions de complétude : c'est l'incapacité de la mécanique Newtonienne et des équations de Maxwell à rendre compte de la totalité de la réalité qui a amené Einstein à jeter les bases des mécaniques relativiste et quantique. Mais Einstein lui-même doute de la complétude de cette dernière [Einstein *et al.*, 1935][9].

composition

[8] La question est en fait de savoir si les auteurs disposent d'un équivalent du théorème de Herbrand qui, en logique des prédicats, permet d'établir une sémantique constructive en se contentant de l'examen d'un domaine particulier, dit *univers de Herbrand*. Dans ce cas, ce domaine universel contient, en quelque sorte, tous les domaines possibles.

[9] Article écrit au début du (long) séjour d'Einstein à Princeton, en compagnie de Gödel. Coïncidence ?

Les objets s'intéressent de très près aux relations dites de composition, éventuellement sous le nom d'agrégation, dont [Winston *et al.*, 1987] propose une analyse des sémantiques possibles. Curieusement, la logique aussi : Lesniewski la met au cœur de sa logique [Lesniewski, 1989] et [Wand, 1989] la voit au premier rang de l'ontologie de [Bunge, 1977 ; Bunge, 1979]. Certains — au hasard, du côté de C++ — paraissent même avoir confondu héritage et agrégation.

On trouve des implémentations des objets composites dans de nombreux systèmes [Dugerdil, 1988 ; Ducournau, 1991 ; Blake et Cook, 1987]. [Magnan, 1994] en propose une modélisation assez complète.

La composition étant assez orthogonale avec le reste — [Napoli, 1992] parle d'espace à plusieurs dimensions : héritage, composition, temps, etc. — il est bien sûr possible d'étudier la combinaison des objets composites et de tout autre paradigme : POINTS DE VUE ou CLASSIFICATION, par exemple [Mariño, 1991 ; Mariño, 1993 ; Dekker, 1994], ou exceptions [Magnan, 1994].

concept

On utilisera ce terme pour désigner spécifiquement l'entité générique définie et manipulée par les LT, l'équivalent de CLASSE pour les LT.

conception

Au sens propre, c'est mettre un problème sous forme conceptuelle, donc définir les concepts (CLASSES) qui participent à ce problème et leurs RELATIONS. Le modèle *entité / relation* n'est pas loin.

Dans l'industrie du génie logiciel, la division du travail aboutit à considérer la conception comme immaculée, seul le programmeur devant plonger les mains dans le cambouïs de l'implémentation. Dans l'artisanat de la représentation des connaissances, ce partage a-t-il (encore / vraiment) cours ? Voir MCO.

connaissances

Opposition entre connaissances et programmes : si l'on peut toujours considérer les connaissances comme un programme — et réciproquement —, du point de vue de l'application informatique, leurs statuts respectifs sont radicalement différents. Les connaissances constituent en général — voire toujours — un "objet de première classe" de l'application, ce qui n'est jamais le cas des programmes. A la limite, les connaissances peuvent constituer le seul objectif d'une application comme le montre [Grivaud et Rechenmann, 1992].

Opposition entre l'*être* et le *faire* : une base de connaissance représente un état du monde, mais la résolution de problèmes nécessite d'agir sur cet état du monde. Voir ACQUISITION, ÊTRE, PROGRAMME, FAIRE.

contraintes

Les contraintes constituent un élément de représentation plus ou moins déclaratif suivant leur réalisation. On distingue en gros deux catégories de systèmes de contraintes : les CSP et les "contraintes directionnelles" à la GARNET [Myers *et al.*, 1990], KALEIDOSCOPE [Freeman-Benson, 1990b ; Freeman-Benson, 1990a ; Freeman-Benson, 1990c] ou THINGLAB [Borning, 1981].

Les rapports entre contraintes et objets sont, en toute généralité, de trois types [Kökény, 1994] :

1. les contraintes peuvent être implémentées par des objets [Kokeny, 1993] ;
2. les contraintes peuvent porter sur des attributs d'objets, au lieu de porter sur des "variables" non structurées ;
3. les domaines des contraintes peuvent être hiérarchisés, les valeurs étant alors assimilables à des types.

C'est le point 2 qui intéresse la RCO puisqu'il offre une formalisation potentiellement déclarative des contraintes informelles qui portent sur les objets. Les contraintes portent en général sur des attributs mono-valués, mais quelquefois sur des attributs multi-valués, voire sur leur cardinal.

Associer contraintes et objets apporte des problèmes d'ordre divers, dont beaucoup obligent à sortir du cadre strict des CSP :

- problèmes d'efficacité puisqu'en toute généralité le problème est NP-difficile ; s'y greffent des questions formelles, par exemple pour générer automatiquement des réflexes de FILTRAGE à la propagation comme [Caseau, 1993a] ;
- problèmes d'expressivité et de sémantique : comme les contraintes portent sur des attributs d'objets dont les valeurs sont des objets dont les attributs sont eux-mêmes contraints, le système de contraintes devient dynamique. L'instanciation d'une variable peut induire l'ajout ou le retrait d'une nouvelle contrainte, voire la création de nouveaux objets. Les notions de *méta-contrainte* ou de contrainte d'ordre supérieur ont été proposées comme cadre de ces systèmes de contraintes dynamiques [Berlandier et Neveu, 1992].
- problèmes de relaxation des contraintes : un CSP est un problème simple à énoncer mais insuffisant dès qu'il n'a plus de solution. On cherche alors à calculer une solution approchée — mais il faut d'abord définir ce que c'est — en relâchant quelques contraintes. Les *hiérarchies de contraintes* ont été proposées [Borning *et al.*, 1987] comme cadre général à ce problème.
- problème d'évolution des solutions : à partir d'une solution, le problème est d'obtenir une solution "voisine" après une modification "locale" du CSP ou de l'instanciation de quelques variables.
- l'intelligence artificielle est très sensible à la question des explications, mais les CSP font partie de ces problèmes très peu modulaires dans leur traitement, donc difficilement explicables[10]. Expliquer l'absence de solution d'un CSP, ou l'absence de solution contenant une certaine instanciation partielle, doit être aussi difficile (en termes de complexité) que le calcul de la solution.

Dans tous les cas, il est nécessaire de formaliser toutes ces notions dans un cadre élargi et unifié.

coréférence

> De vrais questions sur l'identité peuvent surgir lorsque nous avons deux manières de nous référer à quelque chose et que nous pouvons à bon droit nous demander si c'est à chaque fois de la même chose qu'il s'agit.
>
> W.V. Quine, "Identité" in [Quine, 1993]

La coréférence réside dans le fait que deux expressions différentes font référence à la même chose. Ferber et Volle [**?** ; Volle, 1989 ; Ferber, 1983] ont proposé une technique de représentation basée sur la coréférence et qui se place dans la mouvance des techniques de points de vue. Ne pas confondre signification et référence : Frege s'est lui aussi intéressé à la coréférence : "le vainqueur d'Iéna" et "le vaincu de Waterloo" ont deux significations différentes mais font tous deux référence à Napoléon [Frege, 1971]. Voir PERSPECTIVE.

CSP (constraint satisfaction problem)

Un CSP se définit très simplement par la donnée d'un DOMAINE, d'un ensemble de variables et d'un ensemble de CONTRAINTES portant sur ces variables. Le but du problème est de trouver une (ou toutes les) valuation(s) des variables satisfaisant les contraintes (voir [Kökény, 1994] par exemple).

Il faut noter qu'un "vrai" CSP ainsi défini a une sémantique parfaitement déclarative (à condition d'en considérer l'ensemble des solutions : voir PROLOG), ce qui n'est pas le cas des systèmes de contraintes à la GARNET [Myers *et al.*, 1990] dont non seulement la sémantique n'est qu'opérationnelle, mais dont on suspecte volontiers qu'elle n'est pas entièrement spécifiée, donc qu'elle repose sur l'implémentation (cf. note 11). En particulier, rien ne garantit qu'une solution d'un ensemble de contraintes vérifie bien toutes ces contraintes, ce qui est pourtant la moindre des choses. La plupart des extensions du problème du CSP provoque le même scepticisme.

[10] On verrait presque poindre une autre sorte de déclarativité, celle du traitement ! Les techniques numériques de reconnaissance des formes — traitement du signal ou réseau de neurones — ont souvent, voire toujours, ce défaut de ne pouvoir expliquer leur résultat : leur traitement n'est pas décomposable.

déclaratif

La controverse déclaratif / procédural est absolument inépuisable, les deux positions extrêmes déjà notées par Winograd (op. cité) étant : "tout est déclaratif" ou "tout est procédural". C'est donc clairement une affaire de degré et de point de vue.

La *déclarativité* semble reposer sur la présence relative de trois caractéristiques étroitement corrélées[11] :

1. une certaine *indépendance vis-à-vis du contexte*, ou *modularité*, qui fait que les connaissances peuvent être rentrées "en vrac" (en particulier sans ordre), et examinées (et appréhendées) isolément. Cette indépendance est bien sûr relative. Voir DÉCOMPOSABLE.
2. une certaine *indépendance vis-à-vis des traitements* qui fait que l'on n'a pas besoin de savoir comment le système va se servir d'une connaissance précise. Dit autrement, on n'a pas besoin de faire tourner un programme pour savoir ce qu'il signifie.
3. une *sémantique immédiate* (et intuitive) qui permet d'associer un sens à un élément de connaissance isolé.

Ces trois points sont les trois faces de la même hyper-médaille : la sémantique immédiate ne peut s'appliquer qu'à un élément pris isolément, etc. Le dernier point ne paraît pas figurer, au moins explicitement, dans les critères classiques de la déclarativité, mais il nous semble tellement fondamental que son absence doit résulter d'une évidence trop forte. Voir PROCÉDURAL, DÉCOMPOSABLE, QUANTIQUE.

déclarative (sémantique)

On appellera *sémantique déclarative* une sémantique dénotationnelle suffisamment intuitive, lorsqu'elle autorise une indépendance suffisante vis-à-vis des traitements ou du contexte. Les *pessimistes* notent que ces trois caractéristiques de la déclarativité sont des vœux pieux : que ce soit vis-à-vis du contexte ou des traitements, l'indépendance est toujours relative, et la sémantique intuitive trompeuse. Plus précisément, les sémantiques formelles connues ne savent pas allier intuition, *robustesse* — c'est-à-dire résistance aux bruits et aux contradictions — et déclarativité. Un exemple typique est celui de la logique classique : on peut la trouver intuitive[12] mais elle n'est pas robuste car elle ne supporte pas la contradiction. Pour la rendre robuste, on aboutit inéluctablement à PROLOG ou aux systèmes de PRODUCTION dont la déclarativité n'est pas le point fort. Les *pessimistes* en concluent donc que la déclarativité est trompeuse : il n'y aurait pas de sémantique déclarative robuste non triviale. Voir DÉCLARATIF.

décomposable (système presque)

> [...] the short-run behavior of each of the components is approximately independent of the short-run behavior of the other components. [...] In the long run, the behavior of any of the components depends in only an aggregate way on the behavior of the other components.
>
> Herbert Simon, *The architecture of complexity*, 1969.

Winograd [Winograd, 1975] cite, à propos de déclarativité, les "systèmes presque décomposables" d'Herbert Simon, qui présentent une propriété nécessaire en fait à la compréhension de n'importe quel système. Voir NEWTONIENNE, DÉCLARATIF, QUANTIQUE.

définition

On parlera de "définition" d'une propriété pour désigner l'introduction de cette propriété (de son nom) dans la description d'une classe dont aucune super-classe ne décrit la propriété (une propriété du même nom). La classe sera alors une "classe de définition" de la propriété. Dans un système autorisant des propriétés HOMONYMES, on pourra même parler de LA classe de définition.

[11] On trouve dans la littérature bien d'autres définitions, certaines plus contestables que d'autres. [Horty, 1994] semble ainsi réduire le procédural à ce qui dépend de l'implémentation, rendant ainsi déclaratif tout algorithme bien spécifié ! [Myers *et al.*, 1990] place quant à lui la déclarativité dans l'absence de méthode, pour un langage, GARNET, qui n'a même pas une sémantique opérationnelle complètement spécifiée ! Certains définissent la déclarativité dans les termes mêmes que d'autres utilisent pour définir (informellement) la sémantique formelle des langages de programmation.

[12] Quoique ce n'ait sans doute pas été l'opinion de Lewis Carroll ou d'Aristote.

On parlera aussi de "définition" d'une CLASSE pour désigner l'introduction de cette classe et de son INTENSION. Mais cela ne signifie pas que cette classe soit considérée comme *définie*, par apposition à PRIMITIVE, par des conditions d'appartenance NÉCESSAIRES et SUFFISANTES.

démon

Au sens propre, un démon est une procédure qui scrute en parallèle un système dans l'attente d'un événement spécifique, comme un démon UNIX. Utilisé souvent improprement pour désigner les RÉFLEXES.

dénotationnelle (sémantique)

Au sens large, la sémantique dénotationnelle correspond à la sémantique en théorie des MODÈLES de la logique. Elle est donnée par une fonction d'interprétation entre les termes du langage et un DOMAINE. Ce domaine peut, soit être construit plus ou moins formellement — c'est le cas de [Dionne *et al.*, 1993 ; Coupey et Fouqueré, 1994] où le domaine est une algèbre libre (ou universelle ?), voir COMPLÉTUDE —, soit être un ensemble "plat", sans structure particulière — c'est le cas des sémantiques usuelles des LT, par exemple celle de [Borgida et Patel-Schneider, 1994] —, soit enfin être le *domaine modélisé*. Dans ce dernier cas, on l'appellera une sémantique ONTOLOGIQUE puisque l'interprétation a pour cible la RÉALITÉ. Voir MODÉLISATION.

Au sens strict, la sémantique dénotationnelle a une définition beaucoup plus restrictive [Meyer, 1990 ; Mosses, 1990], d'une façon que nous avons préféré appeler RÉDUCTIONNISTE : [Abadi et Cardelli, 1994] l'utilise pour lui-même par exemple.

dialectique

Une espèce de valse argumentaire, ou une *trope*. Voir MARMITE.

disjoints (concepts)

De façon générale, il faut prendre en compte, en plus de la relation de généralisation / spécialisation entre classes, une relation de compatibilité (ou d'incompatibilité) exprimant le caractère disjoint de leurs extensions. Cette relation sert, dans un cadre de mono-instanciation mais d'HÉRITAGE MULTIPLE, à imposer des restrictions sur la définition de nouvelles sous-classes. En cas de MULTI-INSTANCIATION, elle sert alors à décrire les combinaisons de classe dont l'instanciation est valide. Cette relation d'incompatibilité est partiellement implicite : deux classes qui définissent le même attribut avec des DOMAINES disjoints, sont forcément incompatibles. L'une des façons d'expliciter la disjonction entre concepts consiste à exprimer le fait qu'un ensemble de sous-classes directes d'une classe partitionne (l'extension de) cette classe. C'est le cas, de façon aussi naturelle qu'implicite, en héritage simple avec mono-instanciation ! Voir PRIMITIF, TROPES.

La compatibilité est fortement contrainte par la hiérarchie : si 2 classes C_1 et C_2 sont compatibles (resp. incompatibles), toute super-classe (resp. sous-classe) de C_1 est compatible (resp. incompatible) avec toute super-classe (resp. sous-classe) de C_2. La compatibilité est une relation réflexive et symétrique, mais pas transitive. L'incompatibilité est seulement symétrique. D'autre part, une classe est toujours compatible avec ses super-classes.

En toute généralité, l'incompatibilité devrait être décrite par un hyper-graphe. On doit pouvoir exprimer que 3 classes C_1, C_2 et C_3 sont incompatibles, sans être obligé de l'exprimer pour toutes les combinaisons de sous-classes possibles : $C_1 \wedge C_2$ et C_3, $C_1 \wedge C_3$ et C_2, et $C_2 \wedge C_3$ et C_1 (en cas de treillis et de mono-instanciation, bien sûr). Si l'on dispose, comme dans les LT, d'un opérateur de conjonction de classes, et si l'on applique la relation d'incompatibilité à des classes non atomiques, la relation binaire est équivalente. Curieusement, [Nebel, 1990] restreint la relation d'incompatibilité au cas binaire et aux concepts PRIMITIFS.

domaine

D'un point de vue mathématique, c'est l'ensemble de définition (la source) d'une fonction (ou plus généralement d'une relation binaire). Dans la pratique, "domaine" est très souvent utilisé à l'envers pour désigner l'ensemble des valeurs possibles de ce qui peut être assimilé à une fonction, donc son *co-domaine* (la cible). C'est le cas des objets, où "domaine" désigne les valeurs possibles d'un attribut, c'est-à-dire l'ensemble des VALEURS qui peuvent être en

RELATION avec l'objet. Dans le cas d'un attribut MULTI-VALUÉ, ce n'est donc pas l'ensemble des parties qui peuvent valuer l'attribut, mais l'ensemble des éléments qui constituent ces parties.

"Domaine" est aussi utilisé pour les CONTRAINTES, pour désigner les valeurs possibles des variables, dans deux sens qui se ramènent aisément l'un à l'autre : le domaine global valable pour toute les variables, et le domaine spécifique de chaque variable, qui peut être considéré comme une contrainte unaire [Kökény, 1994].

Ces deux usages de "domaine" pour les attributs et pour les variables sont cohérents. Lorsque c'est l'attribut d'un objet qui est contraint, le domaine de la variable est une partie du domaine de l'attribut : il n'y a pas de raison qu'ils soient égaux, car le premier concerne l'instance, et le second, la classe.

Enfin, la sémantique DÉNOTATIONNELLE (au sens large) repose sur la donnée d'un domaine plus ou moins arbitraire, qui est pris comme co-domaine d'une fonction d'interprétation dont le domaine est l'ensemble des termes du langage considéré. Le *domaine modélisé* (ou *applicatif*) peut se ramener à cette dernière signification, puisqu'il s'agit alors du domaine de la sémantique ONTOLOGIQUE.

Par ailleurs, "domaine" est le terme consacré pour désigner les ensembles ordonnés inductifs à la base de la sémantique des langages fonctionnels en théorie du point fixe, pour ne pas parler des "domaines sémantiques" de la sémantique DÉNOTATIONNELLE (au sens strict).

E-I

encapsulation

La plupart les LPO mettent l'accent sur l'*encapsulation* des données qui consiste à prolonger la démarche des TYPES abstraits et à l'appliquer aux objets en différenciant le statut des ATTRIBUTS et celui des MÉTHODES [Snyder, 1991]. Les attributs seraient ainsi privés et interfacés par des méthodes. On peut se demander si cette encapsulation n'est pas due à un certain conservatisme, la nouveauté radicale des objets ayant été mal perçue, en particulier par un langage comme C++. S'il s'agit de raisonner en termes de services ou d'interfaces, on ne voit pas pourquoi il faudrait en éliminer l'état de l'objet. S'il s'agit de raisonner en termes de privé ou de public, on ne voit pas pourquoi la ligne de démarcation devrait passer entre état et comportement[13]. Toujours est-il que la RCO n'éprouve pas grand intérêt vis-à-vis de l'encapsulation. L'exemple de CLOS montre bien comment des attributs peuvent être interfacés automatiquement par des méthodes, et les *intégristes autoritaires* font justement remarquer que le fait de pouvoir redéfinir un tel accesseur pour lui faire faire n'importe quoi rend particulièrement peu sûre cette fonctionnalité due originellement à une exigence de sécurité. Le fait que les accès à un attribut soient implémentés par une méthode ou par une primitive est perçu, en RCO, comme un épiphénomène.

ensemble

Les ensembles interviennent de différentes façons dans la sémantique des objets. Ils apparaissent d'abord dans l'interprétation extensionnelle des CLASSES ainsi que dans celle des RELATIONS. Si les classes peuvent être comprises comme des ensembles, ce n'en sont pourtant pas réellement. Réciproquement, tous les ensembles ne correspondent pas à des classes. Les *relations* sont des attributs à sémantique de RELATION BINAIRE, et les attributs MULTI-VALUÉS ont une sémantique ensembliste : on peut les assimiler à une fonction prenant ses valeurs dans l'ensemble des parties du DOMAINE de l'attribut. LORE [Caseau, 1987] est un langage qui repose sur une interprétation ensembliste forte.

Enfin, on peut considérer que les LT définissent leurs concepts comme des ensembles de propriétés — donnant ainsi une interprétation intensionnelle ensembliste des classes —, alors que les attributs des objets sont attachés à des classes. Voir CONCEPT, RÔLE, ONTOLOGIQUE.

équivoque (propriété)

Un nom de propriété est équivoque dans un contexte donné s'il n'est pas possible de déterminer à partir de son nom de quelle propriété il s'agit. Cela suppose bien sûr l'existence de propriétés HOMONYMES. Dans une expression du langage, c'est un conflit de nom qui apparaît dans l'HÉRITAGE.

être

> Ils ne se doutaient pas que l'assiette pleine cachait
> l'assiette vide, comme l'être cache le néant.
>
> Raymond Queneau, *Le chiendent*.

[13] On peut toujours assimiler les attributs à des fonctions — c'est ce que font toutes les approches de type théorie des types —, mais on perd forcément la spécificité des attributs, à savoir qu'ils représentent un état : non seulement ce sont de "pures fonctions", sans effet de bord, mais en plus elles sont constantes, en première approximation du moins.

Le paradigme objet est réputé permettre de représenter un état (l'*être*), et implémenter un comportement (le *faire*) dans une même entité, l'*objet*. L'état est représenté par les attributs, et le comportement — au moins en pur LPO — par les méthodes. Voir FAIRE, ATTRIBUT, MÉTHODE, TEMPS.

exceptions

Les exceptions à l'HÉRITAGE sont en fait des exceptions au caractère NÉCESSAIRE des conditions exprimées par la description des classes. Ces exceptions pouvant a priori concerner tous les éléments de description d'une classe, on peut donc rencontrer :

- **1.** des exceptions sur les super-classes, qui se traduisent par ce que l'on appelle HÉRITAGE NON MONOTONE et qui permettent d'excepter la totalité des conditions exprimées par une classe (et ses super-classes), à l'exception de ce qui est réaffirmé par ailleurs ;
- des exceptions sur les propriétés, qui peuvent concerner
  - **2.** l'existence de la propriété elle-même,
  - la valeur de tel ou tel "aspect" de la propriété (*masquage*), ce qui traduit
    - **3.** une exception sur le DOMAINE de la propriété,
    - **4.** une exception sur sa valeur par défaut.

Dans la pratique, LPO et RCO ne présentent couramment que des exceptions du quatrième type. Les exceptions du premier type, qui font l'objet des théories d'héritage non monotone, ne sont pas offertes par les LPO ou RCO — à l'exception notable de YAFOOL — et les exceptions de type 2 ou 3 — le premier cas est un cas particulier du second, avec un domaine vide — ne sont pas plus courantes ([Coupey et Fouqueré, 1994] peut-être ?). Les LPO permettent cependant une exception sur la présence d'une MÉTHODE en lui donnant une valeur fonctionnelle d'erreur.

Dans le cas des valeurs par défaut, type 4, il y a deux interprétations possibles :

- la valeur par défaut est une valeur nécessaire pour toutes les instances de la classe qui ne sont pas instances d'une sous-classe qui redéfinit (par masquage) la valeur par défaut.
- la valeur par défaut est une valeur heuristique pour suppléer à l'absence de valeur.

Seul le deuxième cas induit de la non monotonie : dans le premier cas, un raisonnement purement monotone est possible, à condition d'attendre d'être sûr de la non appartenance de l'objet aux sous-classes considérées avant d'utiliser la valeur par défaut.

exclusivité

D'après [Euzenat, 1993a], propriété d'un système CLASSIFICATOIRE dont les extensions des classes sont soit disjointes, soit incluses l'une dans l'autre.

exhaustivité

D'après [Euzenat, 1993a], propriété d'un système CLASSIFICATOIRE pour lequel tout objet est instances de puits de la relation, c'est-à-dire de classes sans sous-classe. Toute taxinomie d'espèces naturelles est exhaustive : il n'existe pas de pur mammifère, qui ne soit ni chat, ni chien, ni baleine, etc.

extensibilité

Le génie logiciel considère que l'un des principaux intérêts des LPO réside dans la possibilité d'étendre un système existant en lui ajoutant de nouvelles classes (et propriétés). Si c'est une évidence pratique, cette extensibilité est en fait limitée en théorie, que ce soit avec les TYPES des LPO, ou avec la sémantique extensionnelle de la RCO, en particulier en ce qui concerne la description des classes DISJOINTES.

Une hiérarchie se traduit en termes d'EXTENSION comme un ensemble recouvrant de parties d'un domaine. Ce recouvrement peut éventuellement posséder certaines propriétés algébriques, comme celle de fermeture par union et intersection : c'est alors un TREILLIS. Dans ces termes extensionnels, l'*extension* dont il s'agit ici se traduit par l'ajout de nouvelles parties, qui doivent respecter certaines propriétés (chaque nouvelle partie doit être incluse dans au moins une ancienne), et éventuellement par une extension du domaine lui-même.

Si l'on suppose donnée une (hyper-)relation d'incompatibilité, on peut avoir deux points de vue différents sur la combinaison de classes qui n'ont pas encore de sous-classes communes :

– la relation d'incompatibilité n'est pas présumée complète, et toute combinaison de classes qui n'est pas interdite par des considérations explicites ou implicites (type des attributs) est considérée comme licite ; dans ce cas, l'interprétation extensionnelle est en quelque sorte minimale : elle décrit l'intersection de toutes les extensions potentielles ; c'est le cas des LT ; l'extension d'une hiérarchie peut se faire dans le demi-treillis inférieur de la combinaison des classes compatibles.
– la relation d'incompatibilité est présumée complète, et toute combinaison de classes qui n'est pas définie est interdite ; dans ce cas, l'interprétation extensionnelle est en quelque sorte maximale : c'est la structure de l'extension réelle ; ce serait le cas de TROPES si on l'étendait pour accepter un héritage multiple dans chaque point de vue, en conservant la complétude de la relation d'incompatibilité dans chaque point de vue (mais pas entre points de vue).

On peut aussi considérer que l'extension du domaine s'interpréte par l'extension des parties préexistantes, ce qui permet de rendre compatibles des classes auparavant incompatibles.

Dans tous les cas, il est toujours possible d'étendre une hiérarchie *par héritage simple*, c'est-à-dire qu'il est possible d'attacher à chaque classe d'*extension propre* non vide la racine de n'importe quelle nouvelle sous-hiérarchie.

L'extension des hiérarchies de TYPES n'est pas plus simple. Si l'on considère qu'un type c'est son EXTENSION associée à un ensemble d'opérations (son INTENSION est réduite à des méthodes), il faut a priori considérer que ces opérations laissent les extensions stables, c'est-à-dire qu'un objet ne va pas changer de type au fur et à mesure des opérations que l'on lui applique. Compte-tenu des objectifs sécuritaires des types, cela paraît essentiel. Définir un sous-type — c'est-à-dire partitionner une des parties en deux — ne peut se faire que si ce nouveau sous-type est bien stable pour toutes les opérations qui s'appliquaient au type. Si on définit un type `personne` avec un attribut `âge` à valeur entière, il n'est pas possible d'avoir à la fois un sous-type pour les personnes d'`âge` pair, et une opération pour incrémenter l'`âge` de toute PERSONNE le jour de son anniversaire[14].

extension

L'extension d'un concept recouvre classiquement l'ensemble des individus qui "tombent sous" le concept. Dans une approche objet, il lui correspond donc l'ensemble des instances d'une classe. La notion d'extension se prête au double sens : il y a l'extension réelle et l'extension potentielle. Théoriquement, on demande une sémantique qui dépende de l'extension potentielle, donc qui soit valable pour toute extension. Dans la pratique, la correction d'une base de connaissances dépendra d'actes de FOI portant sur son extension réelle, c'est-à-dire sur les instances que l'on est susceptible de rencontrer.

"Extension" est aussi pris dans le sens d'"étendre" : voir EXTENSIBILITÉ ou les *extensions linéaires* utilisées, entre autres, pour les LINÉARISATIONS.

"Intension" et "extension" s'utilisent aussi pour les CONTRAINTES dont les relations peuvent être données soit par un prédicat (fonction) qui vérifie que la contrainte est bien satisfaite par l'INSTANCIATION considérée, soit, de façon ensembliste, par un ensemble des tuples qui doit contenir cette instanciation.

facette

Terme utilisé pour désigner le second niveau de description dans les langages de FRAMES. On utilise aussi le mot "aspect" [Faucher, 1992] mais on préfèrera ici le réserver à un usage informel — un aspect d'une propriété, comme dans [Ducournau *et al.*, 1995] — en gardant "facette" pour son usage syntaxique dans les *frames*.

"Facette" est aussi utilisé par [Perrot et Wolinski, 1992] pour désigner un modèle de représentation de la famille des POINTS DE VUE ou PERSPECTIVES. Ces deux usages sont tout à fait cohérents : une facette est une sorte de point de vue sur une propriété.

faire

[14] Si l'exemple vous paraît artificiel, essayez de définir un type `jeune`, avec la même incrémentation de l'`âge`.

Il est bon de s'interroger sur ce qu'un système peut "faire", que ce soit en termes techniques, ou en termes de résolution de problèmes. D'un point de vue technique, agir sur une base de connaissances objet se ramène à trois types d'effets de bord :

1. créer un nouvel objet, ou symétriquement le détruire ;
2. modifier la valeur d'un attribut ;
3. modifier le type d'un objet : le faire migrer ou le (re)classer.

D'un point de vue résolution de problèmes, et en accord avec la fonction de représentation du système à base de connaissances, agir sur cette base peut consister :
- à la *compléter par déduction* : la base ne contient qu'un modèle du monde réel et il faut la compléter par inférences pour en déduire l'état complet du monde ;
- à la *compléter par interrogation* : la base ne contient pas la totalité des connaissance dont elle a besoin pour couvrir son domaine. Il lui faut donc acquérir les informations qui lui manquent (après d'un utilisateur ou d'une base de données) ;
- à la *mettre à jour* : le monde évolue et sa représentation doit suivre ;
- à la *réviser* lorsque la base n'est pas (ou plus) cohérente par exemple à la suite d'une mise à jour ;
- à *agir* à proprement parler sur la réalité pour arriver à un but fixé, que cette action soit réelle ou simulée[15].

Voir ÊTRE.

filtrage

Le mécanisme de requêtes en RCO qui permet de retrouver les classes et/ou instances qui possèdent un ensemble de propriétés, le *filtre*. Ce filtre peut être assimilé à une classe, comme le fait [Dekker, 1994 ; Dekker, 1993]. Les LT ne distinguent pas CLASSIFICATION et filtrage.

Il ne faut pas confondre cet usage de "filtrage" avec celui des CSP ("filtering") qui désigne les techniques de réduction des DOMAINES des variables qui laissent invariantes les solutions.

flou

Les techniques de flou ou de raisonnement incertain semblent rar(issim)es dans la problématique objet. Les attributs à valeur floue ne posent guère de problème, si ce n'est d'implémentation[16] : ils n'apportent qu'une fonctionnalité orthogonale. Le cas typique et critique serait le traitement des "classes floues" et des "instances floues", l'EXTENSION d'une classe étant assimilée à un ensemble flou. La CLASSIFICATION incertaine se contente d'une modalité *possible*, qui est quand même très loin du flou. Le système expert DIVA raisonne bien sur l'incertain mais, à chaque pas de classification, il finit par conclure de façon certaine sur l'appartenance d'une instance à sa classe [David et Krivine, 1987 ; David et Krivine, 1988]. Ni les classes ni les instances ne sont floues, seule la technique de classification l'est. [**?**] propose un FILTRAGE flou qui est du même ordre.

Le système de classification CLASSIC [**?** ; Granger et Thonnat, 1985 ; Granger, 1986 ; Granger, 1988] et divers autres systèmes [Binaghi *et al.*, 1989] font de la classification floue, mais ne semblent pas avoir vraiment des instances floues ? En revanche, [Rossazza, 1990] propose à la fois, classes et instances floues. Voir TYPICALITÉ.

foi (acte de)

> Car la foi, paraît-il, déplace les montagnes. C'est ce qu'on raconte, et je le crois en partie, disait Shakespeare. Quant à moi, j'espère qu'il avait raison.
>
> Robin Cook, *J'étais Dora Suarez*, 1990

Un système de représentation des connaissances repose sur de nombreux actes de foi. On peut en distinguer deux catégories. L'*acte de foi ontologique* s'applique à la relation (de dénotation

[15] Cette classification ne m'emballe pas : elle est due, pour partie, à Jérôme — qui n'en est pas plus fier que ça : j'ai dédoublé la complétion et ajouté le dernier point pour les singes et les bananes, la réalité n'étant qu'un modèle idéalisé de la réalité. Je ne suis pas sûr que ce soit suffisant. D'autres ont dû se pencher sur ce genre de choses, non ?

[16] Si la valeur est floue, c'est assez simple. S'il y a une multitude de valeur, associées à des probabilités ou des possibilités, c'est dejà plus compliqué.

ou de référence) entre la représentation et le monde réel : il est a priori inévitable, tout langage reposant sur un consensus. L'adéquation d'une hiérarchie — pourquoi les cétacés ne sont-ils pas des poissons ? — est un des ces actes de foi inévitables.

L'*acte de foi formel* s'applique à la correction formelle de la représentation, pour tout ce qui n'est pas pris en compte par la syntaxe ou par les mécanismes automatiques du système. Un exemple permettra de mieux voir en quoi peut consister un acte de foi. Dans les systèmes de classification censés aboutir à une classification exacte, unique et certaine (mono-instanciation), un acte de foi est nécessaire si la correction de la hiérarchie (du point de vue de cette propriété là) n'est pas vérifiée automatiquement par le système. Les LT ont, par construction, cette propriété mais ce n'est pas le cas de la totalité des autres sytèmes à objets qui permettent, par exemple, de définir deux classes aux propriétés exactement identiques mais néanmoins incomparables par la relation de subsomption. Les systèmes de classification multi-valués (ternaires) dont l'une des modalités est *sûre* ou *certaine* imposent la même crédulité, sauf à considérer qu'il y a tromperie sur le sens de ces modalités. Dans une TAXINOMIE au sens strict — arborescence dont seules les feuilles sont instanciables —, un acte de foi du même ordre est nécessaire pour vérifier que les domaines d'un attribut dans les sous-classes recouvrent bien le domaine de cet attribut dans la classe, ou pour admettre (postulat) qu'aucune instance ne pourra avoir la combinaison de valeurs correspondant à l'intersection des domaines de deux sous-classes. Dans ce dernier cas, l'acte de foi vient de la différence entre l'EXTENSION réelle du concept et son extension potentielle : c'est quand même un problème ontologique. En termes de théorie des modèles, c'est la distance entre le modèle particulier que le système est censé exprimer, et tous les autres modèles possibles. On voit que la limite n'est pas si nette que ça.

On peut considérer que les actes de foi formels sont une mesure de la la qualité de la sémantique d'un système. L'objectif de la représentation des connaissances devrait donc être d'éliminer tout acte de foi formel, en construisant les mécanismes nécessaires pour y remédier. En particulier, tout système de RCO avec classification devrait faire de la subsomption automatique, comme les LT. Mais, dans ce cas, qu'est-ce qui différencierait RCO et LT ?

Vouloir utiliser des systèmes de production ou PROLOG comme s'il s'agissait d'implémentations de la logique du premier ordre classique, ou prétendre représenter en PROLOG des relations MONO-VALUÉES imposent des actes de foi du même ordre. Il est clair que tout usage du procédural impose lui-même un acte de foi, mais est-il pire que d'assurer à la main les bonnes propriétés formelles de la représentation ?

Ces différents cas diffèrent pourtant sur un point primordial : techniquement, on peut envisager de remédier relativement facilement à certains problèmes, alors que, pour d'autres, il n'existe pas de solution technique. Les problèmes soulevés par la classification — au moins ceux cités plus haut — ne paraissent pas insurmontables, quitte à faire des révisons déchirantes : les LT y remédient bien. En revanche, dans le cas du procédural, il est vain d'espérer quoi que ce soit. Mais qu'il s'agisse de logique ou de classification, il n'est jamais totalement exclu de ne pas tomber soit sur une indécidabilité, soit sur un problème de complexité défavorable.

L'acte de foi formel provient du fait que le système est censé vérifier des propriétés SUBJECTIVES car elles ne sont pas établies par construction par le système mais supposées par l'utilisateur. Pour supprimer ces actes de foi, il faut donc rendre ces propriétés OBJECTIVES, à condition que ce soit possible.

formelle (sémantique)

— Mais à quoi ça sert ? interrogea Jenny, en employant les paroles mêmes d'Isabelle de Castille lorsqu'elle reçut la première grammaire espagnole.

Robin Cook, *Bombe surprise*, 1963.

Le problème de la sémantique des objets commence à avoir fait couler beaucoup d'encre, sans pour autant que les résultats atteints puissent être considérés comme définitifs. L'approche de théorie des types de nombreux auteurs [Abadi et Cardelli, 1994] ne favorise pas

la dissémination de ces résultats, surtout dans la communauté intelligence artificielle. Cependant les *intégristes* de la RCO considèrent l'existence d'une "sémantique propre" comme déterminante. Nous allons donc nous attacher, naïvement, à discuter de ce que peut être une telle sémantique. *Le problème principal est celui de la sémantique qu'il faut accorder à ces sémantiques.*

Le terme de sémantique possède à la fois ubiquité et équivocité : toute syntaxe a sa sémantique, mais lorsque la sémantique est incorporée à la syntaxe — ce qui est le cas des types dans tous les langages typés — quelle en est la sémantique ? Dans le sens qui nous intéresse ici, la sémantique est censée établir la "signification" d'un langage ou d'un système formel. Il y a, semble-t-il, plusieurs façons courantes d'établir une telle signification : OPÉRATIONNELLE, RÉDUCTIONNISTE, DÉNOTATIONNELLE ou ONTOLOGIQUE.

Cette classification est certainement discutable — faut-il réellement distinguer sémantiques réductionniste et dénotationnelle ? —, au moins dans sa terminologie. Il n'y a sans doute pas de solution de continuité entre sémantique ontologique et sémantique réductionniste. Du côté intelligence artificielle, [Winston, 1984] distingue lui aussi trois types de sémantique — *procedural, equivalence and descriptive semantics* — qui correspondent assez bien aux nôtres. Du côté du génie logiciel, donc appliqué aux langages de programmation, [Meyer, 1990] en distingue une quatrième, la *sémantique axiomatique*, et sa définition de la sémantique dénotationnelle — qui est plus officielle que la nôtre — correspond à un domaine mathématique, en général un ensemble de fonctions.

Une comparaison avec la logique classique apporte quelques analogies : la sémantique opérationnelle correspondrait à la théorie de la démonstration — avec donc un côté très "syntaxique" pour une sémantique —, alors que la sémantique dénotationnelle correspondrait à la théorie des modèles[17]. Le même système peut (doit ?) bien sûr avoir plus d'une sémantique — une sémantique opérationnelle et une sémantique dénotationnelle —, dont les rapports s'appréhendent, comme d'habitude, en termes de *correction* et de COMPLÉTUDE : la sémantique opérationnelle respecte-elle la sémantique dénotationnelle ? permet-elle de "démontrer" tout ce qui est vrai dans la sémantique dénotationnelle ?

Dans tous les cas, il est certain que l'évolution des points de vue dont cet article rend compte est due, pour une grande part, à l'apparition des LT et de leurs sémantiques formelles : comme le rappelle [Woods et Schmolze, 1992], les logiques terminologiques sont nées de la volonté de donner une assise sémantique solide aux langages de frames et aux réseaux sémantiques.

frame

La RCO a été identifiée à l'origine aux langages de *frames* [Minsky, 1975 ; Winograd, 1975 ; Masini *et al.*, 1989] : dans ces langages, les procédures étaient représentées par des *attachements procéduraux*, appelés plus justement des *réflexes*, et déclenchés lors de certains accès en lecture ou en écriture aux attributs des objets. C'était la programmation *dirigée par les accès* ou *orientée donnée*. Les *frames* sont traditionnellement décrit(e)s par un double niveau attribut-facette, les FACETTES décrivant le DOMAINE de l'attribut par des contraintes sur les valeurs possibles, des RÉFLEXES, des valeurs (par défaut) ou toute autre sémantique possible [Faucher, 1992].

Historiquement, les *frames* sont souvent assimilés à des systèmes de PROTOTYPES, en partie à cause des idées originelles de Minski, en partie à cause de certaines réalisations ([Winston et Horn, 1984] par exemple), mais souvent à tort. SHIRKA et FROME sont ainsi des langages de frames à distinction CLASSE / INSTANCE. D'un autre côté, on trouve des systèmes comme KEE ou YAFOOL où la distinction entre classe et instance se fait sur la base d'un système de prototypes.

Le terme de "frame" a reçu plusieurs traductions en français, en particulier "schéma", qui est aussi utilisé par les bases de données. Aucune ne s'est imposée.

[17] Les LT qualifient d'ailleurs fréquemment leur sémantique de *model-theoretic* ou de Tarskienne, alors que les théories d'héritage non monotone, par exemple, se revendiquent comme *proof-theoretic* [Thomason, 1992 ; Horty, 1994]. Il semble, comme le remarque Quine, que ce soit la théorie des MODÈLES qui soit, à l'heure actuelle, politiquement correcte.

Cet usage de "frame" n'a bien sûr rien à voir avec le fameux "problème du cadre" (*frame problem*) qui concerne la stabilité de la représentation de l'environnement lorsqu'une action est effectuée : si je change de cravate, mon état civil, mon adresse, ma famille et mon travail ne changent pas forcément et il est a priori inutile de les réaffirmer. Deux seules exceptions connues au problème du cadre : le nez de Cléopâtre et les personnages de Philip K. Dick.

(F)Rome

Œuvre(s) de Bernard Carré et Lenneke Dekker et de leur équipe du LIFL à Lille. Sur la base d'un LPO avec POINTS DE VUE et évolution d'objets — ROME [Carré, 1989 ; Carré et Geib, 1990 ; Carré *et al.*, 1990] —, un langage de *frames* défini par MÉTA-programmation, avec FILTRAGE et CLASSIFICATION — FROME [Carré et Dekker, 1991 ; Dekker et Carré, 1992 ; Dekker, 1993 ; Dekker, 1994]. Tendance œcuméniste, libertaire pour la RCO mais plus sécuritaire pour le LPO.

héritage

Le mécanisme de partage de propriétés entre classes qui se spécialisent. On distingue deux niveaux d'héritage [Ducournau *et al.*, 1995]. L'*héritage de nom* se traduit par une inclusion d'ensembles de propriétés : la sous-classe possède toutes les propriétés de la super-classe. L'*héritage de valeur* est le mécanisme déductif qui recherche la valeur héritée d'une propriété héritée. L'héritage de valeur est en général régi par le *masquage* associé à une redéfinition de la propriété dans une sous-classe, mais ce masquage peut — doit — être interdit ou limité dans des cas spécifiques.

L'héritage peut être *simple* ou *multiple* suivant qu'une classe peut avoir une seule super-classe ou plusieurs. Voir HÉRITAGE MULTIPLE.

Le terme "héritage" est souvent utilisé pour désigner un partage de propriété entre objets ou classes, sans que ce partage suive la relation de spécialisation entre classes. En général cet usage est abusif : le terme de partage a une sémantique réflexive de relation d'équivalence, alors que l'héritage a une sémantique de relation d'ordre, sémantiques qu'il faut essayer de conserver. Lorsque l'on dit que l'héritage est un partage de propriétés, il faut comprendre que toutes les sous-classes (et instances) partagent les propriétés qu'elles héritent. L'héritage qui est souvent associé à la composition [Dugerdil, 1988] doit être qualifié autrement : "partage", "propagation", "diffusion", etc. mais pas "héritage" [Magnan, 1994]. Il arrive que l'on pratique de l'héritage sur des relations d'ordre qui ne sont pas la relation de spécialisation entre classes, mais dans ce cas, les entités qui supportent cet héritage semblent avoir une sémantique de classe et la relation une sémantique de spécialisation. Ainsi [Napoli *et al.*, 1994] décrit une utilisation de l'héritage sur un graphe d'inclusion de structures moléculaires, mais chaque structure peut-être interpétée comme la "classe des molécules qui contiennent la structure", et l'inclusion de structures devient alors une généralisation entre ces classes de molécules. [Ducournau *et al.*, 1992 ; Ducournau *et al.*, 1995]

Le terme "héritage" est aussi employé pour parler d'inclusion d'environnements dans les langages de programmation [Mulet et Marco, 1994] ou d'inclusion de "mondes" dans une approche de type *assumption based truth maintenance system* (ATMS) [Finin et McGuire, 1991]. Dans les deux cas, l'interprétation en termes de classe est d'autant plus difficile que l'analogie avec les prototypes est assez forte.

héritage multiple

L'héritage multiple introduit des problèmes de conflit de nom et de valeur. Les premiers sont syntaxiques : ils sont dûs à un phénomène d'homonymie de propriétés et doivent se résoudre syntaxiquement dans le(s) contexte(s) où le nom considéré est équivoque. Voir HOMONYMES, ÉQUIVOQUE, UNIVOQUE. Les seconds sont plus sémantiques et nécessitent de recourir,

- soit à la sémantique spécifique de la propriété considérée, dans le cas où cette sémantique autorise une résolution correcte : il faut en particulier que l'ensemble de valeur de la propriété ait une structure de demi-treillis complet ;
- soit à un mécanisme heuristique, uniforme et par défaut, qui choisit une valeur parmi les valeurs en conflit : le seul mécanisme connu est la LINÉARISATION.

La technique des POINTS DE VUE est souvent présentée comme une solution au(x) problème(s) de l'héritage multiple [Dugerdil, 1988 ; Mariño, 1993 ; Dekker, 1994]. Mais dire cela ne fait que remplacer un problème ou un mot par un autre.

héritage non monotone

L'héritage non monotone désigne habituellement les théories d'héritage avec EXCEPTIONS dans des réseaux sémantiques réduits à leur plus simple expression : nœuds et relations d'héritage [Thomason, 1992 ; Horty, 1994]. Comme le *masquage* de l'héritage de propriétés des objets rend aussi l'héritage non monotone, le terme est ambigü et il paraît préférable d'utiliser *héritage non transitif* pour désigner ces théories où la relation de généralisation n'est plus transitive [Ducournau *et al.*,]. [Simonet, 1994] montre comment peut se concevoir l'équivalence entre ces deux théories d'héritage. Le seul exemple connu de système à objets proposant un héritage non transitif est YAFOOL [Ducournau et Habib, 1989 ; Ducournau et Habib, 1991].

hiérarchie

On conviendra d'utiliser ce terme à la place de CLASSIFICATION pris dans le sens d'ensemble ordonné de classes, pour le distinguer de son usage comme processus d'insertion d'une classe ou d'une instance. Une hiérarchie est un ordre partiel, éventuellement un TREILLIS ou une arborescence, voire une TAXINOMIE au sens strict.

homonymes (propriétés)

Se dit de deux propriétés distinctes de même nom. Des propriétés homonymes peuvent devenir équivoques dans certains contextes, en particulier dans l'HÉRITAGE, ce qui amène alors à un conflit de nom.

La technique des POINTS DE VUE est souvent — mais pas toujours, voir TROPES — présentée comme une solution au traitement des propriétés homonymes : elle permettrait de considérer des `étudiant-salarié` ayant deux attributs `département`, distincts bien qu'homonymes, en tant qu'`étudiant` et en tant que `salarié`.

L'homonymie de propriété n'est pas toujours possible : les LT — et plus généralement les systèmes qui proposent un mécanisme de classification, à l'exception notable de [Dekker, 1994] qui essaie de traiter le problème — la refusent généralement. Voir IDENTITÉ RÔLE, PRIMITIF.

identité

> [...] dire d'une chose quelconque qu'elle est identique à elle-même est évidemment trivial. Dès lors, comme le faisait remarquer Wittgenstein, à quoi bon parler d'identité ?
>
> W.V. Quine, "Identité" in [Quine, 1993]

En mécanique QUANTIQUE, le *principe d'exclusion de Pauli* impose que deux systèmes distincts diffèrent par au moins une de leurs valeurs propres. De même, l'ONTOLOGIE de Bunge (d'après [Wand, 1989]) impose à deux objets distincts de différer par la valeur d'au moins un de leurs attributs.

La *clé* de TROPES (et plus généralement des bases de données) est un succédané de ce principe : c'est un sous-ensemble d'attributs qui est supposé suffisant pour distinguer 2 objets du même type et donc pour les désigner de façon univoque. Dans la RÉALITÉ, les objets peuvent avoir plusieurs clés différentes — donc plusieurs désignations possibles et plusieurs contraintes de différence à respecter — et le postulat de Bunge-Wand signifie simplement qu'il en existe au moins une pour tout objet.

Le principe d'exclusion s'applique aussi aux propriétés : deux propriétés distinctes doivent différer soit par leur nom, soit, si elles sont HOMONYMES, par leur classe de DÉFINITION : les deux constituent leur *clé*. On a vu aussi des langages dont la clé des relations est donnée par la classe de définition et le DOMAINE de la relation : un objet ne peut pas avoir deux relations différentes avec des objets d'un certain type. Par exemple, avoir une valise dans chaque main. En revanche, un carton à chapeau et un cabas ... SHIRKA ou SMECI ? [Magnan, 1994].

instance

Désigne les entités individuelles, dans le modèle classe-instance qui distingue entités générique et entités individuelles. Contrairement aux apparences (Larousse) son usage en français est correct [Lalande, 1926] et les traducteurs de [Quine, 1993] ne le craignent pas. Dans ce cadre, instance est synonyme d'OBJET, mais on l'utilise surtout pour marquer sa relation avec la classe.

On rencontre aussi l'expression "instance terminale" pour désigner une instance. On voit bien l'origine de cette expression dans un graphe d'héritage mélangeant les relations de spécialisation et d'instanciation, comme dans certains langages à la limite des classes et des prototypes (KEE ou YAFOOL). Dans un modèle RÉFLEXIF, on peut utiliser cette expression pour désigner une instance qui n'est pas une classe, donc pour un "objet ordinaire", à condition que ces objets ordinaires puissent avoir des propriétés que n'auraient pas les classes, par exemple avoir une interprétation dans la RÉALITÉ. Si ce n'est pas le cas, la précision paraît complètement superflue.

instance directe

On utilisera ce terme pour désigner les instances d'une classe qui ne sont instances d'aucune de ses sous-classes. On pourrait aussi dire *instance propre*. Voir INSTANCE, TAXINOMIE.

instanciation

Dans le modèle objet traditionnel, un objet est instance directe d'une seule classe : il s'agit de "mono-instanciation". Voir MULTI-INSTANCIATION. La relation d'instanciation lie un objet à sa classe : le premier est INSTANCE DIRECTE de la seconde. Le terme est aussi utilisé pour désigner le processus de création d'une instance.

Dans les CONTRAINTES (ou la programmation logique), "instanciation" désigne la valuation d'une ou plusieurs variables par des valeurs du domaine.

intégration

Il s'agit de l'intégration de la RPO dans un LPO. L'intégration peut se caractériser, par opposition à l'implémentation, par les points suivants :

1. les objets (resp. classes) de la RCO sont un sous-ensemble des objets (resp. classes) du LPO ; les relations d'instanciation et de spécialisation de la RCO sont un sous-ensemble de celles du LPO ; les attributs de la RCO sont un sous-ensemble des attributs (du LPO) des objets de la RCO (c'est-à-dire que les objets de la RCO peuvent avoir des attributs extérieurs à la RCO) ;
2. les objets du LPO — étendus aux types de base — constituent le domaine général de valeurs des attributs de la RCO ; on réservera *relation* pour les attributs de la RCO qui prennent leurs valeurs dans les objets de la RCO ;
3. les fonctionnalités du LPO sont applicables aux objets (resp. classes) de la RCO *dans des limites à préciser*, c'est-à-dire tant qu'elles préservent la sémantique de la RCO.

Les points 1 et 2 énoncent le fait que la RCO est une restriction ensembliste du LPO. C'est bien sûr la dernière clause qui pose le plus problème et qui empêche que la RCO, considérée comme telle (et non comme une application du LPO), ne soit une simple extension du LPO.

intension

Pris dans le sens de "compréhension", comme dans "en compréhension", "intension" désigne la définition d'un concept par ses PROPRIÉTÉS nécessaires et/ou suffisantes et s'oppose à EXTENSION qui désigne l'ensemble des individus qui tombent sous ce concept. Voir RÔLE.

intention

La communauté francophone a certaines difficultés avec les mots *intenTion* et *intenSion*, ce dernier pris avec le sens de *compréhension*, par opposition à *extension*. Au premier abord, *intension* ne paraît pas français (il est absent du Larousse par exemple), mais on le trouve dans [Lalande, 1926] où il est décrit comme ancien — citations à l'appui de Leibniz — avec le sens de *compréhension*. Il est donc normal d'utiliser *intension* ou le néologisme *intensionnel*. Il est par contre incohérent d'utiliser en anglais *intenTional* sous prétexte que *intenSional*

n'est pas dans le dictionnaire (*Oxford Dictionary* par exemple). Par ailleurs, si l'on suppose que *intenSion* n'est pas français, mais, si on l'utilise, c'est un anglicisme et il faut bien sûr utiliser l'orthographe anglaise et non l'orthographe du mot français le plus proche ! Là, on n'est pas loin de l'argumentaire de la MARMITE...

Si vous aviez l'intention de regarder à INTENSION, cet article était vraiment fait pour vous !

intuition

L'intuition semble intervenir de plusieurs manières en Intelligence Artificielle. Les théories d'HÉRITAGE NON MONOTONE ont donné lieu à des échanges d'exemples de même structure formelle, c'est-à-dire isomorphes du point de vue de la théorie, avec des querelles byzantines pour savoir comment l'intuition permettait de les interpréter [Ducournau et Habib, 1991]. Cet usage de l'intuition est à proscrire, la discussion portant sur des éléments extérieurs à la représentation. Par contre, il paraît nécessaire de baser la déclarativité sur le caractère intuitif de la sémantique de la représentation. Voir DÉCLARATIF.

L-O

libertaire

> "Libre de toute contrainte" sonne à l'oreille comme un pléonasme, mais, en fait, la contrainte est bien une liberté de second ordre : elle libère du poids des décisions.
>
> W.V. Quine, "Liberté" in [Quine, 1993]

Avec quand même une tendance libertaire indéniable : LISP par exemple, par opposition à ADA, C ou PL1 par opposition à PASCAL, ou YAFOOL par opposition à SHIRKA. Voir SÉCURITAIRE, AUTORITAIRE.

linéarisation

La linéarisation est un mécanisme de résolution des conflits d'héritage de valeur *uniforme* et *par défaut*, c'est-à-dire qu'il s'applique uniformément à toute propriété pour laquelle un mécanisme spécifique présumé correct n'a pas été défini. La linéarisation consiste à construire un ordre total sur les super-classes d'une classe, ordre que l'on suit pour chercher la propriété concernée. Pour respecter le masquage, il faut et il suffit que cet ordre soit une *extension linéaire* de la relation d'héritage [Ducournau et Habib, 1989 ; Ducournau *et al.*, 1995].

Il apparaît de plus que la linéarisation doit être *monotone*, c'est-à-dire que la linéarisation de la sous-classe contient — au sens ensembliste — les linéarisations des super-classes, pour que la résolution soit elle-même monotone, c'est-à-dire pour que la classe se comporte comme au moins l'une de ses super-classes pour toute propriété qu'elle ne redéfinit pas [Ducournau *et al.*, 1995 ; Ducournau *et al.*, 1994].

La linéarisation est principalement utilisée par CLOS et, plus près de nous, par YAFOOL : dans les deux cas, c'est bien une extension linéaire, mais elle n'est pas monotone.

LPO (langage de programmation à objets)

Outre le modèle objet commun, les LPO se caractérisent par l'accent mis sur les TYPES et le TYPAGE, sur le POLYMORPHISME et l'ENCAPSULATION. Voir [Stroustrup, 1987 ; Wegner, 1990].

LT (logique terminologique)

Désigne tout le courant des systèmes de représentation des connaissances issus de KL-ONE, et, au-delà, des langages de *frames* et des réseaux sémantiques [Woods et Schmolze, 1992]. On dit aussi "logique de description" (LD). La trait distinctif des LT réside dans le calcul automatique de la SUBSUMPTION entre CONCEPTS.

En emploiera en général LT en opposition à RCO, et a fortiori à LPO.

marmite

Claude Levi-Strauss rapporte cet exemple succulent de DIALECTIQUE : *1) mais non, je ne t'ai pas rendue une marmite percée, 2) d'ailleurs, elle était déjà percée quand tu me l'as prêtée, 3) de toute façon, tu ne m'as jamais prêté de marmite.*

On se prend à rêver que c'est dans cette marmite qu'aurait dû rôtir le missionnaire du paradoxe (condamné à être mangé, il a le choix du mode de cuisson : suivant qu'il dit la vérité ou non, il sera bouilli ou rôti).

MCO (méthode de conception objet)

Il s'agit, en toute généralité, de toutes les méthodes basées sur un modèle objet, et permettant de guider un projet informatique, pour passer de la RÉALITÉ à un PROGRAMME opérationnel,

à travers des phases nommées diversement : ANALYSE, ACQUISITION des CONNAISSANCES, MODÉLISATION, CONCEPTION, REPRÉSENTATION, sans oublier l'implémentation ou la validation.

message (envoi de)

La métaphore de l'envoi de message vient de SIMULA [Birtwistle *et al.*, 1973], où les objets étaient des *coroutines* dont l'asynchronisme justifiait la métaphore. Reprise par SMALLTALK [Goldberg et Robson, 1983], la métaphore a perdu tout son sens en même temps que les objets perdaient leur asynchonisme. Les acteurs et autres objets concurrents la justifient à nouveau. Voir MÉTHODE.

La métaphore de l'envoi de message est manifestement inapplicable aux *multi-méthodes*, ou fonctions génériques, dont la sélection se fait sur plus d'un argument, comme c'est le cas en CLOS : il n'y a plus alors de notion de destinataire ou de récepteur du message.

méta (niveau)

Se dit en général de la partie d'un système qui décrit ou implémente ce système. Le système peut être qualifié de réflexif si ce niveau méta est décrit dans les termes du système lui-même. Dans les systèmes à objets, ce niveau est en général constitué des méta-classes et de divers méta-objets, en particulier pour les propriétés (attributs ou méthodes). Le niveau méta est le lieu de la *méta-programmation* ou du *meta-object protocol* [Kiczales *et al.*, 1991] grâce auxquels [Rathke, 1991 ; Dekker, 1994] ont implémenté des langages de frame au-dessus d'un LPO. Représenter les diverses procédures — méthodes par exemple — comme des objets est ainsi une solution possible à la représentation des procédures et à leur intégration dans un cadre déclaratif. Voir RÉFLEXIF, RÉIFICATION, MÉTA-CLASSE, RUSSEL.

méta-classe

Ce sont les classes dont les instances sont des classes. Un élément important du débat concerne le niveau méta des LPO, qui effraie les *intégristes pessimistes* sur trois plans distincts :

1. Sur le plan sémantique, la circularité de l'instanciation de certaines métaclasses — comme en CLOS ou OBJVLISP [Cointe, 1987] — amène à considérer un ensemble mal fondé (qui se contient lui-même) qui fait craindre un paradoxe de Russel.
2. Sur le plan de la représentation, il y a un mélange des objets qui ont une interprétation dans le monde réel et des objets qui ne s'interprètent que dans le programme lui-même.
3. Sur le plan de la sécurité, il n'y aurait aucune garantie de permanence de la sémantique du système à partir du moment où *n'importe qui* (sic) peut se permettre de modifier des choses aussi fondamentales que l'instanciation.

Les deux derniers points ne nécessitent bien sûr pas d'avoir recours au niveau méta : il suffit de considérer, d'une part, tous les objets du système qui servent par exemple aux interfaces graphiques (fenêtres ou menus), d'autre part les méthodes des classes qui peuvent être soit redéfinies, soit ajoutées par combinaison des méthodes en CLOS.

Sur les trois craintes exprimées par les *intégristes pessimistes*, les *œcuménistes optimistes* répondent :

1. Il n'y a pas lieu de craindre de paradoxe de Russel, car les classes sont en correspondance avec des ensembles sans en être vraiment. En particulier, n'importe quel ensemble n'est pas représenté par une classe[18]. L'ensemble des classes qui sont (resp. ne sont pas) leur propre instance n'est absolument pas paradoxal. Voir BOOTSTRAP.
2. La sémantique de représentation peut très bien ne s'appliquer qu'à un sous-ensemble des objets (classes), par exemple aux descendants d'une racine de la représentation. A la limite, on retourne dialectiquement l'argument en étendant le domaine modélisé aux objets informatiques — par exemple les fenêtres et menus à l'écran : on peut même avancer qu'il s'agit là de la seule partie de la représentation qui soit absolument correcte,

[18] L'argument doit être mieux étudié dans le cadre des LT dont l'ensemble de termes — c'est-à-dire l'univers du discours — est beaucoup plus vaste que l'ensemble des classes des objets (LPO et RCO).

par construction, puisque le domaine modélisé ne pré-existe pas à l'exécution du système, ce qui n'est bien sûr pas vrai pour le programme de paie de l'INRIA[19].

3. Le problème se pose plus généralement pour tous les langages d'implémentation dynamiques, LISP en premier. Il faut forcément faire confiance à l'UTILISATEUR.

Bien au contraire, les *œcuménistes* un tantinet explorateurs rêvent d'amener les classes de la RCO au cœur de la circularité : ils soutiennent qu'il n'y a pas de raison d'exclure a priori du domaine représenté les concepts qui servent à le représenter. Voir RUSSEL.

métaphore

La programmation est riche en métaphores. Les LPO sont basés, souvent abusivement, sur celle de l'ENVOI DE MESSAGE. Les langages de frames utilisent celle du RÉFLEXE. On peut citer aussi la métaphore du *tableau noir* qui prolonge la démarche des règles de PRODUCTION, ou celle des DÉMONS de UNIX..

[Gensel et Girard, 1992] propose une métaphore à base de *tâches* dont la principale originalité repose sur son usage de la hiérarchie d'héritage. Au lieu de chercher, par héritage donc en montant, la méthode à activer, elle recherche la tâche par classification, en descendant, les arguments de la tâche servant de filtre dont le classement détermine la tâche à activer. Dans les deux cas, il s'agit de rechercher un minimal dans la hiérarchie d'héritage. La sélection des tâches par classification s'apparente à la sélection des *multi-méthodes*. Voir PROCÉDURES, MÉTHODE.

Sur la métaphore et la métonymie, et sur leurs relations métaphoronymiques, voir [Tort, 1989].

méthode

Les procédures des LPO sont les *méthodes* qui sont activées par le mécanisme appelé, de façon impropre, ENVOI DE MESSAGE : il est possible par ce biais d'attacher un comportement spécifique à chaque classe, avec possibilité d'une part d'hériter, d'autre part de masquer.

La seule chose qui distingue une méthode d'une fonction ou procédure classique, c'est le choix de la méthode à déclencher, mais son appel lui-même reste impératif. Dans certains cas, la sélection se fait sur plus d'un argument, comme en CLOS : on les appelle alors des *multi-méthodes*.

D'un strict point de vue du vocabulaire, il faut distinguer la méthode en tant que fonction générique globale, et la méthode en tant que fonction locale attachée à une classe particulière pour implémenter, pour cette classe, la fonction générique. CLOS utilise les termes respectifs de "fonction générique" et de "méthode". Certains langages utilisent "sélecteur" (ou "message") et "méthode" : mais cela suppose que l'espace de nom des propriétés soit global et non pas par classe (avec héritage) comme le prône l'*héritage de nom*. Voir HÉRITAGE. En fait, on a bien trois notions distinctes : la propriété globale, son nom, et les fonctions qui l'incarnent dans chaque classe et qu'il faut interpréter comme des valeurs fonctionnelles.

Enfin, "méthode" s'emploie toujours au sens vulgaire : les méthodes constituent ainsi une méthode comme une autre de programmation impérative. C'est dans ce sens qu'il faut entendre les méthodes d'ANALYSE ou de CONCEPTION.

modèle

> Récemment, la tendance était de sacrifier la simplicité sur l'autel de la théorie des modèles.
>
> W.V. Quine, "Mathématose" in [Quine, 1993]

Le terme de modèle a deux sens qui peuvent être utilisés quasi simultanément. Le "modèle mathématique" (ou "modèle réduit") est une abstraction de la réalité. Le "modèle" d'une théorie, au sens de la *théorie des modèles* est une interprétation de cette théorie constituée par un domaine et une fonction faisant correspondre à tout élément de la théorie un élément du domaine vérifiant les propriétés prédites par la théorie et son interprétation. Voir MODÉLISATION.

[19] Seul le solipsiste le plus endurci peut prétendre que les auteurs du présent article ne pré-existent pas à leur bulletin de paie. Voir RÉALITÉ.

modélisation

La sémantique ontologique a été décrite un peu rapidement, et il faut préciser. On commence par construire un *modèle* (au sens de "modèle mathématique"), de la réalité, par abstraction et restriction, dont on construit alors une théorie — la représentation — dont ce modèle est un *modèle*, cette fois-ci au sens de la théorie des modèles. La fonction d'interprétation a pour co-domaine le MODÈLE de la réalité, et non la réalité elle-même. Sur la distinction entre modélisation et représentation, voir [Berthier, 1994]. Voir ACQUISITION, REPRÉSENTATION, ONTOLOGIQUE.

mono-valué

Un attribut est mono-valué si sa valeur est, pour chaque objet, élémentaire (non ensembliste). Vu comme une relation mathématique, l'attribut est une fonction (partielle). Voir aussi le traitement des relations mono et multi-valuées par [Caseau, 1991] — il traite une relation mono-valuée de façon multi-valuée, en considérant l'ensemble des valeurs possibles —, qui n'est pas très explicite sur la manière dont il obtient la cohérence pour ses relations mono-valuées. Voir RELATION, MULTI-VALUÉ.

La distinction mono/multi-valué n'est qu'un cas particulier des restrictions de cardinalité offertes par les LT et certqins RCO.

On peut voir les ATTRIBUTS comme des RELATIONS ou comme des FONCTIONS. Les deux sont équivalents, à condition de considérer des relations de $\mathcal{D} \times \mathcal{D}$ ou des fonctions de $\mathcal{D} \mapsto 2^{\mathcal{D}}$. Nebel remarque que ce n'est pas le cas pour les ψ-termes de [Aït-Kaci et Nasr, 1986] dont les attributs sont mono-valués, c'est-à-dire des fonctions de $\mathcal{D} \mapsto \mathcal{D}$. Nebel remarque aussi que le passage au multi-valué introduirait une grande complexité de calcul, comme c'est aussi le cas pour les CSP.

multi-instanciation

> Mais souvent tel objet, qui par une ou plusieurs de ses propriétés a été placé dans une classe, tient à une autre classe par d'autres propriétés [...].
>
> d'Alembert, *Discours préliminaire* à l'Encyclopédie, cité par [Eco, 1994].

C'est une notion atypique qui permet à un objet d'être INSTANCE DIRECTE de plusieurs classes incomparables. On peut citer (F)ROME [Carré, 1989 ; Dekker, 1994], TROPES [Mariño, 1993] ou SHOOD [Rieu *et al.*, 1992]. La multi-instanciation est en général associée aux notions de POINTS DE VUE, chaque classe décrivant plus ou moins un point de vue sur l'instance.

De façon générale, la multi-instanciation impose de prendre en compte, en plus de la relation de généralisation / spécialisation une relation de compatibilité (ou d'incompatibilité) entre classes exprimant le caractère DISJOINT de leurs extensions.

multi-valué

Un attribut est multi-valué si sa valeur est, pour chaque objet, ensembliste. Vu comme une relation mathématique, l'attribut n'est pas une fonction. Voir RELATION, MONO-VALUÉ.

Si l'on en croit [Girod, 1991], les attributs multi-valués semblent poser des problèmes extra-ordinaires aux LPO ordinaires et typés, qui ne savent pas exprimer le type de leurs valeurs par un DOMAINE élémentaire mais doivent définir explicitement un type `list(machin)` par "POLYMORPHISME paramétrique" [Cardelli et Wegner, 1985 ; Milner, 1978].

naturelle (sémantique)

> Nous avons une pente à parler d'objets et à penser à des objets.
>
> W.V. Quine, *Parler d'objets*, in [Quine, 1977]

Les objets bénéficient d'un caractère apparemment "naturel" dû à leur sémantique ontologique. Les notions de classes et d'héritage de propriétés viennent en effet tout droit de la syllogistique aristotélicienne (Socrate, homme, mortel) dont notre culture a fini par nous convaincre du caractère naturel.

Le terme de *sémantique naturelle* est aussi utilisé pour désigner une approche de la sémantique des langages de programmation issue de la *déduction naturelle* [Kahn, 1987 ; Borgida, 1992].

nécessaire

Les ATTRIBUTS décrivent en général des conditions nécessaires — portant sur leur DOMAINE —, au sens où toute instance d'une classe doit vérifier les conditions décrites par les attributs de la classe. C'est ce caractère nécessaire qui ouvre la possibilité d'un HÉRITAGE déductif, à partir du moment où l'on sait qu'un objet est instance directe d'une classe. Cette nécessité peut être relativisée par des EXCEPTIONS, bien qu'elles portent plutôt sur des valeurs par défaut qui ne sont pas considérées comme nécessaires. Ces conditions nécessaires ne sont en général pas SUFFISANTES pour assurer l'appartenance d'un objet à l'extension d'une classe si cette appartenance n'a pas été explicitement assertée.

newtonnienne (mécanique)

La mécanique newtonienne offre un exemple simple de système presque DÉCOMPOSABLE : le problème de Hilbert des trois corps en est presque une caricature. Avec deux corps, la mécanique s'énonce "déclarativement", mais avec trois, c'est déjà chaotique ! Cela n'empêche cependant pas de calculer les tables des marées pour le siècle suivant. La RCO peut-elle échapper au KO ? Voir DÉCLARATIF.

nil

L'être ou le néant, voilà le problème.
Raymond Queneau, *Zazie dans le métro.*

objective (propriété)

Une propriété d'un système informatique est objective lorsqu'elle est maintenue, par construction, par le système lui-même et non par la discipline du programmeur.

objet

Eu égard à la difficulté de transcender notre patron de pensée orienté vers l'objet, il convient plutôt de l'examiner de l'intérieur.
W.V. Quine, *Parler d'objets*, in [Quine, 1977]

Désigne à la fois le paradigme général, plus spécifiquement le couple LPO-RCO (donc tout sauf les LT), ou enfin, très spécifiquement les LPO. "Objet" est aussi bien applicable aux prototypes qu'aux systèmes basés sur la distinction classe-instance : dans ce dernier cas, au sens strict, un objet est une INSTANCE (c'est-à-dire que, dans un système non RÉFLEXIF, une CLASSE n'est pas, au sens strict, un objet). Voir RÉIFICATION.

Dans une collision frontale avec le paradigme objet, Common Lisp [Steele, 1990] a popularisé la notion d'*objet de première classe* qui désigne les entités d'un langage de programmation qui sont explicitement représentées pendant l'exécution, par une structure de donnée effectivement manipulable par le programme(ur). Dans un langage comme C++, une classe n'est pas un objet de première classe, encore moins un objet. A part dans cette expression, nous réserverons le terme "objet" au paradigme objet et nous emploierons "entité" pour désigner les objets qui n'en sont pas, au sens des objets du moins. On trouve aussi du *code-objet*, ou même du *langage-objet* qui est l'objet d'un méta-langage (Tarski).

Objlog

Langage de FRAME avec POINTS DE VUE implémenté au-dessus de PROLOG par P. Dugerdil à Marseille [Dugerdil, 1988 ; Dugerdil, 1991]. Un OBJLOG II a vu le jour [Faucher, 1992]. Voir BDO.

occurrence

Il est rare que l'on distingue entre une "occurrence" et une instance et, cependant, une occurrence n'est ni une instance ni un type — plutôt un objet intermédiaire [...] Les occurrences d'instances sont des instances et les occurrences de type [...] des types.
W.V Quine, "Type *versus* instance" in [Quine, 1993]

[Meyer, 1991] utilise ce terme dans le sens d'instance (dans la traduction française, page 100), ce qui peut s'expliquer de la façon suivante. En anglais, *instance* a les deux sens d'exemple et d'occasion ou circonstance. C'est le premier qui est utilisé dans la problématique objet : un objet est un exemple de sa classe. Et le Robert & Collins traduit la locution française "en

l'occurrence", où le mot est pris dans le sens de circonstance, par "in this instance"[20]. Pour être complet, on peut se demander si l'usage de INSTANCE en français est bien correct ou si c'est un anglicisme. Le Larrousse ne lui donne absolument pas ce sens là. Mais [Lalande, 1926] reprend aussi le sens anglais d'exemple, en citant (il est vrai) deux éminents philosophes français, Leibniz et Bacon.

En revanche, "occurrence" est parfait pour désigner les "instances temporelles" de YAFOOL, qui sont des sortes d'instances d'instances permettant de décrire l'évolution des objets dans le TEMPS.

ontologie

Le terme d'*ontologie* est utilisé depuis quelques années en Intelligence Artificielle, en général au pluriel, pour désigner des modélisations de portions de la RÉALITÉ, des "ontologies particulières" : l'espace, le temps, la composition, etc. [Lenat et Guha, 1990a; Lenat et Guha, 1990b]. Nous l'utilisons ici, au singulier, dans son sens originel de *partie de la philosophie qui spécule sur "l'être en tant qu'être" suivant le mot d'Aristote* (in [Lalande, 1926]).

L'*Encyclopædia Universalis* propose à *Ontologie* une petite discussion passionnante sur Frege, Russel et Wittgenstein où il est dit, en gros, que la représentation repose sur une INTENTION de dénoter quelque chose, une *intenTion d'extenSion*, en quelque sorte ... On peut aussi noter que si les philosophes s'intéressent de près à la SÉMANTIQUE du langage, à l'entrée *Sémantique*, les linguistes ne parlent pas d'*Ontologie* : ces deux entrées s'ignorent superbement.

On peut faire une synthèse osée de ces deux sens, en résumant la thèse de [Wand, 1989] par : *le modèle objet est l'ontologie de l'ontologie.*

ontologique (sémantique)

— Qu'est-ce que vous avez ? [...]
— Une ontalgie, répondit Thérèse. [...]
— Qu'est-ce que c'est que ça ?
— Une maladie essentielle, répondit Thérèse,
ça ressemble à de l'asthme mais c'est plus distingué.
Raymond Queneau, *Loin de Rueil*

Pour remédier à l'absence de SÉMANTIQUE claire des objets, [Wand, 1989] propose un modèle formel des objets orienté vers la modélisation applicative et basé sur l'ONTOLOGIE de [Bunge, 1977; Bunge, 1979].

D'après Wand, l'ontologie de Bunge est constituée d'*entités individuelles* munies de *propriétés*, et régies par des *lois*. L'ensemble repose sur quelques *postulats*, dont un équivalent du principe d'exclusion de Pauli et un principe à la Lavoisier — *rien ne se perd, rien ne se crée, tout se transforme* — que ne renierait ni un *garbage collector*, ni la logique linéaire. Voir IDENTITÉ, QUANTIQUE, TEMPS.

Wand traduit ces notions de base, respectivement, en *objets*, *attributs* et *contraintes* portant sur les attributs, ce qui lui donne d'abord un modèle d'objets non typés. Il définit ensuite les classes comme des ensembles de propriétés. Il s'agit donc clairement d'un cadre général de sémantique ontologique.

Curieusement, le modèle de Wand s'applique beaucoup mieux aux LT qu'aux LPO qui constituent son objectif : Wand modélise en effet les classes à partir d'ensembles de propriétés, très exactement comme les termes des LT sont construits à partir des RÔLES. C'est encore une variante de la poule et de l'œuf : qu'est-ce qui est premier de la classe ou des propriétés ? Voir aussi "classes *versus* propriétés" in [Quine, 1993].

Le reste du modèle de Wand ne nous intéresse pas ici : il modélise aussi la COMPOSITION mais il est en revanche beaucoup moins convaincant dans sa tentative de modéliser l'interaction entre objets — il se restreint au cas binaire en le prétendant suffisamment général, voir NEWTONNIENNE, RELATION BINAIRE — ou leur évolution. Ce qui nous intéresse ici est moins le modèle lui-même que son existence.

[20] La traduction n'est bien sûr pas symétrique, se serait trop simple !

Les *œcuménistes* considèrent en particulier que [c'est] l'existence de ce modèle naturel [qui] impose — ou en tout cas favorise — au programmeur-concepteur une étape préalable de représentation qui n'est pas si éloignée que ça de la RCO.

opérationnelle (sémantique)

Elle offre un modèle de fonctionnement d'un système sans référence à autre chose qu'au système lui-même, à la manière d'un algorithme ; dans ce cas, la "signification" d'un système se ramène à son comportement qui doit être prévisible.

Une "bonne" sémantique opérationnelle doit être "déterministe" non seulement pour le programme lui-même, mais aussi pour l'utilisateur. Beaucoup de systèmes n'offrent pourtant pas un comportement déterministe, dans la mesure où la documentation ne donne pas les moyens à l'utilisateur de connaître le déroulement exact du programme : c'est le cas des RÉFLEXES dans beaucoup de systèmes qui ne précisent pas l'ordre exact de leur déclenchement, ou de certains systèmes de CONTRAINTES, GARNET par exemple. Voir CSP. L'héritage multiple traité par les techniques de LINÉARISATION est un exemple de "bonne" sémantique opérationnelle [Ducournau et Habib, 1989 ; Ducournau *et al.*, 1995].

optimistes

Alphonse Allais les qualifiait d'imbéciles gais. Certains pensent qu'il est possible d'intégrer LPO et RCO. D'autres — ou les mêmes ? — pensent qu'il est possible d'atteindre le paradis de la représentation et du déclaratif [Aït-Kaci, 1991], une RCO parfaite, dans le monde des idéaux platoniciens, quelque part entre le dodécaèdre régulier et la qualité zéro défaut. Serait-ce encore une question de point de vue ? Voir PESSIMISTES.

P-R

paradigme

[Bobrow et Stefik, 1983] a popularisé le vocable de *paradigme de programmation* qui a été abondamment repris dans la littérature française. Parler de paradigme dans un dictionnaire "objet" n'est pas que de la cuistrerie d'érudit : sous ce mot se cache en fait les deux grandes interprétations des systèmes à objets, les CLASSES ou les PROTOTYPES. Tout dépend dans quelle langue : en anglais, *paradigm* représente un exemple typique, un prototype, alors qu'en français, où son usage est plus technique, linguistique surtout, il désigne une classe (d'équivalence).

perspective

Cette technique apparentée aux POINTS DE VUE consiste à considérer des objets comme composés de diverses perspectives distinctes. C'est sans doute la seule technique qui permette de traiter la question des points de vue en toute généralité, puisque rien n'empêche a priori d'avoir plusieurs perspectives de même type, ce qui est impossible avec les solutions proposées par (F)ROME, OBJLOG ou TROPES. Leurs objectifs respectifs sont cependant assez différents : on pourrait dire que TROPES s'intéresse à la CO-RÉFÉRENCE de classes, alors que les perspectives s'intéressent plutôt à la co-référence d'instances. Voir COMPOSITION, HOMONYMES, RÔLE.

pessimistes

Alphonse Allais les qualifiait d'imbéciles tristes. Certains pensent qu'il n'est pas possible ou souhaitable d'intégrer LPO et RCO. D'autres — ou les mêmes ? — pensent qu'il n'est pas possible[21] d'atteindre le paradis de la représentation et du déclaratif, en particulier que toute sémantique déclarative suffisament intuitive est triviale. Voir OPTIMISTES.

Platon

Dominique admettait aisément une chose :
qu'elle représentât un idéal.
[...] Elle platonisait à bloc.
Raymond Queneau, *Loin de Rueil*

D'un point de vue philosophique, la notion de CLASSE des systèmes à objets est un argument de poids dans la controverse entre Platon et Aristote [Kay, 1993]. Le premier croyait à l'existence des *idéaux*, avec d'ailleurs un sens plus proche des PROTOTYPES que des classes, alors que le second considérait qu'il ne s'agissait que d'abstractions sans réelle existence. Les objets ont bien tranché, puisque c'est la classe qui crée ses instances, et qui leur préexiste donc. Le cas des LT est moins clair, car il est souvent possible de définir une instance en définissant en même temps, implicitement, le concept sous lequel elle tombe : dans leur cas, ce sont en fait les RÔLES qui sont premiers.

points de vue

Outre le modèle objet fondamental, la RCO a plusieurs autres besoins de représentation, au premier rang desquels figure ce que l'on appelle fréquemment *points de vue* ou PERSPECTIVES

[21] On dirait une publicité SNCF.

[Dugerdil, 1988; Dugerdil, 1991; Carré, 1989; Carré et Geib, 1990; Mariño *et al.*, 1990; Mariño, 1993; Dekker, 1994; Rathke et Redmiles, 1993] et qui correspond plus ou moins à ce que Ferber et Volle [**?**; Volle, 1989; Ferber, 1983] avaient appelé CORÉFÉRENCE, aux FACETTES de [Perrot et Wolinski, 1992] ou aux *vues* des BDO.

Nous ne discuterons pas en détail de ces points de vue ici. Notons cependant que :

- ces notions touchent de près au cœur du modèle objet puisqu'elles mettent en jeu les descriptions des objets, c'est-à-dire les classes, aussi bien que la relation d'appartenance des objets aux classes : aussi bien (F)ROME [Carré, 1989; Dekker, 1994] que TROPES [Mariño, 1993] proposent un modèle objet atypique avec MULTI-INSTANCIATION, dans lequel la hiérarchie des classes ne correspond pas vraiment à l'ONTOLOGIE[22] [Euzenat, 1993b].
- ces besoins sont aussi présents en LPO : le cas de [Perrot et Wolinski, 1992] est à la frontière LPO-RCO[23], et on doit bien trouver l'expression de ces besoins dans les MCO;
- les choix de représentation ne sont certainement pas sans effet sur la sémantique ONTOLOGIQUE : par définition, la CORÉFÉRENCE consiste à rendre la fonction d'interprétation non injective.

polymorphisme

> Le français parlé a engendré pléthore de glissades sémantiques en usant de l'expédiant simple mais on ne peut plus efficace qui consiste [...]. Grâce à quoi [...] *rien*, à l'origine le *rem* latin, "chose", [suffit] pour [exprimer] "aucune chose".
>
> W.V Quine, "Glissade sémantique" in [Quine, 1993]

Au sens étymologique du terme, "polymorphisme" devrait s'appliquer à une entité pouvant prendre plusieurs formes : pervers polymorphe ou loup-garou. Appliqué à un langage, et en termes saussuriens : un *signifié* pour plusieurs *signifiants*, un sens pour plusieurs formes. Curieusement, l'informatique — depuis Strachey en 1967, d'après [Milner, 1978] — l'utilise en sens inverse : un *signifiant* pour plusieurs *signifiés*, une forme — c'est-à-dire une S-expression en LISP — pour plusieurs sens / significations / interprétations / évaluations. Bref, polymorphisme pour *polysémie*.

La notion de polymorphisme vient des langages de programmation "classiques" et sert à interpréter la notion de MÉTHODE ou d'envoi de MESSAGE, d'une façon qui n'est peut-être pas très adaptée. La RCO y est indifférente. Voir ENCAPSULATION.

primitif

Dans les LT, les concepts sont *définis* ou *primitifs*. Dans le premier cas, ils sont définis par des conditions NÉCESSAIRES et SUFFISANTES. Dans le second cas, ils sont simplement décrits par des conditions nécessaires (que toute instance vérifie par construction), mais pas suffisantes (pour assurer l'appartenance d'un objet à l'extension du concept si cette appartenance n'a pas été explicitement assertée). Le caractère primitif d'un concept exprime donc une certaine transcendance du concept vis-à-vis des autres.

Dans certains systèmes de LT, les concepts primitifs sont deux à deux DISJOINTS, mais c'est arbitraire : ces deux notions sont indépendantes.

On peut assimiler les classes de la RCO à des concepts primitifs : ce sont de simples descriptions en termes de conditions nécessaires. Mais cela interdit bien sûr toute possibilité de classification. Les *concepts* et les *classes* de TROPES correspondent respectivement aux concepts primitifs et définis des LT.

Un point n'est pas clair, concernant l'identité des rôles d'un concept primitif : dans quel mesure le rôle r du concept primitif C_1 est-il le même que celui du concept défini C_2 ou que celui du concept primitif C_3 ? Ces concepts sont-ils tous trois subsumés par le concept défini

[22] Dans un sens à préciser, hein Jérôme! Faut-il comprendre que l'ontologie s'occupe de la création des instances et la taxinomie de leur reclassement ? Mais l'ontologie se préoccupe non seulement du fait d'être, mais aussi de quoi être. Voir LT.

[23] Mais il faut se rappeler que le point de vue de (certains) *œcuménistes* est que la programmation par objet se fait sur une base de représentation — le LPO se fait sur une base de RCO —, quitte à ce que cette représentation soit en pratique moins expressive que ce que fait la RCO.

réduit au rôle r ? Il y a ici, semble-t-il, un moyen de régler la question de l'univocité des rôles en définissant leur IDENTITÉ par leur nom et leur concept primitif de DÉFINITION. Cela n'aurait aucune influence sur la généralité de la classification puisque la mention du primitif est indispensable pour qu'il soit reconnu.

procédural

C'est l'un des termes de la controverse fameuse. La RCO le considère de deux façons différentes : quelles procédures peut-on utiliser ? comment représenter les procédures ? Voir DÉCLARATIF.

[Pitrat, 1990] définit une connaissance déclarative comme une "connaissance séparée de son mode d'emploi". C'est évidemment une condition nécessaire à l'existence de méta-connaissance portant sur cette connaissance. Rien n'empêche cependant de représenter explicitement et déclarativement le mode d'emploi d'une PROCÉDURE. En fait, lorsque l'on parle de connaissance procédurale, il faut distinguer le caractère procédural du contenu de la connaissance, du caractère procédural de son mode d'emploi, c'est-à-dire des conditions de son déclenchement. Les RÉFLEXES ou les règles de PRODUCTION constituent ainsi deux façons peu procédurales de déclencher des procédures. Il est assez significatif que Pitrat parle de "connaissances déclaratives" plutôt que de "représentation déclarative des connaissances". En particulier, la plupart de ses exemples sont des phrases en français qui expriment des connaissances en quête de représentation.

procédures (représentation des)

Si les LPO se prêtent bien à implémenter un comportement, c'est sous une forme procédurale qu'il faudrait pouvoir intégrer à la représentation déclarative pour pouvoir lui appliquer les fonctionnalités requises des systèmes à base de connaissance comme l'explication du raisonnement etc.

La représentation des procédures est un sujet de recherches mal identifié mais très actif depuis fort longtemps. On peut citer par exemple :

- Les systèmes de PRODUCTION, qui sont considérés comme procéduraux[24] par [Winograd, 1975]. A la limite, toute la problématique des architectures de *tableau noir* se pose ce genre de question : c'est à la fois un problème de contrôle et un problème de représentation.
- Les systèmes de *tâches* de SMECI [ILO, 1991] ou de [Gensel et Girard, 1992] sont une façon déclarative de représenter les procédures.
- Le système expert DIVA [David et Krivine, 1987; David et Krivine, 1988] utilise des *tâches* dans un autre sens pour engendrer des explications lors d'envois de message : ces tâches sont générées au niveau méta par le LPO d'implémentation, LORE [Caseau, 1987].
- F. Rechenmann avait exposé lors du workshop *Trends in Knowledge Representation* à l'occasion du 25^{e}anniversaire de l'INRIA la façon dont il voyait la représentation des méthodes — où "méthode" ne doit pas être pris au sens des LPO — de calcul de valeurs manquantes [?].
- [Rieu *et al.*, 1992] propose de faire jouer aux méthodes — au sens LPO cette fois — un rôle de représentation et de relais entre connaissances déclaratives et procédurales.

[Kristensen *et al.*, 1987; Borgida, 1981] ont aussi proposé des sortes de classification de procédures ou de méthodes.

D'une façon générale, le but est de représenter la procédure comme une boite noire, complètement indépendante du reste du système (donc sans effet de bord sur le système lui-même) et dont le contrôle et la représentation sont à la charge d'un mécanisme à la sémantique bien spécifiée.

A leur échelle, les RÉFLEXES étaient une première tentative, modeste, de solution à ce problème. Voir OPÉRATIONNELLE.

production (règles ou systèmes de)

[Winograd, 1975] les considère comme une façon de structurer les PROCÉDURES[24]. C'est certainement l'une des tentatives les plus anciennes de représenter les procédures et de résoudre

[24] De façon d'ailleurs un peu étonnante, puisque les règles ont longtemps été l'archétype du déclaratif.

le problème du contrôle. Les architectures de *tableau noir* ont généralisé la problématique. Restreints aux relations binaires — c'est-à-dire aux triplets objet-attribut-valeur —, c'est une sorte d'équivalent des RÉFLEXES : ils ont une origine commune et plusieurs mécanismes de compilation ont été proposés [Caseau, 1989].

programmes

Dans un système à objets, les programmes sont en général constitués par la description des classes, alors que les instances sont de l'ordre des données.

Prolog

Que faut-il reprocher à PROLOG ? D'abord le *cut* dont l'usage procéduralise inéluctablement tout programme. Ensuite l'impossibilité d'y représenter des RELATIONS mono-valuées : toute relation n-aire peut-être considérée comme une fonction d'un produit cartésien de $(n-1)$ domaines dans le n-ième (la relation est alors MONO-VALUÉE) ou dans l'ensemble des parties du n-ième (la relation est alors MULTI-VALUÉE). PROLOG oblige à considérer l'ensemble des solutions — dans un sens non précisé : possibles ? ce qui ressemble au traitement des relations mono-valuées par [Caseau, 1991] — des relations mono-valuées, ce qui empêche de lui associer une sémantique (hors *cut*) claire. L'usage du *cut*, permet d'obtenir des relations mono-valuées, mais leur sémantique est basée sur l'ordre des clauses : la valeur unique devient la première valeur possible, ce qui est difficilement acceptable, en tout cas peu DÉCLARATIF. Voir CSP.

[MacGregor, 1991] montre aussi que Prolog n'a pas de capacité définitionnelle, mais au lieu d'être restreint à des CN comme le sont les systèmes à objets, il est en fait restreint à des CS. Ceci dit, l'exemple de MacGregor — il définit `grand-parent` comme composition de `parent` et remarque que Prolog ne sait pas conclure de l'existence d'un `grand-parent`, l'existence d'un `parent` — met en jeu la composition de relation pour laquelle il n'apparaît pas que les LT offrent *gratuitement* plus de capacités inférentielles.

propriété

Terme utilisé informellement pour désigner les propriétés au sens large d'un objet : il s'agit alors de l'union des ATTRIBUTS et des MÉTHODES. Dans ce cas, les propriétés ont un nom, une "sémantique" spécifique, un DOMAINE de valeur etc.

C'est la distinction claire entre la propriété et son nom — le fait que la première soit irréductible au second — qui est à la base de l'*héritage de nom* et qui mène naturellement à une perspective RÉFLEXIVE où les propriétés elles-mêmes sont des objets. Voir IDENTITÉ, HÉRITAGE, HOMONYMES, MÉTA.

Dans un cadre objet strict — c'est-à-dire en excluant les RÔLES des LT — les propriétés ont une (ou plusieurs) classe de DÉFINITION.

Plus généralement encore, on peut utiliser "propriété" pour désigner informellement tous les prédicats applicables à (ou qui sont vrais de) l'objet.

prototypes

Nous faisons ici délibérément l'impasse sur les systèmes de *prototypes* (par exemple [Myers *et al.*, 1990 ; Chambers et Ungar, 1992]). Deux raisons à cela : 1) la simplification du discours : il est déjà difficile de s'entendre sur le modèle classe-instance, sans qu'il soit nécessaire de compliquer le problème, 2) il est tout-à-fait probable que le modèle des prototypes se ramène bien à celui des classes, comme c'est le cas pour l'héritage par exemple [Ducournau et Habib, 1989 ; Ducournau *et al.*, 1995]. Voir CLASSE, INSTANCE. Les prototypes sont en général basés sur des théories psychologiques qui ne nous intéressent pas ici.

Le terme de prototype est utilisé de façon atypique dans divers systèmes : DIVA [David et Krivine, 1987 ; David et Krivine, 1988] ou SMECI [ILO, 1991] par exemple.

Le modèle usuel de prototypes est particulièrement simple, mais on peut envisager de le complexifier en l'intégrant à un modèle classe-instance, les instances pouvant être considérées comme des prototypes d'autres instances de la même classe ou d'une classe comparable (sous-classe). C'est un peu ce que fait YAFOOL avec ses "instances temporelles". Voir OCCURRENCE, TEMPS.

D'un point de vue formel, il faudrait étudier les rapports entre les systèmes àprototypes et la théorie axiomatique des ensembles (Zermelo-Fraenkel), dont les seules entités sont des ensembles, de la même manière que les seules entités des prototypes sont des prototypes. En revanche, dans le modèle classe-instance, même réflexif, il y a deux types d'entités, les ensembles et les éléments.

quantique (mécanique)

> [...] il se peut qu'il y ait un nombre indéfini ou infini d'atomes, mais il faut les répartir en un nombre limité et manipulable d'espèces, de telle sorte que les atomes de même espèce jouent un rôle identique à l'intérieur des lois de la théorie.
>
> W.V. Quine, "Atomes" in [Quine, 1993]

En mécanique quantique, un système (par exemple une particule) est représenté par un opérateur de Hilbert (pour simplifier une matrice) caractérisé par ses valeurs propres de la même façon qu'un objet est caractérisé par ses attributs. On a d'ailleurs un principe d'IDENTITÉ des objets de mêmes valeurs d'attributs (le principe d'exclusion de Pauli). Voir ONTOLOGIQUE.

En toute généralité, une matrice est particulièrement peu DÉCLARATIVE tant qu'elle n'est pas diagonalisée (c'est-à-dire tant que ses valeurs propres ne sont pas apparentes). Les opérations que l'on peut lui appliquer sont plus ou moins déclaratives, suivant l'état de la matrice (ou sa représentation) et suivant l'opération elle-même : une addition de matrice est très déclarative (parce que complètement DÉCOMPOSABLE), alors qu'une inversion ne l'est — décomposable et déclarative — que si la matrice est diagonale.

RCO (représentation des connaissances par objets)

La RCO présente une voie moyenne[25] entre les LPO d'une part, et les LT d'autre part. Elle partage avec les premiers le modèle objet commun, et avec les secondes le mécanisme de CLASSIFICATION et un même objectif de représentation des connaissances.

réalité

> Les théories qu'ils ont obtenues sont moins simples que l'on ne l'avait souhaité, mais, après tout, ce n'est pas leur faute, leur premier désir en tant que savants étant de coller à leurs données, qui s'obstinent à être ce qu'elles sont, et ils font ce qu'ils peuvent.
>
> W.V. Quine, "Atomes" in [Quine, 1993]

Peut-on se fier à un solipsiste pour la définir ? On parle aussi de "monde réel", voire de monde *réel* ^{TM} [Stein, 1992] ! Voir MODÉLISATION, ACQUISITION.

Le but de la représentation des connaissances est de représenter la façon dont nous concevons la réalité, et non pas la façon dont nous la nommons. En particulier, la représentation des connaissances s'occupe de représenter les *choses* et non pas les *noms* que nous utilisons pour désigner ces choses.

réductionniste (sémantique)

Elle consiste à établir une correspondance (une sorte de morphisme injectif) entre le système étudié et un autre système formel ou domaine mathématique, supposé plus fondamental et connu. C'est la technique utilisée en HÉRITAGE NON MONOTONE pour les théories dites *translationnelles* [Thomason, 1992 ; Ducournau *et al.*,], le domaine cible étant alors une logique non monotone. [Horty, 1994] note que, pour qu'une telle sémantique réductionniste soit déclarative, il faudrait que la traduction ait une propriété de *localité* — correspondant à l'indépendance du contexte ou modularité de DÉCLARATIF — qu'elle n'a pas : une petite modification du réseau d'héritage nécessite une complète recompilation. [Thomason, 1992] rajoute que, pour que cette sémantique soit intéressante, il faudrait que la logique cible apporte quelqu'éclairage nouveau sur le choix entre telle ou telle théorie, ce qui est loin d'être le cas.

On sait depuis longtemps [**?**] qu'il est possible de traduire un modèle objet simple en logique du premier ordre, ou mieux, du second ordre, si l'on veut traiter correctement l'héritage.

[25] Que ce soit en politique ou en jazz, les troisièmes voies (ou *third stream*) ont des difficultés d'existence et de reconnaissance.

[Abadi et Cardelli, 1994] utilise des fonctions comme domaine cible d'une sémantique des objets, mais c'est ce que [Meyer, 1990] appelle une sémantique DÉNOTATIONNELLE (dans un sens subtilement différent du nôtre).

réflexe

Procédure déclenchée automatiquement lors de certains accès aux attributs : calcul de valeur manquante à la lecture, vérification de valeurs et propagation des modifications à l'écriture. Il faut reconnaître aux réflexes le fait qu'ils constituent un mécanisme de déclenchement de procédures à la sémantique OPÉRATIONNELLE complètement spécifiée, au moins dans certains systèmes [Ducournau, 1991 ; Napoli, 1992] : un réflexe n'est en tout cas jamais appelé de façon impérative. Voir FRAME, MÉTHODE.

Quine sera sans doute d'accord avec nous — cf. "Glissade sémantique" à POLYMORPHISME — pour constater que, si "réflexif" paraît bien réfléchi, "réflexe" est tout ce qu'il y a de plus irréfléchi !

réflexif

> Le théorème de Gödel est apparenté aux paradoxes réflexifs. Sa démonstration revient à cajoler la notation de la théorie des nombres pour l'amener à se décrire elle-même.
>
> W.V. Quine, "Théorème de Gödel" in [Quine, 1993]

Se dit de tout système qui contient son propre niveau MÉTA, ou dont le niveau méta est décrit dans les termes mêmes du système. OBJVLISP [Cointe, 1987] est réflexif, mais SMALLTALK ne l'est qu'imparfaitement [Goldberg et Robson, 1983]. Voir MÉTA-CLASSE, RUSSEL.

réification

> Si on nous demande de penser à un objet, n'importe lequel, ce qui nous vient à l'esprit est quelque corps de taille moyenne. [...] On admet aussi [...] certains objets abstraits, ou *universaux* : propriétés, nombres, fonctions, classes.
>
> W.V. Quine, "Universaux" in [Quine, 1993]

Le processus consistant à transformer une notion abstraite en un OBJET. La réification est d'abord utilisée pour les notions de base des objets : classe, propriétés, etc., ce qui conduit en particulier à la RÉFLEXIVITÉ. Elle est utilisée aussi pour représenter les RELATIONS d'arité supérieure à 2, comme dans les réseaux sémantiques. Elle est enfin utilisée dans les applications, où les méthodes d'ANALYSE, de CONCEPTION ou de REPRÉSENTATION conduisent à considérer de nombreux objets "immatériels".

relation

Une relation se définit classiquement en mathématiques comme un sous-ensemble du produit cartésien de 1, 2 ou plusieurs ensembles. Les relations sont abondamment utilisées en informatique, en particulier dans les bases de données dites *relationnelles*. Voir BDO. Dans le cadre d'une approche objet, on appelle souvent *relation* un ATTRIBUT qui prend ses valeurs dans l'ensemble des objets. Il lui correspond alors de façon naturelle une relation binaire au sens mathématique du terme. Lorsque tout est objet, à la SMALLTALK ou à la LT, la distinction n'a plus de sens. Voir MONO-VALUÉ, MULTI-VALUÉ.

Il est souvent commode d'avoir accès aux relations inverses : leur définition avec maintien automatique de la symétrie est une fonctionnalité assez souvent offerte par les systèmes de RCO et les BDO, mais très rarement par les LPO. Pourtant cela ne dépasse pas ce que l'on peut exiger d'un bon langage de programmation.

relation binaire

Le modèle objet se prête bien à la représentation des relations binaires. Sa limitation fondamentale réside dans son incapacité à traiter directement des relations d'arité strictement supérieure à deux[26]. Il est nécessaire, soit de *réifier* ces relations, dans une approche tout à fait comparable à celle des RÉSEAUX SÉMANTIQUES ou des graphes conceptuels [Sowa, 1991] — si l'on veut rester dans un cadre pur-objet —, soit d'utiliser, conjointement avec les objets,

[26] Constat à mettre en parallèle avec le problème des trois corps. Après on s'étonne que des informaticiens aiment le jazz ou sachent compter jusqu'à trois. Voir DÉCOMPOSABLE, NEWTONIENNE.

un modèle relationnel, par exemple à base de règles comme le proposent bon nombre d'environnements de développement de systèmes à base de connaissances[27]. [Caseau, 1991] propose simultanément les deux techniques mais, comme il se restreint, même dans les règles, à des prédicats binaires, seule la première lui permet réellement de traiter les arités supérieures. Voir RELATION.

représentation

La différence entre représentation et MODÉLISATION est lumineusement expliquée par [Berthier, 1994]. Par contre, la représentation se distingue mieux de la programmation. Voir LPO, RCO.

réseau sémantique

Cette structure de représentation à base de graphe étiqueté peut être considérée comme l'ancêtre des objets, du côté de la représentation des connaissances du moins[28], avec les FRAMES et les LT [Sowa, 1991]. Les sommets du graphe sont des objets et les étiquettes des arcs, des relations. L'une de ces relations a une sémantique de spécialisation / généralisation. Dans la version dégénérée de l'HÉRITAGE NON MONOTONE il ne reste plus que cette dernière. Ce sont les graphes conceptuels qui semblent être la version actuelle la plus pure des réseaux sémantiques.

rôle

Le terme utilisé par les LT pour désigner les ATTRIBUTS (ou relations). La différence entre les rôles et les attributs est double. D'abord, les rôles décrivent des conditions NÉCESSAIRES et SUFFISANTES d'appartenance d'une instance à une classe, alors que les attributs ne décrivent que des conditions nécessaires. Ensuite, les rôles n'ont pas de classe (ou concept) de définition : c'est la raison pour laquelle les LT ne permettent pas l'homonymie de rôles. Voir PROPRIÉTÉ, HOMONYMES, IDENTITÉ.

Les LT ne distinguent en général pas des rôles MONO-VALUÉS ou MULTI-VALUÉS : ils sont tous multi-valués, avec des contraintes portant sur leur cardinalité. Un rôle mono-valué est ainsi considéré comme un cas particulier de rôle multi-valué. A la limite, ne pas avoir une propriété est un cas particulier de l'avoir de façon non nécessaire : il suffit d'imposer une cardinalité nulle. La cardinalité paraît offrir une généralisation parfaite de la distinction mono / multi-valué, mais ce n'est pas si simple : on perd ainsi le fait que certains attributs sont des fonctions et les CONTRAINTES sont rendues notoirement plus compliquées.

Les rôles des LT peuvent être en général être composés, (*role chains*), avoir un rôle inverse ou être hiérarchisés en sous-rôles, toutes choses s'exprimant bien dans l'algèbre relationnelle (au sens général du terme) sous-jacente.

Les raisons du choix historique du mot "rôle" — qui semble avoir le même sens en anglais et en français — en KL-ONE (me) sont inconnues et mystérieuses. N'eut été cet usage consacré, on aurait bien proposé de l'adopter à la place de PERSPECTIVE pour désigner la notion de POINT DE VUE lorsqu'elle est réalisée par COMPOSITION. Cette proposition repose sur l'analogie suivante : un individu a plusieurs "rôles" sociaux, dans son travail, ses études, sa famille et ses loisirs.

Russel

C'est — entre beaucoup d'autres choses — le fondateur de la théorie des TYPES qu'il a proposée pour tenter de résoudre le paradoxe dit "de Russel" des ENSEMBLES mal fondés qui se contiennent eux-mêmes [Russel, 1921]. Les modèles objets RÉFLEXIFS à la CLOS ou OBJVLISP [Cointe, 1987] semblent poser le même type de problème, puisqu'il existe au moins une classe — la première MÉTA-CLASSE — qui est instance d'elle-même. La réalité (informatique) même de ces méta-classes écarte cependant tout risque de paradoxe.

Il faudrait aussi envisager l'interprétation des systèmes à objets réflexifs (avec classe et méta-classes) dans une théorie des ensembles *non fondés* — c'est-à-dire qui ne contient pas l'*axiome*

[27] Objets et règles : encore un domaine que nous évitons soigneusement bien qu'il soit tout à fait honorable.

[28] Dans son historique de SMALLTALK, [Kay, 1993] se sent autant redevable à l'intelligence artificielle qu'à SIMULA.

de fondation qui assure plus ou moins que la relation d'appartenance est un *bon ordre* — et l'inclusion de ces systèmes réflexifs dans la problématique de la classification.

S-Z

sécuritaire

La tendance principale est quand même sécuritaire : les TYPES (abstraits), l'ENCAPSULATION, les redondances pour vérifier que le programmeur dit bien ce qu'il veut dire (ADA), mais là, on n'est pas loin du mépris. Voir LIBERTAIRE, AUTORITAIRE.

sémantique

> La sémantique n'est pas davantage un moyen astucieux de prouver qu'hormis l'auteur et ses amis, tous les autres disent des inepties.
> Alfred Tarski, *The semantic conception of truth...* [Tarski, 1952].
>
> La sémantique [...] est de nos jours si compliquée et le nom qui la distingue tellement ambivalent qu'il convient d'appliquer l'analyse sémantique au terme même de "sémantique" [...].
> Adam Schaff, Préface à *Introduction à la sémantique*, 1960 [Schaff, 1960]

La sémantique est théoriquement la science des significations, mais on utilise plutôt le terme, en informatique du moins, pour désigner un "système de signification". Ce premier usage de "sémantique" est générique : il s'applique à un modèle abstrait. Un deuxième usage est spécifique : "sémantique" désigne alors la signification d'une "instance" concrète, une application de ce modèle abstrait. On réservera "sémantique" pour le premier sens et on utilisera SIGNIFICATION pour le second.

En linguistique, mais on peut l'étendre sans difficulté à l'informatique, la sémantique est basée sur le triangle signifiant-signifié-référent (Saussure). Le signifiant est au niveau syntaxique des expressions et l'ONTOLOGIE s'occupe du référent et des cas difficiles tels que licorne, 30 février, milliardaire heureux, etc. D'un point de vue fonctionnel ou procédural, la référence pourrait être assimilée à l'évaluation, ou plutôt à son résultat, et la signification à quelque chose de l'ordre de l'exécution. La CORÉFÉRENCE ne rentre pas bien dans ce cadre, puisque l'évaluation du "vainqueur d'Iéna" va retourner l'objet qui représente le vainqueur d'Iéna, lequel est, comme chacun sait, co-référent à "Napoléon" (tous deux font référence à Napoléon), mais différent. On s'intéresse ici essentiellement aux sémantiques FORMELLES.

Shirka

Langage de *frames*, de type classe-instance, avec un fort accent mis sur le filtrage, puis sur la classification. Tendance intégriste forte [Rechenmann, 1988].

signification

On utilisera ce terme pour désigner la sémantique spécifique d'une application, par opposition à SÉMANTIQUE que l'on réserve pour le modèle de représentation qui l'implémente.

subjective (propriété)

Se dit d'une propriété non OBJECTIVE d'un système informatique, c'est-à-dire d'une propriété non assurée par le système lui-même, mais dont le maintien incombe au concepteur (par des actes de FOI) ou au programmeur, par sa discipline. Une propriété subjective traduit une INCOMPLÉTUDE syntaxique.

subsomption

Terme savant pour "généralisation", utilisé par les LT qui l'ont mis à la mode et se sont appelés un temps "langages de subsomption terminologique". La relation de subsomption désigne la relation entre une classe A plus générale et une classe B plus spécifique : A

subsume B. Son interprétation extensionnelle correspond à une inclusion de l'extension de B dans l'extension de A : $Ext(B) \subseteq Ext(A)$. En logique, la subsomption correspond à une implication matérielle : $(C \rightarrow D)$ parce que les conditions de vérité de C sont plus générales que les conditions de vérité de D.

On entend souvent "subsomption" comme CLASSIFICATION, au sens du mécanisme qui calcule la relation de subsomption lors de la définition d'un nouveau terme. Ce mécanisme de subsomption des LT se distingue de plusieurs façons de son équivalent de la RCO :
- il s'applique aussi bien aux classes qu'aux instances,
- il est implicite : la seule définition d'un TERME, voire son usage pour valuer un RÔLE, suffit pour le classer.

suffisantes

Dans certain cas, les attributs expriment des conditions non seulement NÉCESSAIRES, mais aussi suffisantes : tout objet les vérifiant est, de plein droit, une instance de la classe. C'est le cas des RÔLES des concepts définis — c'est-à-dire non PRIMITIFS — des LT, et c'est une condition plus ou moins nécessaire pour permettre la CLASSIFICATION. [Dekker, 1994] propose de décrire explicitement le caractère suffisant des attributs, dont la valeur, ou simplement la présence, peut être suffisante pour classer une instance. Dans tous les cas, l'ensemble des attributs exprimant une condition suffisante est nécessaire pour permettre la classification : "suffisant" s'applique à l'ensemble des conditions et non pas à chacune. Mais on pourrait imaginer avoir des sous-ensembles suffisants alternatifs, ce que fait TROPES dans une certaine mesure : un ensemble de conditions suffisantes est alors suffisant, mais non nécessaire, pour reconnaître un objet.

S'il y a de nombreux exemples de propriétés exprimant des conditions nécessaires mais pas suffisantes, l'inverse ne paraît pas exister (peut-être [Coupey et Fouqueré, 1994] ?). Ce serait pourtant une bonne façon d'exprimer la TYPICALITÉ.

surcharge

Ce terme désigne dans les langages de programmation le fait qu'un nom désigne des entités différentes qui sont distinguées contextuellement. Son usage en LPO est particulièrement confus, et il paraîtrait souhaitable de restreindre son usage à deux cas de figures : l'existence de propriétés HOMONYMES, en particulier quand cette surcharge peut subvenir dans le corps d'une même classe, comme en C++, où deux méthodes d'une même classe peuvent porter le même nom, à condition de se distinguer par leur signature.

Deux autres cas d'emploi de "surcharge" se rencontrent souvent mais sont à proscrire : le *masquage* de propriété n'est pas un cas de surcharge, puisque le nom désigne toujours la même propriété, seule sa VALEUR changeant. De même, le fait de pouvoir définir des propriétés de même nom dans des classes incomparables ne peut pas être considéré comme un cas de surcharge dans les langages qui ne permettent pas d'avoir des propriétés homonymes, en SMALLTALK par exemple. En effet, le fait que le même bout de code puisse s'appliquer aussi bien à l'une qu'à l'autre montre bien qu'il s'agit de la même propriété (principe d'*univocité* de [Ducournau *et al.*, 1995]).

synthèse

Le troisième temps : comment peut-on faire cohabiter LPO et RCO ? L'intégration de la seconde dans la première a-t-elle un sens ? Voir THÈSE, ANTITHÈSE, INTÉGRATION, MÉTA.

taxinomie

Synonyme de CLASSIFICATION dans son sens de structure hiérarchique. Comme HIÉRARCHIE ou "classification", "taxinomie" peut être entendu dans un sens très variable, du moins strict au plus strict : ensemble quelconque de classes partiellement ordonné, arborescence de classes, arborescence de classes dont seules les feuilles sont instanciables (les autres classes n'ont pas d'instances directes) [Euzenat, 1993a]. On appellera ce dernier cas une *taxinomie au sens strict*. Voir HÉRITAGE.

D'après [Lalande, 1926], "taxonomie" est une orthographe possible, que Littré trouve incorrecte. De plus, la signification de taxinomie serait celle de science de la classification et non

une classification particulière.

temps

Il était fatal de passer de l'*être* au *devenir* : la haute fidélité ontologique des objets amène rapidement à considérer leur évolution, donc à prendre en compte le temps. L'ontologie de [Bunge, 1977; Bunge, 1979] a ainsi un postulat à la Lavoisier sur l'évolution des objets. Sur le temps, l'informatique et la gastronomie, voir [Ringard, 1993].

Les langages qui entendent permettre aux objets d'évoluer n'offrent en général qu'une possibilité de reclassement ou migration des instances — c'est le cas de YAFOOL, CLOS, SMECI, (F)ROME et sans doute de beaucoup d'autres — sans réelle prise en compte du temps.

On peut citer cependant une timide exploration d'objets temporels dans YAFOOL, ainsi qu'une approche plus systématique, mais plus tellement objet — puisqu'elle porte sur des variables — dans KALEIDOSCOPE [Freeman-Benson, 1990b ; Freeman-Benson, 1990c ; Augeraud et Freeman-Benson, 1990].

Le temps intervient bien sûr aussi au niveau MÉTA, lorsque l'on considére l'EXTENSIBILITÉ d'un système.

terme

Terme utilisé dans les LT pour désigner aussi bien les classes que les instances. On dit aussi CONCEPT, mais il faudrait spécifier "concept générique" ou "concept individuel".

thèse

C'est le premier temps dialectique : dans toutes ses manifestations, l'approche objet repose sur un modèle sous-jacent commun et "naturel". Voir ANTITHÈSE, SYNTHÈSE, ONTOLOGIQUE, ÊTRE.

totalitaire

Tout ce qui n'est pas autorisé est interdit. Qui n'est pas avec moi est contre moi. Voir AUTORITAIRE.

treillis

C'est une notion mathématique précise, mais un terme employé très souvent abusivement. Un treillis — en anglais *lattice* — est la première généralisation naturelle de la notion d'ordre total. Dans un ordre total $\leq$, deux éléments x et y quelconques sont toujours comparables : $x \leq y$ ou $y \leq x$. Dans un treillis, deux éléments x et y quelconques possèdent toujours un unique majorant minimal (borne supérieure ou *supremum*) et un unique minorant maximal (borne inférieure ou *infimum*) : bien souligner l'existence et l'unicité. Dans un ordre total, les infimum et supremum sont x et y eux-mêmes. La notion de treillis se généralise ensuite à celle de demi-treillis inférieur (resp. supérieur) lorsque seul l'infimum (resp. supremum) existe. Une *arborescence* est un demi-treillis supérieur, qui peut être considéré comme un treillis si on lui rajoute un élément distingué minimal, $\bot$. Infimum et supremum peuvent être considérés comme des opérateurs ou lois de composition (associatives), notés alors $\vee$ et $\wedge$ par analogie avec le treillis booléen, ce qui fait du treillis une structure algébrique. Le treillis peut aussi être *distributif* lorsque les deux opérations sont distributives l'une par rapport à l'autre, ou *complet* lorsque l'on généralise la définition à une partie quelconque (au lieu de 2 éléments x et y). Tout treillis fini est complet et possède deux éléments distingués, respectivement minimal ($\bot$) et maximal ($\top$).

Les treillis interviennent de deux façons différentes dans la problématique objet. D'abord, les hiérarchies sont des ordres partiels, et il est naturel de se demander dans quelle mesure c'est — ou ce doit être — un treillis. Toute interprétation extensionnelle des hiérarchies conduit à une analogie avec le treillis ensembliste qui fait suspecter une nécessité de treillis. Mais ce n'est pas si simple.

- Les TYPES des langages de programmation forment en général un treillis [Scott, 1976], bien qu'il semble qu'un demi-treillis inférieur suffise : il faut toujours pouvoir faire l'intersection de deux types, mais l'union est rarement demandée.
- Dans les LT, la hiérarchie de concepts explicitement définis (ou primitifs) ne constitue pas un treillis, mais elle est basée sur un demi-treillis supérieur virtuel de tous les termes

nécessaires pour contenir la hiérarchie explicite. Tout se passe comme si tous ces termes avaient été explicitement définis. On peut le voir aussi comme un demi-treillis inférieur, avec un élément distingué minimal ($\perp$) qui spécialise tout concept. Il n'est alors pas possible de distinguer l'ouverture vers le bas de l'inconsistance. Toute mention implicite d'une combinaison de termes compatibles mais dont l'infimum serait $\perp$ doit alors se traduire par la définition du terme correspondant comme infimum de la combinaison.

- Pour tout ce qui est LPO ou RCO, il n'y a a priori aucune obligation à ce que les hiérarchies soient des treillis, malgré la fréquence de l'usage abusif de l'expression "treillis d'héritage multiple" qui est à proscrire.
- Dans les contraintes, une approximation des DOMAINES exacts des variables par un treillis recouvrant permet à [Caseau, 1993a] de réduire algébriquement la complexité de la résolution.

Il est certain que le fait que la hiérarchie soit un treillis présente certains avantages : les classes sont en général utilisées pour "typer" les RELATIONS et il peut être intéressant d'avoir un demi-treillis inférieur pour résoudre les conflits d'héritage multiple sur ces types. Mais un demi-treillis inférieur est une condition très forte qui limite terriblement l'EXTENSIBILITÉ d'un système à objets : il faudrait en effet prévoir toutes les combinaisons de classes existantes, ou avoir un élément distingué ($\perp$) pour désigner aussi bien les combinaisons encore inconnues que l'inconsistance[29]. Lorsque la condition de demi-treillis supérieur est acquise, on peut vouloir remplacer la condition globale de demi-treillis inférieur par la condition locale de demi-treillis inférieur sur toutes les restrictions aux super-classes d'une classe (classe incluse). De façon générale, la condition de treillis est très contraignante dès que ce treillis n'est plus statique, ce qui est le cas dans la problématique objet : elle impose des mécanismes automatiques pour maintenir ce treillis qui sont bien fournis par le classifieur des LT mais qui n'ont pas d'équivalent naturel pour les systèmes à objets, et dont le coût n'est pas négligeable. Divers systèmes offrent néanmoins des mécanismes de maintien automatique de (demi-)treillis [Aït-Kaci et Nasr, 1986 ; Aït-Kaci *et al.*, 1989 ; Agrawal *et al.*, 1991 ; Caseau, 1993b].

Les treillis interviennent aussi dans la solution des conflits d'HÉRITAGE MULTIPLE de valeur, lorsque la propriété a une sémantique spécifique qui permet une résolution correcte : dans ce cas, le domaine de la propriété semble devoir être un treillis. Le cas typique est celui des conflits sur les aspects des attributs qui décrivent leur domaine. Le conflit se résout par l'infimum (ou le supremum) des valeurs en conflit.

treillis de Galois

Etant donné un vocabulaire de base constitué par un ensemble de propriétés munies d'un domaine, il est possible de constituer l'univers du discours engendré par ce vocabulaire comme étant l'ensemble des termes (au sens des LT) définissables par combinaison de ces propriétés et des restrictions sur leur domaine. On obtient alors un treillis de Galois [Dicky *et al.*, 1994]. Les approches logiques, PROLOG par exemple, procèdent de manière similaire avec l'*univers de Herbrand*. [Coupey et Fouqueré, 1994] construit ainsi un domaine récursif qui devient à chaque étape de la récursion le domaine des propriétés, mais il en fait une algèbre (?).

Tropes

> La classification des tropes, éclairée par leur théorie [...] permet en effet d'énoncer des vérités fondamentales sur la nature de toute classification, car [...] elle en décrit la structure. *La classification des tropes est donc la classification des racines de la classification.*
>
> Patrick Tort, *De la double racine du principe de classification*, 1983, in [Tort, 1989].

Le modèle TROPES a été élaboré dans sa thèse par Olga Mariño, dans l'équipe de F. Rechenmann à Grenoble [Mariño, 1989 ; Mariño *et al.*, 1990 ; Mariño, 1991 ; Mariño, 1993][30]. Dans ce modèle, les connaissances sont organisées en trois niveaux :

[29] Il ne semble pas y avoir moyen d'utiliser deux éléments distingués différents pour représenter les sous-classes non encore définies et les sous-classes inconsistantes.

[30] Nous avons pris implicitement le parti de ne pas décrire en détail tous les systèmes de RCO existants : nous faisons ici une exception pour TROPES qui présente un modèle original en donnant un sens tout à fait particulier au vocabulaire du domaine, tout en offrant un bon exemple pour le débat en cours. Par ailleurs, cette description de TROPES décrit plus les travaux d'Olga Mariño, que ceux de Jérôme Euzenat.

– la base de connaissances est partitionnée en *concepts* indépendants — DISJOINTS donc incomparables (pas de hiérarchie de concepts) —, chaque concept décrivant toutes ses *instances* par le même ensemble d'attributs ; en particulier, TROPES ne traite absolument pas le problème des attributs HOMONYMES pour un même concept ;
– chaque concept est lui-même organisé en *points de vue* ;
– chaque POINT DE VUE d'un concept est une arborescence[31] de *classes* qui définissent et spécialisent certains des attributs du concept. Une instance d'un concept est instance directe d'une (et d'une seule) classe dans chaque point de vue du concept. TROPES est donc un modèle avec MULTI-INSTANCIATION.

Enfin, transversalement, les points de vue d'un même concept sont reliés par des *passerelles* qui décrivent des relations de généralisation / spécialisation entre classes de points de vue différents. En particulier, ces passerelles peuvent être implicites : deux classes de même nom dans des points de vue différents sont considérées comme identiques.

Il est important de voir que ces passerelles ont un rôle purement déductif et ne servent pas directement à la classification. Soit une passerelle $A \rightarrow B$: si x est un B, A ne sera pas examiné comme classe d'instanciation possible de x, à moins que x soit aussi une instance d'une super-classe directe de A. De même, si y est une instance d'une super-classe directe de A, les propriétés de y définies' dans B mais pas dans A ne seront pas examinées pour savoir si y n'est pas une instance possible de A. C'est ce rôle purement déductif qui permet aux passerelles d'améliorer le problème des valeurs manquantes.

Le modèle TROPES propose ainsi implicitement une relation de compatibilité entre classes, soit par la disjonction entre concepts, entre sous-classes d'une même classe, ou entre classes de points de vue différents dont les domaines d'au moins un attribut sont incompatibles, soit par l'inclusion inter-point de vue des passerelles ou intra-point de vue de la relation de généralisation entre classes. L'expression de la compatibilité n'est cependant pas complète puisque l'on pourrait vouloir décrire comme DISJOINTS des classes de points de vue différents dont les domaines sont compatibles.

Le modèle TROPES peut se traduire dans un modèle objet traditionnel avec mono-instanciation. A partir de l'ensemble des points de vue d'un concept, on obtient un graphe d'héritage multiple "équivalent" en procédant comme suit : on commence par calculer le produit d'ordre (cf. [Ducournau et Habib, 1989]) des points de vue, c'est-à-dire la relation d'ordre produit sur le produit cartésien des points de vue considérés comme des ensembles de classes. A chaque classe-produit, on associe naturellement une extension qui est l'intersection des extensions des classes qui forment la classe produit : $Ext((C_1, C_2)) = Ext(C_1) \cap Ext(C_2)$. On lui associe tout aussi naturellement une intension décrite par l'intersection des domaines sur la réunion des propriétés. Par le jeu des relations de spécialisation intra-points de vue ou inter-points de vue (passerelles), beaucoup de classe-produit ont mêmes extension et intension : si $Ext(C_1) \subset Ext(C_2)$, alors $Ext((C_1, C_2)) = Ext(C_1)$. En particulier, beaucoup de classe-produit sont inconsistantes car elles contiennent au moins deux classes incompatibles pour un attribut au moins. On quotiente donc le produit cartésien par cette relation d'équivalence pour obtenir un ensemble de classe-produit. Dans cette opération, on perd de l'information : le fait que certains ensembles de sous-classes forment des partitions de l'une de leurs super-classes. Mais le résultat est strictement équivalent aux données de départ dans la mesure où les classes obtenues sont les seules "classes d'instanciation multiple" de la description du concept dans le modèle TROPES.

L'opération inverse n'est bien sûr pas possible, du moins sans perte de généralité. [Euzenat, 1994a] formalise d'une façon différente cette interprétation de TROPES comme un produit de systèmes CLASSIFICATOIRES.

Il est indéniable que TROPES offre une (première) solution tout à fait élégante à la représentation des points de vue en en faisant des objets distingués de la représentation et de la base

[31] Olga Mariño avait l'intention de donner la possibilité de l'héritage multiple.

de connaissances[32], qui permettent de se focaliser sur une partie du graphe d'héritage, et en offrant en plus implicitement des relations d'incompatibilité (partielle) et des partitions. Enfin, la possibilité de classer dans un point de vue — lorsque les informations nécessaires y sont disponibles — permet de déduire, par le jeu des passerelles, des informations manquantes dans un autre point de vue. Sans aucune généralité bien sûr.

typage

Tous les langages évolués sont typés : la question est le type de leur typage : statique, dynamique, fort, faible, etc.

type

"Type" et "classe" sont très proches. On peut par exemple considérer qu'un type c'est la réunion d'une classe et de toutes ses sous-classes : mais cela se traduit, que ce soit en termes d'intension ou d'extension, par une équivalence entre type et classe.

Le terme est souvent utilisé en référence aux *types abstraits*, donc dans un souci d'ENCAPSULATION : dans ce cas, le type serait l'extension de la classe, et son intension restreinte aux méthodes. Cependant, la littérature objet abonde en articles qui comparent types et CLASSES, héritage et sous-typage, etc. [America, 1991]. Là encore, on peut suspecter la notion de type, telle qu'elle vient des langages de programmation [Cardelli et Wegner, 1985 ; Danforth et Tomlinson, 1988 ; Gunter et Mitchell, 1994], d'être inadaptée aux objets. Voir POLYMORPHISME. On parle aussi de type des attributs, mais cela ne doit pas être confondu avec leur DOMAINE, au moins dans le cas MULTI-VALUÉ. De son côté, le type des méthodes semble poser des problèmes plus ou moins insurmontables de *contravariance*.

[Capponi, 1993] définit le type d'une classe comme le produit cartésien — le type "record" — des types de ses différents attributs.

L'origine lointaine de la notion de type en informatique doit sans doute — mais est-ce si sûr que ça ? — être la "théorie des types" de RUSSEL. [Scott, 1976] propose des types une vision aussi élégante d'un point de vue mathématique qu'ésotérique du point de vue informatique.

typicalité

La notion de typicalité intervient dès que les classes sont décrites avec des propriétés (valeurs) par défaut qui n'expriment plus des conditions absolument NÉCESSAIRES et qui autorisent donc des EXCEPTIONS (masquage) et un HÉRITAGE NON MONOTONE [Rossazza, 1990 ; Garlatti, 1993 ; Padgham et Zhang, 1993 ; Padgham et Nebel, 1993 ; Coupey et Fouqueré, 1994]. La typicalité est très liée aux théories sur les PROTOTYPES. Pour l'instant, c'est une notion plus philosophique qu'informatique dans la mesure où elle est loin d'avoir été formalisée.

univocité

D'après [Euzenat, 1993a], propriété d'un système CLASSIFICATOIRE dans lequel chaque objet est INSTANCE DIRECTE d'une seule classe. C'est un système de MONO-INSTANCIATION.

univoque (propriété)

Un nom de propriété est univoque s'il désigne sans ambiguïté une propriété dans le contexte considéré. Pour les LT, et pour beaucoup de systèmes à objets comme CLOS ou YAFOOL, tous les noms de propriétés sont globalement univoques, car il n'est pas possible d'avoir des propriétés HOMONYMES. Voir HÉRITAGE, IDENTITÉ.

utilisateur

Une sale engeance avec laquelle l'informaticien vit malgré tout en symbiose : il (l'informaticien) lui (l'utilisateur) pique ses problèmes, les lui résout et le supporte dans tous les sens du terme. Mais peut-il lui faire confiance pour ne pas casser son beau jouet et savoir l'apprécier ?

valeur

Tout "objet de première classe" du langage et tout élément du DOMAINE considéré. Tout ce qui peut valuer les attributs des objets et les variables des contraintes.

[32] Par opposition à (F)ROME ou OBJLOG où chaque classe peut devenir un point de vue de n'importe quelle sous-classe.

Les LT ont la particularité de faire de la classification non seulement sur les classes et les instances, mais aussi sur les valeurs, dans la mesure où un objet est classé automatiquement d'après les rôles qu'il value.

Yafool

Intégration des langages à objets et des langages de *frames*, dans un cadre de prototypes avec une sur-couche classe-instance. Système très libertaire, avec sur-couches de règles et environnement de programmation évolué [Ducournau, 1991].

zèbre

Un dictionnaire digne de ce nom se doit d'aller de A à Z, si ce n'est d'α en ω. Le problème du zèbre est un exemple classique de problème de satisfaction de CONTRAINTES, dont la modélisation objet est assez simple puisqu'elle fait intervenir des objets bien typés, dont un zèbre[33] [Kökény, 1994].

Plus curieusement, le problème du zèbre se décrit aussi très bien dans le formalisme des LT — à condition d'avoir des rôles inverses — et sa solution en est une conséquence valide, en fait l'unique modèle minimal (à condition sans doute de considérer la description du monde comme fermée ?) cela ne signifie sans doute pas que le problème des LT est plus général que celui des CSP, mais c'est une bonne indication de la nature combinatoire du premier.

[33] Ce qui tendrait à prouver que l'auteur de ce problème n'avait pas lu Alphonse Allais : cf. note 3.

Bibliographie

[Abadi et Cardelli, 1994] M. Abadi et L. Cardelli. A semantics of object types. In *LICS'94*, 1994.

[Agrawal *et al.*, 1991] R. Agrawal, A. Borgida, et H.V. Jagadish. Efficient management of transitive relationships in large data-bases including is-a hierarchies. Rapport Technique DCS-TR-286, Rutgers University, 1991.

[America, 1991] P. America. A behavioural approach to subtyping in object-oriented programming languages. In Lenzerini et al. [1991], chapitre 11.

[Augeraud et Freeman-Benson, 1990] M. Augeraud et B. Freeman-Benson. Dynamic objects. Rapport Technique LIST-90-??, Université de Nantes, Laboratoire d'Informatique des Sciences et des Techniques, 1990.

[Aït-Kaci *et al.*, 1989] H. Aït-Kaci, R. Boyer, P. Lincoln, et R. Nasr. Efficient implementation of lattice operations. *ACM Trans. Program. Lang. Syst.*, 11(1) :115–146, 1989.

[Aït-Kaci et Nasr, 1986] H. Aït-Kaci et R. Nasr. LOGIN : a logic programming language with built-in inheritance. *Journal of Logic Programming*, 3 :185–215, 1986.

[Aït-Kaci, 1991] H. Aït-Kaci. A glimpse of Paradise. In *Next Generation Information System Technology*, éditeurs J.W. Schmidt et A.A. Stogny, pages 15–25. Springer-Verlag, 1991.

[Baker, 1991] H.G. Baker. CLOStrophobia : its etiology and treatment. *OOPS Messenger*, 2(4), 1991.

[Berlandier et Neveu, 1992] P. Berlandier et B. Neveu. PROSE : un système de contraintes facilement intégrable dans un langage à objets. In Habib et Oussalah [1992], pages 161–170.

[Berthier, 1994] D. Berthier. L'agent rationnel, objet de l'IA ? *Revue d'Intelligence Artificielle*, 8(4) :327–349, 1994.

[Bezivin *et al.*, 1987] éditeurs J. Bezivin, P. Cointe, J.-M. Hullot, et H. Liebermann. *Proceedings of the 1st European Conference on Object-Oriented Programming, ECOOP'87*, LNCS 276. Springer, 1987.

[Binaghi *et al.*, 1989] E. Binaghi, D. Orban, et A. Rampini. Fuzzy Logic Based Tools for Classification and Reasoning with Uncertainty. In IEEE *International Workshop on Tools for Artificial Intelligence (IWTAI'89), Fairfax, Virginia*, pages 572–577, 1989.

[Birtwistle *et al.*, 1973] G. Birtwistle, O. Dahl, B. Myraug, et K. Nygaard. SIMULA *begin*. Petrocelli Charter, 1973.

[Blake et Cook, 1987] E. Blake et S. Cook. On including part hierarchies in object-oriented languages with an implementation in SMALLTALK. In Bezivin et al. [1987].

[Bobrow et Stefik, 1983] D. G. Bobrow et M. Stefik. *The* LOOPS *manual*. Xerox, Palo Alto, 1983.

[Borgida et Patel-Schneider, 1994] A. Borgida et P.F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *JAIR*, 1 :277–308, 1994.

[Borgida, 1981] A. Borgida. On the definition of specialization hierarchies for procedures. In *Proceedings of the 7th IJCAI, Vancouver, Canada*, pages 254–256, 1981.

[Borgida, 1992] A. Borgida. From type systems to knowledge representation : natural semantics specifications for description logics. *Int. J. on Intelligent and Cooperative Information Systems*, 1(1) :93–126, 1992.

[Borning *et al.*, 1987] A. Borning, R. Duissberg, B. Freeman-Benson, A. Kramer, et M. Woolf. Constraints hierarchies. In OOPSLA [1987].

[Borning, 1981] A. Borning. The programming language aspects of THINGLAB, a constraint-oriented simulation laboratory. *ACM Transactions on Programming Languages and Systems*, 3(4) :353–387, Oct 1981.

[Brachman, 1985] R. Brachman. "I lied about the trees", or defaults and definitions in knowledge representation. *AI Magazine*, 6(3) :80–93, 1985.

[Bunge, 1977] M. Bunge. *Ontology I : The Furniture of the World*, volume 3 de *Treatise on Basic Philosophy*. Reidel, Boston, 1977.

[Bunge, 1979] M. Bunge. *Ontology II : A World of Systems*, volume 4 de *Treatise on Basic Philosophy*. Reidel, Boston, 1979.

[Capponi, 1993] C. Capponi. Classification des classes par les types. In Habib et Oussalah [1993], pages 215–224.

[Cardelli et Wegner, 1985] L. Cardelli et P. Wegner. On understanding types, data abstraction and polymorphism. *ACM Computing Surveys*, 17(4), 1985.

[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.

[Carré *et al.*, 1990] B. Carré, L. Dekker, et J-M. Geib. Multiple and evolutive representation in the ROME language. In *Second International Conference on Technology of Object-Oriented Languages and Systems*, Paris, 1990.

[Carré *et al.*, 1995] B. Carré, R. Ducournau, J. Euzenat, A. Napoli, et F. Rechenmann. Classification et objets : programmation ou représentation. In *Actes des 5ième journées nationales du PRC-GDR Intelligence Artificielle*, éditeur D. Kayser, pages 213–237. Teknea, 1995.

[Carré et Dekker, 1991] B. Carré et L. Dekker. Inheriting object-oriented features through metaprogramming. In Mrazik [1991].

[Carré et Geib, 1990] B. Carré et J-M. Geib. The point of view notion for multiple inheritance. In *Proc. OOPSLA/ECOOP'90*. ACM Press, 1990.

[Casais, 1991] E. Casais. *Managing Evolution in Object Oriented Environments : An Algorithmic Approach*. PhD thesis, Université de Genève, 1991.

[Caseau, 1987] Y. Caseau. *Etude et réalisation d'un langage objet : LORE*. Thèse d'Informatique, Université de Paris-Sud, 1987.

[Caseau, 1989] Y. Caseau. A formal system for producing demons from rules. In *Proc. DOOD'89*, Tokyo, 1989.

[Caseau, 1991] Y. Caseau. An object-oriented deductive language. *Annals of Mathematics and Artificial Intelligence*, 3 :211–258, 1991.

[Caseau, 1993a] Y. Caseau. Compiling abstract interpretation over ordered domains, 1993.

[Caseau, 1993b] Y. Caseau. Efficient handling of multiple inheritance hierarchies. In *Proc. OOPSLA'93*, pages 271–287. ACM Press, 1993.

[Chambers et Ungar, 1992] C. Chambers et D. Ungar. *Lisp and Symbolic Computation*, 4(3), 1992. Numéro spécial consacré au langage SELF.

[Cointe, 1987] P. Cointe. Metaclasses are first class : the ObjVlisp model. In OOPSLA [1987], pages 156–167.

[Coupey et Fouqueré, 1994] P. Coupey et Ch. Fouqueré. Classer des concepts définis avec des défauts et des exceptions. In Rechenmann [1994], pages 69–80.

[Cunis *et al.*, 1994] éditeurs R. Cunis, D. de Champeaux, et H. Kaindl. *ECAI'94 workshop on Integrating Object-orientation and Knowledge Representation*, 1994.

[Danforth et Tomlinson, 1988] S. Danforth et C. Tomlinson. Type theories and object-oriented programming. *ACM Computing Surveys*, 20(1), 1988.

[David et Krivine, 1987] J.-M. David et J.-P. Krivine. Utilisation de prototypes dans un système expert de diagnostic : le projet DIVA. In *Actes 7^{e}Journées Internationales "Les systèmes experts et leurs applications"*, Avignon, mai 1987.

[David et Krivine, 1988] J.-M. David et J.-P. Krivine. Acquisition de connaissances expertes à partir de situations types. In *Actes 8^{e}Journées Internationales "Les systèmes experts et leurs applications"*, Avignon, mai 1988.

[Dekker et Carré, 1992] L. Dekker et B. Carré. Multiple and dynamic representation of frames with points of view in FROME. In *Actes de la Conférence Représentations Par Objets (RPO'92), La Grande Motte, France*, pages 97–111. EC2, Nanterre, 1992.

[Dekker, 1993] L. Dekker. La réification des filtres en langage FROME. In Habib et Oussalah [1993], pages 23–35.

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

[Dicky *et al.*, 1994] H. Dicky, C. Dony, M. Huchard, et T. Libourel. Un algorithme d'insertion avec restructuration dans les hiérarchies de classes. In Rechenmann [1994], pages 125–136.

[Dionne *et al.*, 1993] R. Dionne, E. Mays, et F.J. Oles. The equivalence of model-theoretic and structural subsumption in description logics. pages 710–716, 1993.

[Ducournau *et al.*,] R. Ducournau, M. Habib, et G. Simonet. Méta-héritage et héritage non-transitif. *Revue d'Intelligence Artificielle*. (à paraître).

[Ducournau *et al.*, 1992] R. Ducournau, M. Habib, M. Huchard, et M.-L. Mugnier. Monotonic conflict resolution mechanisms for inheritance. In *Proc. OOPSLA'92*, pages 16–24. ACM Press, 1992.

[Ducournau *et al.*, 1994] R. Ducournau, M. Habib, M. Huchard, et M.-L. Mugnier. Proposal for a monotonic multiple inheritance linearization. In *Proceedings of OOPSLA'94, Portland (OR), USA*, special issue of ACM SIGPLAN Notices, 29(10), pages 164–175, 1994.

[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 Habib, 1989] R. Ducournau et M. Habib. La multiplicité de l'héritage multiple. *Technique et Science Informatiques*, 8(1) :41–62, 1989.

[Ducournau et Habib, 1991] R. Ducournau et M. Habib. Masking and conflicts, or to inherit is not to own. In Lenzerini et al. [1991], chapitre 14, pages 223–244.

[Ducournau, 1991] R. Ducournau. Y3 : YAFOOL*, le langage à objets*. Sema Group, 1991.

[Ducournau, 1993] R. Ducournau. *Héritages et Représentations*. Mémoire d'habilitation à diriger des recherches, Université Montpellier II, 1993.

[Ducournau, 1996] R. Ducournau. Des langages à objets aux logiques terminologiques : les systèmes classificatoires. Rapport Technique 96-030, LIRMM, Université Montpellier 2, 1996.

[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.

[Dugerdil, 1991] P. Dugerdil. Inheritance mechanism in the Objlog language : multiple selective and multiple vertical with points of view. In Lenzerini et al. [1991], chapitre 15, pages 245–256.

[Eco, 1994] U. Eco. *La recherche de la langue parfaite*. Seuil, Paris, 1994.

[Einstein *et al.*, 1935] A. Einstein, B. Podolski, et N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical Review*, XLV :777–780, 1935. Trad. française in [Einstein, 1989].

[Einstein, 1989] A. Einstein. *Quanta*, volume I de *Œuvres choisies*. Seuil, Paris, 1989.

[Euzenat, 1993a] J. Euzenat. Définition abstraite de la classification et son application aux taxonomies d'objets. In Habib et Oussalah [1993], pages 235–246.

[Euzenat, 1993b] J. Euzenat. On a purely taxonomic and descriptive meaning for classes. In Napoli [1993].

[Euzenat, 1994a] J. Euzenat. Classification dans les représentations par objets : produits de systèmes classificatoires. In *Actes de la conférence Reconnaissance des Formes et Intelligence Artificielle*, 1994.

[Euzenat, 1994b] J. Euzenat. KR and OOL co-operation based on semantics non reducibility. In Cunis et al. [1994].

[Faucher, 1992] C. Faucher. Extensibilité d'un langage de schémas relativement à ses aspects. In Habib et Oussalah [1992], pages 30–42.

[Ferber, 1983] J. Ferber. MERING *un langage d'acteurs pour la représentation des connaissances.* Thèse de docteur-ingénieur, Université Paris VI, 1983.

[Finin et McGuire, 1991] T. Finin et J. McGuire. Inheritance in logic programming knowledge bases. In Lenzerini et al. [1991], chapitre 18.

[Freeman-Benson, 1990a] B. Freeman-Benson. The evolution of constraint imperative programming. Rapport Technique LIST-90-05, Université de Nantes, Laboratoire d'Informatique des Sciences et des Techniques, sept 1990.

[Freeman-Benson, 1990b] B. Freeman-Benson. Kaleidoscope : mixing objects, constraints and imperative programming. In OOPSLA/ECOOP [1990].

[Freeman-Benson, 1990c] B. Freeman-Benson. The Kaleidoscope programming language : a second report. Rapport Technique LIST-90-06, Université de Nantes, Laboratoire d'Informatique des Sciences et des Techniques, sept 1990.

[Frege, 1971] G. Frege. *Ecrits logiques et philosophiques.* L'ordre philosophique. Seuil, Paris, 1971.

[Garlatti, 1993] S. Garlatti. Les objets comme représentation des catégories naturelles et artefactuelles. In Habib et Oussalah [1993], pages 15–22.

[Gensel et Girard, 1992] J. Gensel et P. Girard. Expression d'un modèle de tâches à l'aide d'une représentation par objets. In Habib et Oussalah [1992], pages 225–236.

[Girod, 1991] X. Girod. *MECANO, une Méthode et un Environnement de Construction d'ApplicatioNs par Objets.* Thèse d'Informatique, Université Joseph Fourier, Grenoble, 1991.

[Gochet et Gribomont, 1991] P. Gochet et P. Gribomont. *Logique, méthodes pour l'informatique fondamentale*, volume 1. Hermès, Paris, 1991.

[Gochet et Gribomont, 1994] P. Gochet et P. Gribomont. *Logique, méthodes pour l'informatique fondamentale*, volume 2. Hermès, Paris, 1994.

[Goldberg et Robson, 1983] A. Goldberg et D. Robson. SMALLTALK *: the language and its implementation.* Addison-Wesley, 1983.

[Granger et Thonnat, 1985] C. Granger et M. Thonnat. Un système de vision fondé sur une méthode structurelle de classification. In *Cognitiva 85*, 1985.

[Granger, 1986] C. Granger. Fuzzy reasoning in a knowledge-based system for object recognition. In *Proc. of ECAI'86*, pages 163–170, 1986.

[Granger, 1988] C. Granger. An Application of Possibility Theory to Object Recognition. *Fuzzy Sets and Systems*, 28(4) :351–362, 1988.

[Grivaud et Rechenmann, 1992] S. Grivaud et F. Rechenmann. Navigation dans les bases de connaissances associant objets et hypertexte. In Habib et Oussalah [1992], pages 269–280.

[Gunter et Mitchell, 1994] éditeurs C. A. Gunter et J. C. Mitchell. *Theoretical Aspects of Object-Oriented Programming : Types, Semantics, and Language Design.* The MIT Press, 1994.

[Habib et Oussalah, 1992] éditeurs M. Habib et M. Oussalah. *Actes des journées Représentation Par Objets, RPO'92*, La Grande Motte, juin 1992. EC2.

[Habib et Oussalah, 1993] éditeurs M. Habib et M. Oussalah. *Actes des journées Représentation Par Objets, RPO'93*, La Grande Motte, juin 1993. EC2.

[Horty, 1994] J.F. Horty. Some direct theories of nonmonotonic inheritance. In *Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 2 : Nonmonotonic Reasoning*, éditeurs D. Gabbay et C. Hogger. Oxford University Press, 1994.

[ILO, 1991] ILOG, Gentilly. SMECI, *reference manual*, 1991.

[Kahn, 1987] G. Kahn. Natural semantics. Rapport Technique RR 601, I.N.R.I.A., 1987.

[Kaindl, 1994] H. Kaindl. Comparing object-oriented analysis with knowledge acquisition. *OOPS Messenger*, 5(3) :1–5, 1994.

[Kay, 1993] A.C. Kay. The early history of SMALLTALK. In *Proc. 2nd History of Programming Languages Conference, HOPL-II*, éditeurs J.A.N. Lee et J.E. Sammet. ACM Press, 1993.

[Keene, 1989] S.E. Keene. *Object-Oriented Programming in* COMMON LISP. *A programmer's guide to* CLOS. Addison-Wesley, Reading, MA, 1989.

[Kiczales *et al.*, 1991] G. Kiczales, J. des Rivières, et D.G. Bobrow. *The Art of the Meta-Object Protocol.* MIT Press, 1991.

[Kokeny, 1993] T. Kokeny. CSPOO : un système de résolution de contraintes orientées objet. In Habib et Oussalah [1993], pages 39–50.

[Kökény, 1994] T. Kökény. *Satisfaction de contraintes dans un environnement orienté objets.* Thèse d'Informatique, Université des Sciences et Techniques du Languedoc, Montpellier, 1994.

[Kristensen *et al.*, 1987] B.B. Kristensen, O.L. Madsen, B. Møller-Pedersen, et K. Nygaard. Classification of actions or inheritance also for methods. In Bezivin et al. [1987], pages 109–118 ?

[Lalande, 1926] A. Lalande. *Vocabulaire technique et critique de la philosophie.* Presses Universitaires de France, 1926.

[Lenat et Guha, 1990a] D.B. Lenat et R.V. Guha. *Building large knowledge-based systems : representation and inference in the* CYC *project.* Addison-Wesley, Reading, MA, 1990.

[Lenat et Guha, 1990b] D.B. Lenat et R.V. Guha. CYC : a midterm report. *AI Magazine*, 11(3) :33–59, 1990.

[Lenzerini *et al.*, 1991] éditeurs M. Lenzerini, D. Nardi, et M. Simi. *Inheritance Hierarchies in Knowledge Representation and Programming Langages.* John Wiley & Sons, 1991.

[Lesniewski, 1989] S. Lesniewski. *Sur les fondements de la mathématique.* Hermès, Paris, 1989.

[MacGregor, 1991] R. MacGregor. The evolving technology of classification-based knowledge representation systems. In Sowa [1991], pages 385–400.

[Magnan, 1994] M. Magnan. *Réutilisation de composants : les exceptions dans les objets composites.* Thèse d'Informatique, Université des Sciences et Techniques du Languedoc, Montpellier, 1994.

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

[Mariño *et al.*, 1990] O. Mariño, F. Rechenmann, et P. Uvietta. Multiple perspectives and classification mechanism in object oriented classification. In *Proc. ECAI'90*, pages 425–430, 1990.

[Mariño, 1989] O. Mariño. *Classification d'objets dans un modèle multi-points de vue.* Mémoire de DEA, Université Grenoble I, 1989.

[Mariño, 1991] O. Mariño. Classification d'objets composites dans un système de représentation des connaissances multi-points de vue. In *Actes RFIA'91*, 1991.

[Masini *et al.*, 1989] G. Masini, A. Napoli, D. Colnet, D. Leonard, et K. Tombre. *Les langages à objets.* InterEditions, 1989.

[Meyer, 1990] B. Meyer. *Introduction to the Theory of Programming Languages.* Prentice Hall, 1990.

[Meyer, 1991] B. Meyer. *Conception et programmation par objets.* 1991.

[Milner, 1978] R. Milner. A theory of type polymorphism in programming. *J. Comput. Syst. Sci*, 17 :348–375, 1978.

[Minsky, 1975] M. Minsky. A framework for representing knowledge. In *The psychology of computer vision*, éditeur P.H. Winston, pages 211–281. McGraw-Hill, 1975.

[Mosses, 1990] P.D. Mosses. Denotational semantics. In *Formal Models and Semantics*, éditeur J. Van Leeuwen, volume 1 de *Handbook of Theoretical Computer Science.* Elsevier, Amsterdam, 1990.

[Mrazik, 1991] éditeur Augustin Mrazik. *Proceedings of the 1st East European Conference on Object-Oriented Programming, EurOOP'91*, Bratislava, sep. 1991.

[Mulet et Marco, 1994] Ph. Mulet et J. Marco. De la parente' entre les environnements de MIT SCHEME et les prototypes de SELF. In Rechenmann [1994], pages 167–178.

[Myers *et al.*, 1990] B.A. Myers, D.A. Giuse, R.B. Dannenberg, B. Van der Zanden, D. Kosbie, E. Previn, A. Mickish, et P. Marchal. Garnet : Comprehensive support for graphical highly-interactive user interfaces. *IEEE Computer*, 23(11) :71–85, 1990.

[Napoli *et al.*, 1994] A. Napoli, C. Laurenço, et R. Ducournau. An object-based approach to organic synthesis planning. *Int. J. Human-Computer Studies*, 41 :5–32, 1994.

[Napoli, 1991] A. Napoli. Le raisonnement par classification. In *Le Raisonnement en Intelligence Artificielle*, chapitre 8. InterEditions, 1991.

[Napoli, 1992] A. Napoli. *Représentations à objets et raisonnement par classification en intelligence artificielle.* Thèse de Doctorat d'État en Informatique, Université Henri Poincaré Nancy 1, 1992.

[Napoli, 1993] éditeur A. Napoli. Nancy, France, 1993. Rapport de Recherche CRIN 93-R-156.

[Nebel, 1990] B. Nebel. *Reasoning and revision in hybrid representation systems.* Numéro 422 dans LNCS. Springer-Verlag, Berlin, 1990.

[OOPSLA, 1987] *Proceedings of the 2nd ACM Conference on Object-Oriented Programming, Languages and Applications, OOPSLA'87*, SIGPLAN Notices, 22(12). ACM Press, 1987.

[OOPSLA/ECOOP, 1990] *Proceedings of the 5th ACM Conference on Object-Oriented Programming, Languages and Applications, 3rd European Conference on Object-Oriented Programming, OOPSLA/ECOOP'90*, SIGPLAN Notices, 25(10). ACM Press, 1990.

[Padgham et Nebel, 1993] L. Padgham et B. Nebel. Combining classification and nonmonotonic inheritance reasoning : a first step. In *Proc. 7^{th} International Symposium on Methodologies for Intelligent Systems*, pages 15–18, Trondheim, Norway, 1993.

[Padgham et Zhang, 1993] L. Padgham et T. Zhang. A terminological logic with defaults : a definition and an application. pages 663–668, 1993.

[Patel-Schneider, 1989] P.F. Patel-Schneider. Undecidability of subsumption in NIKL. *Artificial Intelligence*, 39(2) :263–272, 1989.

[Perrot et Wolinski, 1992] J.-F. Perrot et F. Wolinski. Modélisation par objets en robotique. *Technique et Science Informatiques*, 11(1) :97–115, 1992.

[Pitrat, 1990] J. Pitrat. *Me'taconnaissance, futur de l'intelligence artificielle.* Herme's, Paris, 1990.

[Quine, 1977] W.V. Quine. *Relativité de l'ontologie et autres essais.* Aubier Montaigne, Paris, 1977.

[Quine, 1993] W.V. Quine. *Quiddités. Dictionnaire philosophique par intermittence.* L'ordre philosophique. Seuil, Paris, 1993.

[Rathke et Redmiles, 1993] Ch. Rathke et D.F. Redmiles. Multiple perspectives for supporting explanation in context. Rapport Technique CU-CS-645-93, University of Colorado, Dept. of Computer Science and Institute of Cognitive Science, march 1993.

[Rathke, 1991] Ch. Rathke. Implementing frames in an object-oriented programming language. In Mrazik [1991].

[Rechenmann, 1988] F. Rechenmann. SHIRKA : système de gestion de bases de connaissances centrées-objet. Manuel de référence. Rapport technique, IMAG – INRIA Rhône-Alpes, Grenoble, 1988.

[Rechenmann, 1994] éditeur F. Rechenmann. *Actes des journées Langages et Modèles à Objets, LMO'94*, Grenoble, octobre 1994. IMAG.

[Rieu *et al.*, 1992] D. Rieu, G.T. Nguyen, et J. Escamilla. Méthodes et représentation des connaissances. In Habib et Oussalah [1992], pages 17–30.

[Ringard, 1993] Y.-J. Ringard. Watches and mustard : an integrated approach to time and food. [1993].

[Rossazza, 1990] J.-P. Rossazza. *Utilisation de hiérarchies de classes floues pour la représentation de connaissances imprécises et sujettes à exceptions : le système SORCIER.* Thèse d'Informatique, Université Paul Sabatier, Toulouse, 1990.

[Russel, 1921] B. Russel. *Introduction à la philosophie mathématique.* Payot, Paris, 1921. (trad. française, 1991).

[Schaff, 1960] A. Schaff. *Introduction à la sémantique.* Editions Anthropos, Paris, 1960.

[Schild, 1988] K. Schild. Undecidability of subsumption in 'U'. KIT-Report 67, Technische Universität, Berlin, 1988.

[Schmidt-Schauß, 1989] M. Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In *Proceedings of KR'89, Toronto, Canada*, éditeurs R.J. Brachman, H.J. Levesque, et R. Reiter, pages 421–431. Morgan Kaufmann Publishe, San Mateo (CA), USA, 1989.

[Scott, 1976] D. Scott. Data types as lattices. *SIAM J. Computer*, 5 :452–487, 1976.

[Simonet, 1994] G. Simonet. *Héritage non monotone à base de chemins et de graphes partiels.* Thèse d'Informatique, Université des Sciences et Techniques du Languedoc, Montpellier, 1994.

[Snyder, 1991] J. Snyder. Inheritance in object-oriented programming. In Lenzerini et al. [1991], chapitre 10, pages 153–171.

[Sowa, 1991] éditeur J.F. Sowa. *Principles of Semantic Networks : Explorations in the Representation of Knowledge.* Morgan Kaufmann Publishers, San Mateo (CA), USA, 1991.

[Steele, 1990] G.L. Steele. *Common Lisp, the Language.* Digital Press, second édition, 1990.

[Stein, 1992] L.A. Stein. Resolving ambiguity in nonmonotonic inheritance hierarchies. *Artificial Intelligence*, 55(2-3) :259–310, June 1992.

[Stroustrup, 1987] B. Stroustrup. What is "object-oriented programming" ? In Bezivin et al. [1987].

[Tarski, 1952] A. Tarski. The semantic conception of truth and the foundation of semantics. In *Semantics and the philosophy of language*, éditeur L. Linsky. Urbana, 1952.

[Thomason, 1992] R.H. Thomason. NETL and subsequent path-based inheritance theories. *Computers and Mathematics with Applications*, 23(2-5) :179–204, 1992.

[Tort, 1989] P. Tort. *La raison classificatoire.* Aubier, 1989.

[Volle, 1989] Ph. Volle. Coréférence et mécanismes déductifs dans un langage de représentation par objets. In *PRC-IA Représentation par objets.* Caen, 1989.

[Wand, 1989] Y. Wand. A proposal for a formal model of objects. In *Object-Oriented Concepts, Databases and Applications*, éditeurs W. Kim et F.H. Lochovsky, pages 537–559. ACM Press, 1989.

[Wegner, 1990] P. Wegner. Concepts and paradigms of object-oriented programming. *OOPS Messenger*, 1(1), 1990.

[Winograd, 1975] T. Winograd. Frame representation and the declarative / procedural controversy. In *Representation and Understanding : studies in cognitive science*, éditeurs D.G. Bobrow et A.M. Collins, pages 185–210. Academic Press, 1975.

[Winston *et al.*, 1987] M.E. Winston, R. Chaffin, et D. Herrmann. A taxonomy of part-whole relations. *Cognitive Science*, 11 :417–444, 1987.

[Winston et Horn, 1984] P.H. Winston et B.K.P. Horn. LISP. Addison-Wesley, 1984.

[Winston, 1984] P.H. Winston. *Artificial Intelligence*. Addison-Wesley, 1984.

[Woods et Schmolze, 1992] W. Woods et J. Schmolze. The KL-ONE family. *Computers and Mathematics with Applications*, 23(2-5) :133–177, 1992.

[Woods, 1991] W. Woods. Understanding subsumption and taxonomy : a framework for progress. In Sowa [1991], pages 45–94.