Classification et objets : programmation ou représentation ?

Bernard Carré
Cité scientifique, LIFL, 59655 Villeneuve d'Ascq
(carre@lifl.fr)

Roland Ducournau*
LIRMM, 161, rue Ada, 34392 Montpellier Cedex 5
(ducournau@lirmm.fr)

Jérôme Euzenat
INRIA Rhône-Alpes – IMAG-LIFIA, 46, avenue Felix Viallet, 38031 Grenoble
(Jerome.Euzenat@imag.fr)

Amedeo Napoli
CRIN CNRS – INRIA Lorraine, B.P. 239, 54506 Vandœuvre-lès-Nancy
(napoli@loria.fr)

François Rechenmann
INRIA Rhône-Alpes – IMAG-LIFIA, 46, avenue Felix Viallet, 38031 Grenoble
(Francois.Rechenmann@imag.fr)

*. Ce travail a été réalisé quand l'auteur était à Sema Group, 16-18 rue Barbès – 92126 Montrouge.

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

1 Introduction

La notion d'objet a pris une importance considérable dans l'informatique d'aujourd'hui [MNC+89] : en génie logiciel avec les langages de programmation à objets (LPO), en intelligence artificielle (IA) avec les systèmes de représentation des connaissances à objets (RCO), auxquels peuvent être associées, dans une certaine mesure, les logiques terminologiques (LT) [Neb90] [WS92], mais aussi dans les bases de données orientées objet et dans les méthodologies de conception objet.

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 taxonomique de ces classes en une structure de classification dont un mécanisme spécifique — la *classification* justement — permet d'assister la construction ou l'évolution. 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* [Wan89], même si cette proximité peut conduire à des désillusions.

Par delà les points communs à toutes ces manifestations des objets, il est possible de mettre 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, un dernier thème de recherche correspond à l'étude des possibilités de cohabitation, voire d'intégration.

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

- dans toutes ses manifestations, l'approche objet repose sur un modèle sous-jacent commun et "naturel" (*thèse*) ;
- mais ses deux branches LPO et RCO sont irréductibles l'une à l'autre (*antithèse*) ;
- comment peut-on faire cohabiter LPO et RCO ? L'intégration de la seconde dans le premier a-t-elle un sens ? (*synthèse*).

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 la diversité des points de vue de ses auteurs. Ces derniers peuvent ainsi se positionner sur plusieurs axes :

- les *orthodoxes* s'opposent aux *œcuménistes* dans leur vision de la cohabitation entre RCO et LPO ;
- les *optimistes* s'opposent aux *pessimistes* quant à la possibilité d'atteindre le paradis (de la représentation des connaissances ou des langages de programmation [AK91]).

L'article essaiera de pointer les différents sujets d'opposition qui sont, comme souvent, de l'ordre du verre à moitié vide ou du verre à moitié plein, autrement dit, qui peuvent être appréhendés en fait comme des façons différentes de réagir au même constat de difficulté. Plus généralement, les thèmes débattus dans le groupe se retrouvent au cœur des discussions des conférences *Représentation Par Objets* (RPO 92 et 93) de La Grande Motte [HO92, HO93] et *Langages et Modèles à Objets* (LMO 94) de Grenoble [Rec94], des workshops *Object-Based Representation Systems* de IJCAI 93 [Nap93] et *Integrating Object-Orientation and Knowledge Representation* de ECAI 94 [CdCK94].

Par ailleurs, il y a déjà eu de nombreuses tentatives, plus ou moins abouties, de comparaison entre LPO et RCO : [Neb85] compare les FLAVORS avec les langages de *frames* ; [Kai94b] compare les approches objet respectives de l'intelligence artificielle et du génie logiciel, et [Kai94a] note l'analogie entre méthodologie de conception objet et l'acquisition des connaissances, cette dernière discipline se préoccupant de plus en plus de modélisation. Toutefois, ces tentatives restent superficielles et s'intéressent plus aux aspects de programmation qu'aux aspects sémantiques de la RCO.

Ultime avertissement : en partant d'une question simple et pragmatique, beaucoup de grands débats de l'intelligence artificielle, voire de philosophie, peuvent surgir au détour de la phrase. Que le lecteur soit indulgent devant le nombre de portes ouvertes que nous aurons enfoncées...

L'article est organisé comme suit. Une première partie développe le tronc commun des systèmes à objets, se penche sur la notion de sémantique et décrit la sémantique ontologique de Wand (thèse). La seconde partie développe les arguments visant à prouver la différence entre LPO et RCO et se penche sur la controverse classique déclaratif-procédural dans le contexte plus spécifique des objets (antithèse). Une troisième partie développe les arguments pour l'intégration (synthèse). L'article se termine par des perspectives de travail.

2 Un modèle objet commun

Le succès des objets en programmation (LPO) vient sans doute du fait qu'ils permettent de (ou obligent à) faire de la représentation avant de faire de la programmation, quitte à ce que se soit comme monsieur Jourdain faisant de la prose. L'explication du phénomène semble tenir à la correspondance immédiate, quasi isomorphique, que l'on peut établir entre un modèle objet très général et une approche philosophique — *ontologique* [1] — de la réalité.

2.1 Les concepts de base communs

Tous les systèmes à objets font appel à des entités individuelles — les *objets* — possédant des propriétés — les *attributs* — regroupées dans des entités génériques — les *classes* — ces classes étant elles-mêmes hiérarchisées par une relation de *spécialisation* permettant un *héritage* des propriétés et une *classification* des entités, instances ou classes. La classe décrit les propriétés de ses instances — de façon générale en spécifiant leur domaine de valeur, le type des valeurs, la cardinalité, etc. — et hérite, éventuellement en les affinant, des propriétés des classes qu'elle spécialise.

Les objets sont instances d'une classe, éventuellement de plusieurs, mais c'est atypique. Les objets possèdent donc les propriétés décrites dans la classe directement ou par héritage, avec des valeurs respectant les contraintes imposées dans la classe.

Ce modèle repose sur deux postulats : 1) on peut décrire des classes, 2) on peut instancier une classe, c'est-à-dire créer un objet instance d'une classe donnée, ou, plus généralement, il est possible de créer des objets et de les attacher à des classes. Il est commun à toutes les approches objet qui nous intéressent ici, à savoir LPO, RCO et LT, mais aussi aux bases de données orientées objet et aux méthodologies de conception objet.

Remarques

- Nous faisons ici délibérément l'impasse sur les systèmes de *prototypes* comme par exemple [US87] ou [MGVdZ92], pour deux raisons :
    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 [DH89, DHH$^+$94].
- On peut discuter de la compatibilité des LT avec le modèle classe-instance proposé ici. Il semble que les seules différences soient :
    1. Les *termes* ou *concepts* (= classes) des LT sont constitués à partir de *rôles* (= propriétés), et toute combinaison de rôles est un terme (au moins potentiel),

1. Le terme d'*ontologie* est utilisé depuis quelques années, en général au pluriel, et en particulier par les anglo-saxons, pour désigner des modélisations de portions du *monde réel* : l'espace, le temps, la composition, etc. [LG90, GL90, PFPS$^+$92]. 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 [Lal26]).

alors que les classes de l'approche objet sont premières et décrites, après, par des propriétés.

2. Les LT sont complètement dynamiques : il est possible à tout moment de rajouter des propriétés à une instance, ce qui est exactement équivalent à un reclassement de l'instance (peut-être dans une classe qui ne préexistait pas). La différence ne viendrait pas de l'absence de "structure" des objets des LT, mais de leur extrême flexibilité.

2.2 Trois sémantiques de sémantique

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 (voir par exemple [GM94]) ne favorise pas la dissémination de ces résultats, surtout dans la communauté intelligence artificielle. Cependant, la recherche d'une sémantique "propre" est considérée par beaucoup comme déterminante. Nous allons donc commencer par nous attacher, naïvement, à discuter de ce que peut être une telle 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 plusieurs façons courantes d'établir une telle signification :

1. une **sémantique opérationnelle** offre un modèle de fonctionnement d'un système sans forcément faire 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. L'héritage multiple traité par les techniques de linéarisation en est un exemple [DH89, DHH$^+$94].

2. une **sémantique par équivalence** 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* [Tho92, DHS94], le domaine cible étant alors une logique non-monotone.

3. une **sémantique dénotationnelle** 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 [DMO93] et de [CF94] où le domaine est une algèbre universelle —, soit être un ensemble "plat", sans structure particulière — c'est le cas des sémantiques usuelles des LT, par exemple celle de [BPS94] —, soit enfin être le *domaine modélisé*. Dans ce dernier cas, on l'appellera une **sémantique directe** puisque l'interprétation a pour cible la réalité [2].

Cette classification est certainement discutable, au moins dans sa terminologie. Faut-il réellement distinguer sémantique par équivalence et dénotationnelle ? Existe-il une solution

2. C'est un peu rapide et il faut préciser. On commence par construire un *modèle* (au sens de "modèle mathématique" ou de "modèle réduit") 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. L'approche "par approximation" introduite dans [Smo92, AKP93] rend pleinement compte de cette séparation. Cette remarque sur les modèles s'applique également à la sémantique usuelle associée au calcul des prédicats du premier ordre.

de continuité entre sémantique directe et sémantique par équivalence ? Du côté intelligence artificielle, [Win84] distingue lui aussi trois types de sémantique — *procedural, equivalence and descriptive semantics* — qui correspondent assez bien aux nôtres. Du côté génie logiciel, [Mey90] en distingue une quatrième, la *sémantique axiomatique* ; et la définition (orthodoxe) de la sémantique dénotationnelle donnée dans [Mos90] précise que la fonction d'interprétation est à valeur dans un domaine mathématique qui est en général un ensemble de fonctions, ce qui exclut qu'il puisse s'agir d'une sémantique directe.

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. 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 ?

Les LT offrent une variation sur ce thème, avec leur sémantique *intensionnelle* [3] ou *extensionnelle* [Woo91] : 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.

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 [WS92], les logiques terminologiques sont nées de la volonté de donner une assise sémantique solide aux langages de frames et aux réseaux sémantiques.

2.3 Une sémantique directe des objets

Pour remédier à l'absence de sémantique claire des objets, [Wan89] propose un modèle formel des objets orienté vers la modélisation applicative et basé sur l'ontologie de [Bun77, Bun79].

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 de la mécanique quantique, — à savoir que deux individus distincts doivent différer par au moins une propriété — et un principe à la Lavoisier — *rien ne se perd, rien ne se crée, tout se transforme* — que ne renierait aucun *garbage collector*. Wand traduit ces notions de base, respectivement, en *objets*, *attributs* et *contraintes* portant sur les attributs, ce qui lui donne directement un modèle d'objets non typés. Il définit alors 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 spécifications des rôles.

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

3. La communauté informatique 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 [Lal26] où il est décrit comme ancien avec le sens de *compréhension*. Il est donc donc normal d'utiliser *intension* ou le néologisme *intensionnel*, et ce n'est pas un anglicisme.

l'interaction entre objets — il se restreint au cas binaire en le prétendant suffisamment général — ou leur évolution. Ce qui nous intéresse ici est moins le modèle lui-même que son existence, qui est révélatrice du caractère profondément intuitif du modèle objet : les catégories du philosophe sont très proches des catégories de l'approche objet.

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 cela de la RCO. Il faut cependant bien souligner qu'il s'agit d'une sémantique du tronc commun aux diverses approches objet et non pas d'un tronc commun aux sémantiques de ces diverses approches.

3 L'irréductibilité des approches LPO et RCO

La distinction entre RCO et LPO passe par diverses oppositions :

- 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. La consultation des connaissances peut constituer le seul objectif d'une application comme le montre [GR92], et, à la limite, l'organisation hiérarchique des objets résultant du processus de modélisation d'un domaine peut être l'unique retombée du développement d'une base de connaissances [Rec93a].
- Opposition entre l'*être* et le *faire* : une base de connaissances représente un état d'un *domaine modélisé*, mais la résolution de problèmes nécessite d'agir sur cet état.
- Opposition entre *déclaratif* et *procédural* : cette controverse classique a accompagné la naissance de la RCO [Win75], et les termes n'en ont pas beaucoup évolué.
- La RCO semble susceptible d'être dotée d'une sémantique directe alors que cela ne semble pas être le cas pour les LPO [Euz94].

3.1 Les notions divergentes

LPO et RCO diffèrent principalement sur les capacités procédurales qu'ils s'accordent et sur la sémantique exacte qu'ils exigent des classes.

3.1.1 Procédures et encapsulation

Les procédures des LPO sont les *méthodes* qui sont activées par le mécanisme appelé, *envoi de message*[4] : 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.

Par ailleurs, la plupart des 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 [Mey88] [Sny91]. Dans ce contexte

4. La métaphore de l'envoi de message vient de SIMULA [BDMN73], où les objets étaient des *coroutines* dont l'asynchronisme justifiait la métaphore. Reprise par SMALLTALK [GR83], la métaphore a perdu tout son sens en même temps que les objets perdaient leur asynchronisme. Les acteurs et autres objets concurrents la justifient à nouveau.

un objet se définit par l'ensemble des services qu'il assure et qui constituent son interface (ses spécifications). Les attributs sont considérés ici comme la structure d'implémentation de l'objet sur laquelle repose la réalisation de ces services. Conformément au principe de séparation spécification – implémentation, les attributs sont donc privés (encapsulés) dans l'objet. La RCO quant à elle confère aux attributs un statut à part entière d'éléments de connaissance : l'objet se définit avant tout par les attributs qui le caractérisent. 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. La RCO éprouve donc peu d'intérêt vis-à-vis de l'encapsulation, sauf quand il s'agit d'implémenter un système de RCO avec un LPO.

3.1.2 Procédures : méthodes ou réflexes

La RCO a été identifiée à l'origine aux langages de *frames* [Min75, Win75, MNC+89] : dans ces langages, les procédures figuraient sous la forme d'*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* [BS83].

On a appelé par la suite *langages hybrides* les langages faisant la synthèse entre langages à objets et langages de frames. On peut bien sûr comparer méthodes et réflexes [Neb85], conclure éventuellement des FLAVORS ou de CLOS à leur équivalence, ou au contraire la réfuter. Il semble néanmoins que l'accord se fasse sur le fait : 1) qu'il s'agit de deux formes de programmation complémentaires et aussi désirables l'une que l'autre, même si leurs emplois respectifs diffèrent, 2) que les réflexes, pas plus que les méthodes, ne participent directement d'un modèle déclaratif de représentation des connaissances.

Il s'ensuit en particulier que les langages de frames ou les langages hybrides qui ont longtemps prétendu faire de la représentation des connaissances doivent plutôt être considérés comme des langages de programmation plus évolués et plus déclaratifs que les LPO. C'est le cas par exemple de FROME [Dek94] et de YAFOOL [Duc91].

Il faut cependant reconnaître aux réflexes le fait qu'ils constituent un mécanisme de déclenchement de procédures dont la sémantique opérationnelle est complètement spécifiée [Duc91, Nap92] : un réflexe n'est jamais appelé de façon impérative. En revanche, 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.

3.1.3 Classification et filtrage

La RCO offre (en général) deux fonctionnalités particulières assez voisines : le *filtrage* qui permet d'interroger une base de connaissances objet pour connaître tous les objets qui satisfont des contraintes données (exprimées par un filtre constitué par un ensemble de couples attribut-valeur) et la *classification* qui permet de connaître toutes les classes dont une entité individuelle (ou générique) satisfait toutes les contraintes [Euz93b].

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 [Bra85] [5], et que les propriétés soient vues

5. Bien que l'on cherche de plus en plus à se libérer de cette exigence de monotonie [CF94, PN93].

comme des conditions nécessaires et suffisantes (CNS), ce qui est le cas dans les LT où la classification a été étudiée de manière intensive [Neb90].

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) [Euz93a]. Le premier problème a été traité en SHIRKA [Rec93b], TROPES [Mar93] ou (F)ROME [Dek94], 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*). 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. On peut en fait assimiler chaque classe de la RCO à un concept *primitif* des LT, l'originalité de la RCO étant de chercher à classer dans des concepts primitifs, de façon interactive bien sûr, puisque la classification automatique nécessite des conditions suffisantes. De son côté, [Dek94] propose une expression plus fine des CN et CNS pour les divers attributs des classes : certains sont suffisants, d'autres pas.

3.1.4 Points de vue et autres besoins de représentation

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* [Car89, Mar93, Dek94, RR93], qui remplissent les mêmes fonctions que la *coréférence* de [FV88, Fer89], les *facettes* de [PW92], ou encore les *vues* ou *rôles* des bases de données orientées objet [AB91, SLT91].

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 [Car89, Dek94] que TROPES [Mar93] proposent un modèle objet atypique, dans lequel la taxonomie des classes n'a pas un caractère ontologique [Euz93a] ;

- ces besoins sont aussi présents en LPO [CG90] [AR92] ; le cas de [PW92] est à la frontière RCO-LPO, et on doit bien aussi trouver l'expression de ces besoins dans les méthodologies de conception objet ;

- les choix de représentation ne sont certainement pas sans effet sur la sémantique directe : ainsi, par définition, la coréférence consiste à rendre la fonction d'interprétation non injective.

La RCO se pose bien d'autres problèmes de représentation, en premier lieu l'étude des *relations*, dont celle(s) dite(s) de composition [WCH87] [Nap92]. Les relations sont aussi à la base de la plupart des méthodologies de conception objet. Ce n'est pas l'objet de cet article, mais il faut noter néanmoins une limitation fondamentale du modèle objet : son incapacité à traiter directement des relations d'arité strictement supérieure à deux. Il est nécessaire, soit de *réifier* ces relations, dans une approche tout à fait comparable aux réseaux sémantiques ou aux graphes conceptuels [Sow84], soit d'utiliser, conjointement avec les objets, un modèle relationnel, par exemple à base de règles comme le proposent bon nombre d'environnements de développement de systèmes à base de connaissances. [Cas91] 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. Notons pour terminer que la problématique des relations d'arité supérieure à deux et celle de la réification des relations ont fait l'objet de travaux dans le domaine des LT [Sch89, Mac93].

3.2 L'opposition déclaratif – procédural

Cette controverse est certainement inépuisable, les deux positions extrêmes déjà notées par Winograd étant : "tout est déclaratif" ou "tout est procédural" [Win75]. 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 [6] :

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), examinées et appréhendées isolément. Cette indépendance est bien sûr relative : Winograd cite à ce propos les "systèmes presque décomposables" de Simon [Sim69], dont la propriété de décomposabilité est en fait nécessaire à la compréhension de n'importe quel système.

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 fondamental.

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 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.

6. On trouve dans la littérature bien d'autres définitions, certaines plus contestables que d'autres. [Hor94] semble ainsi réduire le procédural à ce qui dépend de l'implémentation, rendant ainsi déclaratif tout algorithme bien spécifié. [MGVdZ92] place quant à lui la déclarativité dans l'absence de méthodes pour le langage de prototype GARNET

3.3 Être et faire

Le modèle objet se prête ainsi particulièrement bien à représenter un état. Si, de leur côté, les LPO se prêtent bien à implémenter un comportement, c'est sous une forme procédurale qui n'a pas de sémantique directe.

3.3.1 Qu'est-ce que faire ?

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* : l'état du domaine modélisé est entièrement déterminé par l'ensemble des connaissances comprises dans la base, mais il n'est pas entièrement connu ; on cherche alors à déduire par inférences de nouvelles informations ;
- à la *compléter par interrogation* : l'état du domaine modélisé n'est pas entièrement déterminé par l'ensemble des connaissances comprises dans la base ; il faut donc acquérir les informations qui manquent auprès d'un utilisateur ou d'une base de données ; il faut souligner que les deux modes de complétion qui viennent d'être évoqués peuvent coexister ;
- à la *mettre à jour* : le domaine modélisé 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.

3.3.2 Ce qu'il ne faut pas faire

Il est tout à fait significatif que le principal mécanisme que s'autorisent les orthodoxes en RCO est la classification d'instances, qui permet de "reconnaître" un objet pour ce qu'il "est", sans doute depuis toujours mais sans le savoir. La seule autre manière déclarative connue de modifier des objets consiste à poser des contraintes sur des attributs et à résoudre le système de contraintes résultant (CSP)[7].

Pour les *orthodoxes*, la RCO devrait limiter son usage du procédural à des calculs qui respectent la sémantique du système de représentation. Aucun des trois types d'effets de bord cités plus haut ne devrait être autorisé autrement que par des mécanismes spécifiques

7. Il faut noter qu'un "vrai" CSP (*constraint satisfaction problem*) — tel que défini dans [Mac77] ou [Kök94] par exemple — a une sémantique parfaitement déclarative (à condition d'en considérer l'ensemble des solutions), ce qui n'est pas le cas des systèmes de contraintes à la GARNET [MGVdZ92] 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 6).

régis par le système et respectant sa sémantique. Si une procédure calcule l'âge du capitaine, ce n'est pas elle qui est censée modifier l'objet, mais le mécanisme qui la déclenche et qui doit faire partie intégrante de la sémantique. Le procédural devrait ainsi être restreint à du *fonctionnel pur* du point de vue des objets de la RCO [8].

3.4 La représentation représente (et la programmation programme ?)

L'approche utilisée en logique classique, comme dans les LT, pour établir une sémantique dénotationnelle des langages utilisés consiste à définir ce qui est appelé ici une sémantique directe. Cette sémantique interprète donc le langage sur un domaine qui est (une approximation) du domaine modélisé (voir note 2). En général, une telle approche ne fait pas appel à des objets mathématiques n'appartenant pas au domaine modélisé, des fonctions par exemple. En réalité, l'intérêt de l'approche réside dans l'intelligibilité immédiate de la sémantique : puisque ce qui est modélisé par un objet n'est généralement pas une fonction, il n'y a pas de raison pour que la l'interprétation d'un objet en soit une.

Cette précision est importante : la définition d'une sémantique dénotationnelle directe est le fondement même de la représentation de connaissances, et un système de représentation de connaissance doit se comporter en accord avec le domaine modélisé. C'est vers cette approche que tend — sans y être forcément arrivée pour l'instant — la RCO, et c'est cet objectif qui oppose, semble-t-il radicalement, les RCO et LPO.

La sémantique de Wand (voir § 2.3) n'est pas suffisamment aboutie pour servir de sémantique pour les LPO, comme il est difficile de la qualifier de sémantique pour les RCO, car trop incomplète également : établir la sémantique d'un langage consiste à donner l'interprétation de *tous* ses composants, et, dans [Wan89], rien n'est dit sur l'interprétation qui doit être accordée aux messages ou bien aux métaclasses par exemple. Les approches utilisées, en général, pour donner une sémantique aux LPO se font en termes de sémantique opérationnelle ou de sémantique dénotationnelle. Dans ce dernier cas, il n'est généralement pas question de sémantique directe car les objets sont modélisés par des fonctions [CM91] ou des clôtures [Red88].

Si la différence résidait dans une opposition dénotationnel – opérationnel, le fossé pourrait être comblé : ce ne sont que deux manières différentes de voir le même phénomène. Mais certaines caractéristiques des LPO rendent peu probable la possibilité d'établir une sémantique directe. En effet, l'importance accordée à l'extensibilité des LPO a conduit à la définition d'un ensemble de concepts minimaux, mais extrêmement puissants parce que d'ordre supérieur, permettant de modifier le comportement global du système (par exemple, redéfinition de la stratégie d'héritage, création de métaclasses, etc.). Ceci permet effectivement l'extensibilité, mais a deux conséquences importantes sur le plan de la sémantique de ces systèmes :

- Il est nécessaire de tracer la frontière entre les objets qui font partie du domaine modélisé (dénotant un individu particulier) et ceux qui font partie de la machinerie du système (ceci est criant lorsque les piles, fenêtres, programmes et messages sont eux-mêmes des objets [GR83]).

- La signification d'une classe comme dénotant de manière non ambiguë une entité du domaine modélisé (en général un ensemble d'individus) ne peut plus être éta-

8. Mais pas forcément du point de vue des objets du LPO : une procédure doit pouvoir faire des effets de bord sur les objets de service des interfaces graphiques par exemple.

blie clairement dès lors qu'il est possible de définir des classes ayant leurs propres métaclasses dans lesquelles les comportements de base ont été redéfinis (protocole d'héritage par exemple).

Toujours dans les systèmes réflexifs, de simples ensembles ne suffisent plus à définir la dénotation des classes, en particulier celle des classes qui se décrivent elles-mêmes. En effet, la "première" métaclasse, la "racine" du graphe d'instanciation – spécialisation, est sa propre instance. Elle fait donc référence à un ensemble qui se contient lui-même, dont il est difficile d'appréhender la dénotation, mais sa réalité même exclut tout paradoxe de Russell. Bien entendu, l'étape suivante dans cette discussion dialectique consiste à remarquer que le domaine modélisé lui-même pourrait être capable d'auto-représentation (et même d'auto-modification) mais ceci nous mènerait trop loin dans l'état actuel de l'étude sémantique de ces systèmes.

Pour résumer, dans l'état actuel des choses, il n'est pas possible d'associer une sémantique directe aux LPO : en général, les LPO sont conçus comme de (très) puissants langages de programmation et ils laissent une grande latitude au programmeur ; ne pas vouloir se priver de cette puissance signifie ne pas être en mesure de se doter d'une sémantique directe et donc ne pas pouvoir se mettre sur le même plan sémantique que les RCO. En revanche, la restriction d'un LPO à un noyau minimal pourrait être dotée d'une sémantique directe, comme celle qui est présentée dans le paragraphe 2.3, mais qui ne tiendrait pas compte de nombreux traits du langage comme les métaclasses et l'envoi de messages par exemple.

On pourrait conclure en disant qu'une RCO peut être vue comme la moitié du verre qui est pleine : la partie d'un LPO qui peut être dotée d'une sémantique directe si l'on supprime le reste, ce qui laisse de la place pour remplir le verre ultérieurement.

4 Vers une intégration LPO – RCO

Résumé des épisodes précédents : tout le monde s'accorde à reconnaître qu'il est possible d'*implémenter* une RCO dans un LPO. Il est alors raisonnable de rechercher une *intégration* d'une RCO dans un LPO, à condition d'en définir les modalités, ce qui sera fait plus loin. Il faut souligner qu'il s'agit là d'une opération asymétrique : bien que cela soit possible, et sans doute souhaitable, nous ne nous intéressons pas ici à l'intégration d'aspects propres à la RCO, comme l'utilisation de la classification par exemple, dans un LPO.

4.1 L'être du faire

Finalement, tout le monde s'accorde sur les constats contraposés suivants :

- l'*être* peut et doit être représenté par du *déclaratif* ;
- le *procédural* doit représenter du *faire*.

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 par les systèmes à base de connaissances comme l'explication du raisonnement par exemple.

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

- Les systèmes de production, qui sont considérés comme tels par [Win75]. À 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 Shirka [GG92] [Wil94] sont une façon déclarative de représenter les procédures.
- Le système expert DIVA [DK87, DK88] 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 [Cas87].
- F. Rechenmann avait exposé lors du workshop *Trends in Knowledge Representation* à l'occasion du 25e anniversaire de l'INRIA la façon dont il voyait la représentation des méthodes de calcul de valeurs manquantes [Rec92].
- [KMMPN87, Bor81] présentent aussi des propositions pour la classification de procédures ou de méthodes.

D'une façon générale, il s'agit 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. Les orthodoxes soulignent : à la *sémantique directe* bien spécifiée !

À leur échelle, les réflexes étaient une première tentative, modeste, de solution à ce problème, mais avec une sémantique opérationnelle.

4.2 Vers l'intégration

Une façon assez classique d'envisager l'intégration de la RCO dans le LPO consiste à étendre un LPO au niveau méta pour définir les classes nécessaires : les objets de la RCO sont alors vus comme un sous-ensemble des objets du LPO. C'est ce qu'ont fait Lenneke Dekker en construisant FROME [DC92, Dek94] à partir de ROME [Car89], ou [Rat91] et [DB93] au-dessus de CLOS [Ste90].

4.2.1 Qu'est ce qu'intégrer ?

L'intégration se caractériserait, 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 des objets de la RCO sont un sous-ensemble des attributs du LPO, mais les objets de la RCO peuvent avoir des attributs (de service par exemple) qui soient 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 ;

3. les fonctionnalités du LPO sont applicables aux objets (resp. classes) de la RCO *tant qu'elles n'altèrent pas la sémantique* propre à 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 extension du LPO.

4.2.2 Pourquoi intégrer ?

Il n'est pas inutile de revenir sur l'intérêt de l'intégration : unification conceptuelle d'une part, extensibilité d'autre part ou plus généralement *héritage* de toutes les bonnes propriétés du LPO [CD91].

Outre l'élégance d'un modèle unifié, qui est du même ordre que le modèle uniforme instance-classe-métaclasse d'OBJVLISP [Coi87], les *œcuménistes* mettent en avant la possibilité de définir une famille ouverte de systèmes de RCO. Leur but pratique serait de construire une plate-forme commune que chaque chapelle pourrait étendre à sa guise et qui permettrait une confrontation des expériences.

De plus, les *œcuménistes* un tantinet explorateurs rêvent d'amener les classes de la RCO au cœur de la circularité du niveau méta : ils soutiennent qu'il n'y a pas de raison d'exclure *a priori* du domaine représenté les concepts qui servent à le représenter (mais d'autres répondent qu'il n'y a pas plus de raisons, *a priori*, de les y mettre...).

4.2.3 Comment intégrer ?

Les tentatives d'intégration précitées reposent sur la définition de nouvelles classes et métaclasses — voire de méta-métaclasses dans le cas de FROME — auxquelles il faut en général ajouter des classes d'attributs pour obtenir un langage de frames typique. LORE [Cas87], ou [Fer89] dans l'un des avatars de MERING, utilisent des techniques assez semblables.

Dans ce contexte, les *orthodoxes* un tantinet autoritaires exigent une séparation physique inviolable entre RCO et LPO — que les *pessimistes* ne croient d'ailleurs pas possible. Les *œcuménistes optimistes*, mais pas si libertaires que ça, proposent un compromis :

1. Il est possible de définir deux modes étanches : un mode LPO qui permet de construire ou d'étendre (au niveau méta) un RCO existant et un mode RCO qui permet de construire une base de connaissances. Ces deux modes seraient garantis non miscibles. Cela ne présente aucune difficulté si LPO et RCO sont conçus ensemble, mais peut être plus difficile si le LPO est préexistant.

2. Il faut fournir un protocole strict d'utilisation du niveau LPO méta, qui garantisse la constance de la sémantique de la RCO.

Ces deux possibilités peuvent coexister, la deuxième ne concernant que le mode RCO de la première. La difficulté n'est pas, bien sûr, de réussir à le faire, mais de le faire avec une généralité suffisante. Il est certain qu'il n'y a pas de raison de limiter l'usage du LPO pour tout ce qui relève de l'interaction entre les connaissances du système et le monde extérieur — utilisateur ou autre : la visualisation des objets, les interrogations à l'utilisateur doivent pouvoir être redéfinies sans limitation particulière. Il est non moins certain que ce qui touche directement à la sémantique de représentation — l'instanciation par exemple, pour reprendre un point sensible pour les *orthodoxes* — doit être particulièrement surveillé.

On peut conclure que la majorité semble d'accord pour refuser le label "pur RCO" aux systèmes "intégrés" précités [Fer89, Rat91, DB93, Dek94] car 1) la RCO y baigne dans le LPO sans frontière définie, 2) la RCO n'y a pas de sémantique propre. À la limite, existe-il un seul représentant honorable de la RCO ? Pour les *orthodoxes*, sans doute pas. À l'aune des exigences exprimées ici, AIRELLE [NA90], (F)ROME, MERING(S) [Fer89], YAFOOL, etc. ne sont que des LPO particulièrement expressifs.

5 Conclusions et Perspectives

La difficulté de la métaphore du verre à moitié vide ou à moitié plein est qu'elle mène à un dilemme naturel — le remplir ou le vider — ou à un blocage qui n'est pas sans rappeler l'âne de Buridan : remplir le verre, c'est intégrer au risque d'y perdre l'âme de la RCO ; vider le verre, c'est se replier sur une RCO puriste, au risque d'y perdre son caractère pratique.

Cet article aura soulevé plus de problèmes qu'il n'avait l'intenTion de le faire, sans en résoudre beaucoup. La représentation des connaissances par objets n'existe sans doute pas plus à l'état pur que la déclarativité. C'est à la fois une intenTion et une intenSité : *degré 6 sur l'échelle de* RCO. Ou un quark...

Dans la pratique, trois perspectives s'ouvrent aux membres du groupe, collectivement ou plus vraisemblablement par sous-groupes :

- poursuivre l'effort de formalisation et d'implémentation du modèle de RCO multi-points de vue TROPES [Mar93] et commencer à en envisager son application ;
- poursuivre l'effort de formalisation, d'implémentation et d'application du langage de LPO-RCO multi-points de vue (F)ROME [Car89, Dek94] ;
- poursuivre la réflexion autour de l'intégration LPO-RCO(-LT) et commencer à en envisager la réalisation.

Il faut d'ailleurs poursuivre l'étude comparée RCO-LT [Euz93a], dont nous avons montré, en bons avocats du diable, la nécessité. La RCO est en effet prise entre le marteau des LPO et l'enclume des LT : ni l'intérêt de la RCO, ni l'existence d'une sémantique différente de celle qui est associée classiquement aux LT ne sont claires et irréfutables.

Références

[AB91] S. Abiteboul and A. Bonner. Objects and Views. In *Proceedings of the ACM/SIGMOD International Conference on the Management of Data, Denver, Colorado, SIGMOD RECORD, 20(2)*, pages 238–247, 1991.

[AK91] H. Aït-Kaci. A Glimpse of Paradise. In J.W. Schmidt and A.A. Stogny, editors, *Next Generation Information System Technology*, pages 15–25. Springer-Verlag, 1991.

[AKP93] H. Aït-Kaci and A. Podelski. Towards a Meaning of LIFE. *The Journal of Logic Programming*, 16(3–4):195–234, 1993.

[AR92] E.P. Andersen and T. Reenskaug. System Design by Composing Structures of Interacting Objects. In *Proceedings of ECOOP'92, Utrecht, The Netherlands*, Lecture Notes in Computer Sciences 615, pages 133–152, 1992.

[BDMN73] G. Birtwistle, O.J. Dahl, B. Myhrhaug, and K. Nygaard. SIMULA *begin*. Petrocelli Charter, 1973.

[Bor81] A. Borgida. On the Definition of Specialization Hierarchies for Procedures. In *Proceedings of the 7th IJCAI, Vancouver, Canada*, pages 254–256, 1981.

[BPS94] A. Borgida and P.F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[Bra85] R. Brachman. "I Lied about the Trees" or, Defaults and Definitions in Knowledge Representation. *The AI Magazine*, 6(3):80–93, 1985.

[BS83] D.G. Bobrow and M. Stefik. The LOOPS Manual: A Data and Object Oriented Programming System for Interlisp. Knowledge-Based VLSI Design Group Memo KB-VLSI-81-13, Xerox Parc, Palo Alto, California, 1983.

[Bun77] M. Bunge. *Ontology I: The Furniture of the World*, volume 3 of *Treatise on Basic Philosophy*. Reidel, Boston, 1977.

[Bun79] M. Bunge. *Ontology II: A World of Systems*, volume 4 of *Treatise on Basic Philosophy*. Reidel, Boston, 1979.

[Car89] 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 de Lille, 1989.

[Cas87] Y. Caseau. *Étude et réalisation d'un langage objet :* LORE. Thèse d'informatique, Université Paris-Sud (Orsay), 1987.

[Cas91] Y. Caseau. An Object-Oriented Deductive Language. *Annals of Mathematics and Artificial Intelligence*, 3:211–258, 1991.

[CD91] B. Carré and L. Dekker. Inheriting object-oriented features through metaprogramming. In *Proceedings of the First East European Conference on Object-Oriented Programming (EurOOP'91), Bratislava*, pages 126–135, 1991.

[CdCK94] R. Cunis, D. de Champeaux, and H. Kaindl, editors. *ECAI'94 Workshop on Integrating Object-orientation and Knowledge Representation*, 1994.

[CF94] P. Coupey and C. Fouqueré. Classer des concepts définis avec des défauts et des exceptions. In *Actes de la conférence Langages et Modèles à Objets (LMO'94), Grenoble*, pages 69–80, 1994.

[CG90] B. Carré and J.-M. Geib. The Point of View Notion for Multiple Inheritance. In *Proceedings of OOPSLA-ECOOP'90, Ottawa, Canada, ACM SIGPLAN Notices (25)10*, pages 312–321, 1990.

[CM91] L. Cardelli and J. Mitchell. Operations on records. *Mathematical Structures in Computer Science*, 1(1):3–48, 1991.

[Coi87] P. Cointe. Metaclasses are First Class: The ObjVlisp Model. In *Proceedings of the 2nd OOPSLA, Orlando, Florida, ACM SIGPLAN Notices 22(12)*, pages 156–167, 1987.

[DB93] J. Dvorak and H. Bunke. Using CLOS to Implement a Hybrid Knowledge Representation Tool. In A. Paepcke, editor, *Object-Oriented Programming – The* CLOS *Perspective*, pages 295–320. The MIT Press, Cambridge, Massachusetts, 1993.

[DC92] L. Dekker and B. Carré. Multiple and dynamic representation of frame with point of view in FROME. In *Actes de la conférence Représentations Par Objets (RPO'92), La Grande Motte*, pages 97–111, 1992.

[Dek94] L. Dekker. FROME *: représentation multiple et classification d'objets avec points de vue*. Thèse d'informatique, Université de Lille, 1994.

[DH89] R. Ducournau and M. Habib. La multiplicité de l'héritage multiple. *Technique et Science Informatiques*, 8(1):41–62, 1989.

[DHH+94] R. Ducournau, M. Habib, M. Huchard, M-L. Mugnier, and A. Napoli. Le point sur l'héritage multiple. *Technique et Science Informatiques*, 1994. (à paraître).

[DHS94] R. Ducournau, M. Habib, and G. Simonet. Méta-héritage et héritage non-transitif. *Revue d'Intelligence Artificielle*, 1994. (à paraître).

[DK87] J.-M. David and J.-P. Krivine. Utilisation de prototypes dans un système expert de diagnostic : le projet DIVA. In *Actes 7e Journées Internationales "Les systèmes experts et leurs applications"*, pages 889–907, Avignon, 1987.

[DK88] J.-M. David and J.-P. Krivine. Acquisition de connaissances expertes à partir de situations types. In *Actes 8e Journées Internationales "Les systèmes experts et leurs applications"*, pages 45–58, Avignon, 1988.

[DMO93] R. Dionne, E. Mays, and F.J. Oles. The equivalence of model-theoretic and structural subsumption description logics. In *Proceedings of the 13th IJCAI, Chambéry, France*, pages 710–716, 1993.

[Duc91] R. Ducournau. Y3 : YAFOOL*, le langage à objets*. Sema Group, 1991.

[Euz93a] J. Euzenat. On a purely taxonomic and descriptive meaning for classes. In *Proceedings of the IJCAI'93 Workshop on Object-Based Representation Systems*, pages 81–92, 1993. The proceedings are available as Rapport de Recherche CRIN 93-R-156, Nancy, France.

[Euz93b] J. Euzenat. Une définition abstraite de la classification et son application aux taxonomies d'objets. In *Actes de la conférence Représentations Par Objets (RPO'93), La Grande Motte*, pages 235–246, 1993.

[Euz94] J. Euzenat. KR and OOL co-operation based on semantics non reducibility. In *Proceedings of the ECAI Workshop on Integrating Object-Orientation and Knowledge Representation, Amsterdam*, 1994.

[Fer89] J. Ferber. *Objets et agents : une étude des structures de représentation et de communication en intelligence artificielle.* Thèse d'État, Université Paris 6, 1989.

[FV88] J. Ferber and P. Volle. Using Coreference in Object-Oriented Representations. In *Proceedings of the 8th ECAI, Munich, Germany*, pages 238–240, 1988.

[GG92] J. Gensel and P. Girard. Expression d'un modèle de tâches à l'aide d'une représentation par objets. In *Actes de la conférence Représentations Par Objets (RPO'92), La Grande Motte*, pages 225–236, 1992.

[GL90] R.V. Guha and D.B. Lenat. Cyc: A Mid-Term Report. *The AI Magazine*, 11(3):33–59, 1990.

[GM94] C. A. Gunter and J. C. Mitchell, editors. *Theoretical Aspects of Object-Oriented Programming: Types, Semantics, and Language Design.* The MIT Press, 1994.

[GR83] A. Goldberg and D. Robson. *Smalltalk-80, the Language and its Implementation.* Addison-Wesley, Reading, Massachusetts, 1983.

[GR92] S. Grivaud and F. Rechenmann. Navigation dans les bases de connaissances associant objets et hypertextes. In *Actes de la conférence Représentations Par Objets (RPO'92), La Grande Motte*, pages 269–280, 1992.

[HO92] M. Habib and M. Oussalah, editors. *Actes des journées Représentation Par Objets, RPO'92*, La Grande Motte, 1992. EC2.

[HO93] M. Habib and M. Oussalah, editors. *Actes des journées Représentation Par Objets, RPO'93*, La Grande Motte, 1993. EC2.

[Hor94] J.F. Horty. Some Direct Theories of Nonmonotonic Inheritance. In D. Gabbay and C. Hogger, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 2: Nonmonotonic Reasoning.* Oxford University Press, 1994.

[Kai94a] H. Kaindl. Comparing object-oriented analysis with knowledge acquisition. *OOPS Messenger*, 5(3):1–5, 1994.

[Kai94b] H. Kaindl. Object-oriented approaches in software engineering and artificial intelligence. *Journal of Object-Oriented Programming*, 6(8), 1994.

[KMMPN87] B.B. Kristensen, O.L. Madsen, B. Møller-Pedersen, and K. Nygaard. Classification of Actions or Inheritance also for Methods. In *Proceedings of ECOOP'87, Paris (Special issue of Bigre 54 and Lecture Notes in Computer Science 276)*, pages 109–118, 1987.

[Kök94] T. Kökény. *Satisfaction de contraintes dans un environnement orienté-objets.* Thèse d'informatique, Université des Sciences et Techniques du Languedoc, Montpellier, 1994.

[Lal26] A. Lalande. *Vocabulaire technique et critique de la philosophie.* Presses Universitaires de France, Paris, 1926.

[LG90] D.B. Lenat and R.V. Guha. *Building Large Knowledge-Based Systems – Representation and Inference in the Cyc Project.* Addison-Wesley, Reading, MA, 1990.

[Mac77] A.K. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8:99–118, 1977.

[Mac93] R.M. MacGregor. Representing reified relations in Loom. *Journal of Experimental & Theoretical Artificial Intelligence*, 5(2&3):179–183, 1993.

[Mar93] O. Mariño. *Raisonnement classificatoire dans une représentation à objets multi-points de vue.* Thèse d'informatique, Université Joseph Fourier, Grenoble, 1993.

[Mey88] B. Meyer. *Object-Oriented Software Construction.* Prentice-Hall International Series in Computer Science, Englewood Cliffs, New Jersey, 1988.

[Mey90] B. Meyer. *Introduction to the Theory of Programming Languages.* Prentice Hall, 1990.

[MGVdZ92] B.A. Myers, D.A. Giuse, and B. Van der Zanden. Declarative Programming in a Prototype-Instance System: Object-Oriented Programming Without Writing Methods. In *Proceedings of 7th OOPSLA, Vancouver, ACM SIGPLAN Notices (27)10*, pages 184–200, 1992.

[Min75] M. Minsky. A Framework for Representing Knowledge. In P.H. Winston, editor, *The Psychology of Computer Vision*, pages 211–281. McGraw-Hill, New York, 1975.

[MNC+89] G. Masini, A. Napoli, D. Colnet, D. Leonard, and K. Tombre. *Les langages à objets.* InterEditions, 1989.

[Mos90] P.D. Mosses. Denotational semantics. In J. Van Leeuwen, editor, *Formal Models and Semantics*, Handbook of Theoretical Computer Science (Volume 1). Elsevier, Amsterdam, 1990.

[NA90] A. Nicolle-Adam. Le métalangage AIRELLE. *Revue d'intelligence artificielle*, 4(1):51–77, 1990.

[Nap92] A. Napoli. *Représentation par objets et raisonnement par classification en intelligence artificielle.* Thèse d'État, Université de Nancy, 1992.

[Nap93] A. Napoli, editor. *Proceedings of the IJCAI'93 Workshop on Object-Based Representation Systems*, Nancy, France, 1993. Rapport de Recherche CRIN 93-R-156.

[Neb85] B. Nebel. How well does a vanilla loop fit into a frame? *Data & Knowledge Engineering*, 1(2):181–194, 1985.

[Neb90] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems.* Lecture Notes in Computer Science 422. Springer-Verlag, Berlin, 1990.

[PFPS+92] R.S. Patil, R.E. Fikes, P.F. Patel-Schneider, D. McKay, T. Finin, T. Gruber, and R. Neches. The DARPA Knowledge Sharing Effort: Progress Report. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92), Cambridge, Massachusetts*, pages 777–788, 1992.

[PN93] L. Padgham and B. Nebel. Combining Classification and Nonmonotonic Inheritance Reasoning: A First Step. In J. Komorowski and Z.W. Raś, editors, *Methodologies for Intelligent Systems*, Lecture Notes in Computer Science 689, pages 132–141. Springer-Verlag, Berlin, 1993.

[PW92] J.-F. Perrot and F. Wolinski. Modélisation par objets en robotique. *Technique et Science Informatiques*, 11(1):97–115, 1992.

[Rat91] C. Rathke. Implementing frames in an object-oriented programming language. In *Proceedings of the First East European Conference on Object-Oriented Programming (EurOOP'91), Bratislava*, 1991.

[Rec92] F. Rechenmann. Method inheritance and selection in frame-oriented languages. (Article non publié), 1992.

[Rec93a] F. Rechenmann. Building and sharing large knowledge bases in molecular genetics. In *Proceedings of the 1st International Conference on Building and Sharing of Very Large-Scale Knowledge Bases (KBKS'93), Tokyo*, pages 291–301, 1993.

[Rec93b] F. Rechenmann. SHIRKA *: système de gestion de bases de connaissances centrées-objet.* INRIA Rhônes Alpes - IMAG LIFIA, Grenoble, 1993.

[Rec94] F. Rechenmann, editor. *Actes des journées Langages et Modèles à Objets, LMO'94*, Grenoble, 1994. IMAG.

[Red88] U. Reddy. Objects as Closures: Abstract Semantics of Object-Oriented Programming. In *Proceedings of the 4th ACM Symposium on Lisp and Functional Programming, Snowbird, Utah*, pages 289–297, 1988.

[RR93] C. Rathke and D.F. Redmiles. Multiple perspectives for supporting explanation in context. Technical Report CU-CS-645-93, University of Colorado, Department of Computer Science and Institute of Cognitive Science, March 1993.

[Sch89] J.G. Schmolze. Terminological Knowledge Representation Systems Supporting N-ary terms. In R.J. Brachman, H.J. Levesque, and R. Reiter, editors, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89), Toronto*, pages 421–431, San Mateo, California, 1989. Morgan Kaufmann Publishers, Inc.

[Sim69] H.A. Simon. *The Sciences of the Artificial.* The MIT Press, Cambridge, Massachusetts, 1969.

[SLT91] M.H. Scholl, C. Lasasch, and M. Tresch. Updatable Views in Object-Oriented Databases. In C. Delobel, M. Kifer, and Y. Masunaga, editors, *Deductive and Object-Oriented Databases (DOOD'91)*, Lecture Notes in Computer Science 566, pages 189–207. Springer-Verlag, Berlin, 1991.

[Smo92] G. Smolka. Feature Constraint Logics for Unification Grammars. *The Journal of Logic Programming*, 12(1–2):51–87, 1992.

[Sny91] A. Snyder. Inheritance in Object-Oriented Programming Languages. In M. Lenzerini, D. Nardi, and M. Simi, editors, *Inheritance Hierarchies in Knowledge Representation and Programming Languages*, pages 153–171. John Wiley & Sons Ltd, Chichester, West Sussex, 1991.

[Sow84] J.F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine.* Addison Wesley, Reading, Massachusetts, 1984.

[Ste90] G.L. Steele. *Common Lisp, The Language.* Digital Press, Bedford, Massachusetts, second edition, 1990.

[Tho92] R.H. Thomason. NETL and Subsequent Path-based Inheritance Theories. *Computers and Mathematics with Applications*, 23(2–5):179–204, 1992.

[US87] D. Ungar and R.B. Smith. Self: The Power of Simplicity. In *Proceedings of the 2nd OOPSLA, Orlando, Florida, ACM SIGPLAN Notices 22(12)*, pages 227–242, 1987.

[Wan89] Y. Wand. A Proposal for a Formal Model of Objects. In W. Kim and F.H. Lochovsky, editors, *Object-Oriented Concepts, Databases, and Applications*, pages 537–559. Addison Wesley, Reading, Massachusetts, 1989.

[WCH87] M.E. Winston, R. Chaffin, and D. Herrmann. A Taxonomy of Part-Whole Relations. *Cognitive Science*, 11:417–444, 1987.

[Wil94] J. Willamowski. *Modélisation de tâches pour la résolution de problèmes en coopération système-utilisateur.* Thèse d'informatique, Université Joseph Fourier, Grenoble, 1994.

[Win75] T. Winograd. Frame Representation and the Declarative/Procedural Controversy. In D.G. Bobrow and A.M. Collins, editors, *Representation and Understanding: Studies in Cognitive Science*, pages 185–210. Academic Press, New York, 1975.

[Win84] P.H. Winston. *Artificial Intelligence.* Addison-Wesley, Reading, Massachusetts, 1984.

[Woo91] W.A. Woods. Understanding Subsumption and Taxonomy: a Framework for Progress. In J. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 45–94. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1991.

[WS92] W. Woods and J. Schmolze. The KL-ONE Family. *Computers and Mathematics with Applications*, 23(2–5):133–177, 1992.