Un mécanisme d'exploitation à base de filtrage flou pour une représentation des connaissances centrée objets

Philippe Vignard

THÈSE

présentée à

L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

par

Philippe VIGNARD

pour obtenir

LE TITRE DE DOCTEUR EN INFORMATIQUE

Sujet de la thèse :

UN MÉCANISME D'EXPLOITATION A BASE DE FILTRAGE FLOU POUR UNE REPRÉSENTATION DES CONNAISSANCES CENTRÉE OBJETS

Soutenue le 12 juin 1985 devant la Commission composée de :

MM.	**L.**	**BOLLIET**	*Président*
	P.	**BERNHARD**	*Examinateurs*
	J.P.	**LAURENT**	
	H.	**PRADE**	
	F.	**RECHENMANN**	

ISBN - 2-7261-0418-5

KBD046(3)

ORSAY
N° d'ordre :

THÈSE

présentée à

L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

par

Philippe VIGNARD

pour obtenir

LE TITRE DE DOCTEUR EN INFORMATIQUE

Sujet de la thèse :

UN MÉCANISME D'EXPLOITATION A BASE DE FILTRAGE FLOU POUR UNE REPRÉSENTATION DES CONNAISSANCES CENTRÉE OBJETS

Soutenue le 12 juin 1985 devant la Commission composée de :

MM.	**L.**	**BOLLIET**	*Président*
	P.	**BERNHARD**	*Examinateurs*
	J.P.	**LAURENT**	
	H.	**PRADE**	
	F.	**RECHENMANN**	

ISBN - 2-7261-0418-5

Je tiens à remercier ici

Monsieur Louis Bolliet, Professeur à l'Université Scientifique et Médicale de Grenoble, qui a bien voulu me faire l'honneur de présider le jury de cette thèse.

Monsieur Pierre Bernhard, directeur du centre de l'Institut National de Recherche en Informatique et Automatique de Sophia Antipolis, qui m'a accueilli dans son laboratoire et son équipe, et a accepté d'être membre de mon jury.

Monsieur Jean Pierre Laurent, Professeur à l'Université de Savoie à Chambery, qui a bien voulu faire partie de mon jury.

Monsieur Henri Prade, chargé de Recherches au CNRS à l'Université Paul Sabatier de Toulouse, pour sa participation au jury, son interêt pour mes travaux et ses suggestions au cours de nos discussions.

Monsieur Francois Rechenmann, Ingénieur de Recherche à l'INRIA, qui a dirigé mon travail, m'a conseillé tout au long de mes recherches et qui est à l'origine des spécifications de la représentation des connaissances présentée sans laquelle les travaux sur le filtrage n'auraient pas pu être réalisés.

Je suis également reconnaissant à tous les membres de l'équipe du projet EDORA du centre INRIA de Sophia Antipolis, pour leur fructueuse collaboration.

UN MECANISME D'EXPLOITATION A BASE DE FILTRAGE FLOU

POUR UNE REPRESENTATION DES CONNAISSANCES CENTREE OBJETS

Philippe VIGNARD

MOTS-CLES - représentation des connaissances centrée objets
- expression de la dynamique sous forme déclarative
- exploitation d'une base d'objets
- filtrage flou
- distribution de possibilités
- distance entre objets
- sémantique des objets
- analogie, typologie

RESUME

Une représentation des connaissances centrée objets, déclarative et uniforme, est présentée. Elle permet de construire une base d'objets dynamique.

Le mécanisme d'exploitation associé est fondé sur un processus élémentaire de filtrage flou. De façon général, il permet l'exploitaton d'une base d'objets dans laquelle les traitements sont aussi spécifiés de façon déclarative. Il permet aussi la manipulation de termes du langage naturel définis à l'aide d'outils mathématiques extraits de la théorie des ensembles flous.

Le processus manipule la sémantique des objets à l'aide d'informations typées. Il calcule des distances entre objets variant entre 0 et 1 au lieu de rendre de simples réponses binaires. Deux stratégies de filtrage permettent d'utiliser des raisonnements nuancés et de diverses natures.

Ces outils sont manipulés pour élaborer un système intelligent d'aide à la modélisation mathématique en biologie.

PLAN GENERAL

INTRODUCTION

Les représentations des connaissances ainsi que leurs manipulations sont l'un des problèmes cruciaux en Intelligence Artificielle, et tout particulièrement pour le traitement du langage naturel et les systèmes experts.

Il y a plusieurs familles de représentations. Des systèmes utilisent une représentation à base d'objets par opposition à d'autres qui utilisent une représentation relationnelle.

Les systèmes à règles de production utilisent un formalisme relationnel, en plaçant l'importance non pas sur les objets mais sur les relations qui les unissent. Inversement, les systèmes avec "frames" tentent de décrire la réalité à l'aide d'une approche centrée sur les objets et les propriétés qui les caractérisent.

Développement des représentations centrées objets

Les systèmes informatiques actuels, dans tous les domaines, offrent de plus en plus d'outils spécialisés et sont de plus en plus faciles d'utilisation. Ils deviennent ainsi accessibles à des utilisateurs qui ne possèdent pas toujours la connaissance et l'expertise requise. Mais aucun de ces systèmes n'est capable de dire dans quelles circonstances il est judicieux d'utiliser un outil plutôt qu'un autre. Ils sont donc souvent mal ou sous employés.
La connaissance du "comment" est le fait du système, mais celle du "quand" et du "pourquoi" appartiennent encore à l'utilisateur (Vivet 81).

Pour corriger cela, des systèmes intelligents sont à l'étude depuis peu. L'étude de la représentation des connaissances dans de tels systèmes est un problème fondamental. Pour être réellement utiles et pour que les utilisateurs aient confiance, ces systèmes doivent pouvoir expliquer leurs conseils et raisonnements. Ils doivent donc connaître leur propre connaissance et pouvoir facilement la manipuler. Dans ce but, elle doit être mémorisée explicitement, de façon déclarative et modulaire.

Pour ces raisons, parmi d'autres, la représentation des connaissances centrée objets parait la plus favorable dans beaucoup de domaines. Elle constitue un moyen simple et unitaire de représenter de façon explicite et déclarative la connaissance liée aux outils disponibles, aux objets manipulés et aux situations ou contextes d'utilisation.

Les problèmes

Les représentations en objets structurés dérivent essentiellement des travaux de Minsky (Minsky 75) sur les "frames".

Le premier problème est d'améliorer les possibilités descriptives de cette représentation en développant l'aspect déclaratif et en permettant l'utilisation de termes du langage naturel.

Le second problème, essentiel, concerne l'utilisation de cette représentation. Le mécanisme d'exploitation classique ne permet pas une utilisation satisfaisante des objets. Il permet d'extraire de la base l'ensemble des entités correspondant à un objet pris comme modèle. Mais le mécanisme de pattern-matching pour l'association entre objets est trop restrictif, puisque fondé uniquement sur une correspondance exacte et syntaxique entre les noms des composants de chaque objet.

Thème de la thèse

En premier lieu, nous présentons une représentation centrée objets déclarative et homogène.

Nous étudions l'utilisation de termes du langage naturel codés à l'aide d'outils mathématiques extraits de la théorie des ensembles flous.

Notre travail concerne principalement l'étude et la spécification d'un mécanisme d'exploitation à base de filtrage flou. Ce processus permet outre l'exploitation d'une base d'objets classique, la manipulation des termes du langage naturel et l'utilisation de la sémantique des objets. Ce dernier point évite une mise en correspondance trop restrictive. Des notions d'imprécis, d'incertain et d'importance peuvent être manipulées. Le processus est souple et rend une évaluation de distance entre les deux objets mis en correspondance au lieu d'une réponse binaire. Les raisonnements constructibles sont de diverses natures.

Plan de la thèse

Nous décrivons dans le premier chapitre les principales représentations des connaissances utilisées en Intelligence Artificielle, avec leurs mécanismes d'exploitation.

Dans le deuxième chapitre, les notions d'objets et de représentations centrées objets sont détaillées. Les langages orientés objets sont opposés aux représentations centrées objets.

Dans le troisième chapitre, nous présentons une représentation des connaissances centrée objets. Un mécanisme de multi-héritage est proposé permettant de conserver l'avantage de description avec des termes du langage naturel dans les connaissances héritées.

Le quatrième chapitre apporte la spécification du mécanisme d'exploitation à base de filtrage flou. Nous présentons les outils utilisés de la théorie des ensembles flous. Les techniques de codage et de manipulations des valeurs floues associées aux termes du langage naturel sont détaillées, ainsi que le processus de filtrage.

Enfin, le cinquième chapitre présente une approche système expert et l'utilisation de la représentation centrée objets dans le domaine de l'aide à la modélisation mathématique en biologie.

CHAPITRE 1

REPRESENTATION ET UTILISATION DES CONNAISSANCES

Chapitre I/ Représentation et utilisation des connaissances

PLAN

I/ Introduction

II/ Représentation procédurale

- 1/ Deux types de représentations
  - a/ Les automates finis
  - b/ Les procédures
- 2/ Des applications
- 3/ Avantages et inconvénients

III/ Représentation avec la logique des prédicats

- 1/ Calcul propositionnel
- 2/ Calcul des prédicats
- 3/ Raisonnement : le démonstrateur de théorèmes
- 4/ Applications
- 5/ Avantages et inconvénients

IV/ Réseaux sémantiques

- 1/ Présentation
- 2/ Historique
- 3/ Types de liens
- 4/ Le raisonnement
- 5/ Applications
- 6/ Avantages et inconvénients

V/ Régles de Production

1/ Présentation
2/ Domaines appropriés
3/ Raisonnement avec les règles de production: les systèmes de production
   a/ Détection
   b/ Choix
   c/ Déduction
4/ Applications
5/ Avantages et inconvénients

VI/ Propriétés attendues d'une représentation des connaissances

I/ INTRODUCTION

La nature des connaissances et leur représentation a de tout temps intéressé diverses catégories de spécialistes, tels que les philosophes, psychologues, linguistes et sociologues. En Intelligence Artificielle aussi ce problème est crucial en particulier en vue d'élaborer des programmes qui puissent se comporter de façon intelligente (Kayser 84, Woods 83).

Schématiquement un problème abordé en Intelligence Artificielle comporte trois étapes étroitement liées : conceptualisation, recherche d'une représentation adéquate et emploi d'une méthode de résolution adaptée (Vignard 84c/).

La première étape est essentielle. Il s'agit alors de savoir quels sont les concepts ou catégories conceptuelles nécessaires à l'établissement d'une base de connaissances. Cette phase de transfert d'expertise est encore sans méthodologie et souvent laissée à la charge des utilisateurs (Mangin 84).

Lors de l'étude d'une représentation des connaissances selon le domaine, un certain malentendu peut provenir de la nuance à faire entre "informations" et "connaissances". Si on ne fait pas cette distinction, le problème de l'étude de la représentation des connaissances n'a plus de sens. Mais le livre contenant des informations ne connait pas ce qu'il renferme. Connaitre est une opération active qui implique des capacités de mémorisation mais surtout une faculté d'infèrences. La vraie connaissance suppose l'utilisation à bon escient des informations disponibles.

Elaborer une représentation des connaissances consiste donc à trouver des structures informatiques qui permettent le stockage et l'utilisation correcte par la machine des informations.

Le mécanisme d'exploitation est choisi en fonction de la représentation utilisée. Un essai de typologie peut aider à faire ce choix (Laurent 84).

Ces trois étapes sont étroitement liées. Mais nous ne considérons dans ce chapitre que l'étude des principales représentations utilisées en informatique, problème central en particulier lors de la réalisation d'un système expert, mais aussi pour la compréhension du langage naturel, en robotique et pour des bases de données intelligentes (Nicolas 81).

Différentes catégories de représentations ont déjà été étudiées (Lauriére 82 a/,

Mc Calla 83, Pinson 81, Mylopoulos 80). Au cours des années 70, une polémique opposa les tenants d'une représentation procédurale à ceux qui optaient pour une vision déclarative. L'idéal recherché dans ce dernier cas est de spécifier un savoir indépendamment des contraintes et des méthodes d'utilisation. On cherche à répondre à une question de type "quoi" alors qu'une procédure exprime par essence un flot d'informations et traduit "comment" transite la connaissance.
Puis sont apparues des représentations mixtes, telles que les "frames" (Minsky 75), utilisant des procédures et des déclarations. La polémiques oppose alors le formalisme relationnel au formalisme objet. Ce problème abordé au chapitre II nous aménera à l'étude des représentations centrées objets, sujet de cet ouvrage.

Selon l'opposition procédural/déclaratif, les principales représentations des connaissances en Intelligence Artificielle sont purement procédurales, la logique des prédicats, les réseaux sémantiques ou les règles de production.

II/ Représentation procédurale

De même que l'on définit la connaissance comme un mélange d'informations et de modes d'utilisation de celles-ci, Kowalski définit un algorithme comme comportant une partie logique qui correspond à la signification de l'algorithme et une partie contrôle. En représentation procédurale, comme dans un programme classique, ces deux parties sont confondues (Kayser 84).
Avec une telle représentation, la base de connaissances est une collection de procédures. Les procéduralistes affirment que toute la connaissance utile et nécessaire dans un domaine est donnée si l'on sait "comment" se servir des connaissances du domaine. Les procédures impliquent une connaissance très précise a priori sur le problème à résoudre, car tout est codé explicitement.
On distingue deux types de représentations procédurales : les automates finis et les procédures.

1/ Deux types de représentation

a/ Les automates finis

Un automate fini (Pinson 81) est utilisable pour représenter des plans d'actions : les états représentent les actions à entreprendre, les arcs représentent les règles de décisions pour passer d'un état à un autre. Un automate fini est déterministe ou non. Dans ce dernier cas le mécanisme de contrôle est plus complexe car il doit examiner tous les chemins et choisir celui qui aboutit à une solution.

b/ Les procédures

Des programmes classiques sont utilisés et seuls les types d'appel des procédures varient.
Des connaissances purement procédurales sont représentées. Il y a quatre méthodes principales pour appeler une procédure :
-appel direct: l'utilisateur sait quelle procédure doit être utilisée et à quel moment.

-attachement procédural: la procédure est associée à une zone de données de la source de connaissances. A chaque accés à cette zone la procédure est exécutée.

-démon: il est introduit par l'instruction démon(P,C) interprétée comme ceci:
"si condition C est rencontrée dans l'exécution du système alors appel et exécution de la procédure P"
L'utilisation de démons permet de classer des situations exceptionnelles dans la procédure P, rendant ainsi le programme principal plus lisible et facile à organiser. Cependant l'utilisation des démons est coûteuse car le moniteur doit vérifier sans arrêt si un démon ne doit pas être introduit.

-appel dirigé par schéma (ou pattern-matching)
Le nom de chaque programme décrit la tâche qu'il réalise. Ce schéma est utilisé pour appeler le programme. Ainsi le programme principal spécifie le but à réaliser et le moniteur recherche les programmes dont le schéma (pattern)

correspond à ce but.
Par exemple le schéma correspondant au but "Situe-a" est: (situe-a personne objet) qui indique que le programme "Situe-a" planifie la séquence d'actions nécessaires pour transporter une personne près d'un objet. Cette technique permet une décomposition naturelle d'un problème pour atteindre le but recherché, mais donne un système très difficile à modifier et à étendre.

2/ Des applications

PLANNER créé en 1972 par Hewitt utilise la notion d'appel dirigé par schéma. La base de connaissances est un ensemble d'assertions et de théorèmes ou démons. Chaque théorème a un schéma associé.

De même dans les systèmes de production la base de connaissances est un ensemble de règles de production. Ces dernières comme les théorèmes dans PLANNER consistent en un modèle (conditions) qui appelle une ou plusieurs actions.

Tous les attachements procéduraux offerts dans des représentations déclaratives ou mixtes, telle que les frames, sont aussi un exemple de représentation procédurale. C'est le cas des attachements procéduraux classiques comme "if-needed,if-added, if-removed" utilisés dans FRL (Goldstein 80).

Les langages à base d'acteurs comme PLASMA (Granger 82) peuvent aussi être cités. Dans ce cas tous les objets de la base de connaissances sont des acteurs, c'est à dire des agents actifs. Les acteurs sont capables d'envoyer et de recevoir des messages qui sont eux même des acteurs. Ecrire un programme avec un tel langage oblige à décider quels vont être les acteurs, les messages recus par chacun et ce qui doit être fait à la réception des messages.

3/ Avantages et inconvénients

Le raisonnement des systèmes procéduraux est parfaitement guidé. Ils ne peuvent pas utiliser des connaissances inadéquates ou suivre un cheminement incorrect. Il est facile de représenter les heuristiques qui sont le plus souvent

de nature procédurale. Le raisonnement bien dirigé évite une recherche, éventuellement longue et coûteuse, entre les actions possibles. Les procédures et attachements procéduraux permettent de coder des effets de bord.

Mais en contre partie, cette représentation ne permet pas de tout modéliser. Elle n'est pas modulaire et ne facilite pas l'extension ou la modification du système.
Elle est peu explicative. Le système n'a pas de connaissances sur son mode de raisonnement et sur ses connaissances ce qui l'empêche d'expliquer ce qu'il fait et pourquoi il le fait.

III/ Représentation avec la logique des prédicats

Cette représentation est identique à celle qui est obtenue par les psychologues avec l'analyse prédicative (Le Ny 79). Son importance dans la représentation des connaissances chez l'homme n'est plus à prouver (Israel 83).

1/ Calcul propositionnel

La logique fut la première représentation en Intelligence Artificielle. Deux problèmes sont à étudier: la représentation de ce qui peut être dit, ce sont les axiomes, et les méthodes de déduction, ce sont les règles d'inférences (Barr 81, Nilsson 80).
En logique, seule la valeur de vérité d'une proposition importe. Il y a cinq connectives qui sont: et($\wedge$), ou($\vee$), non($\neg$), implication($\Rightarrow$) et l'équivalence ($\Leftrightarrow$). Ces connectives, avec des propositions, constituent le calcul propositionnel qui permet d'exprimer des faits tels que: "le livre est sur la table ". Des règles de calcul des valeurs de vérité à l'aide des tables de vérité permettent la composition de propositions. La première règle d'inférence dans ce calcul est le modus ponens:

(X et (X-->Y) --> Y).

avec son dual, le modus tollens :

(non Y et (X --> Y) --> non X)

Mais dans le calcul propositionnel, les règles s'appliquent directement aux faits. Les prémisses et les conséquences des règles ne peuvent être explicitement que des propositions.

2/ Calcul des prédicats

Pour pouvoir aussi parler d'objets, de relations entre ces objets et pour généraliser ces relations entre classes, il faut utiliser le calcul des prédicats, extension du calcul propositionnel (Barr 81). On dispose en plus de variables et de quantificateurs. Au lieu de ne se préoccuper que de la valeur de vérité des expressions, on cherche aussi à représenter des états pour des objets spécifiques. Les prédicats sont des descriptions d'individus. Un prédicat a une valeur vraie ou fausse. Par exemple, le prédicat Homme(Socrate) a la valeur vraie, mais Infèrieur(5,3) est faux. Des quantificateurs, existentiel ($\exists$) et universel ($\forall$) sont utilisés.

L'expression "tout homme est mortel " se traduit avec les 2 prédicats Homme(X) et Mortel(X) et la variable X:

$$\forall X, \text{Homme}(X) \dashrightarrow \text{Mortel}(X)$$

La manipulation de variables complique le processus de filtrage ou pattern-matching. Ce mécanisme, trés utilisé en Intelligence Artificielle, permet la mise en correspondance de deux formes, le filtre et le fait. Par exemple avec le filtre (A ?X B) et le fait (A C B), où ?X désigne une variable, l'unification des deux formes est possible avec la substitution (?X/C). Le processus de filtrage doit donc fournir, si c'est possible, les substitutions permettant l'unification des deux formes (Granger 84b/, Barr 81). Ce processus est à la base des mécanismes d'exploitation des différentes représentations que nous présentons.

Dans le calcul des prédicats du premier ordre seules les variables peuvent être quantifiées. Des fonctions sont utilisables, qui à la différence des prédicats, rendent un objet et non pas "vrai" ou "faux". De facon formelle, on définit 4 constructions possibles pour le langage des prédicats du premier ordre:

-le terme qui est une variable, une constante (fonction à 0 argument) ou une fonction à n aguments: F(t1,t2,...,tn) où t1,...,tn sont des termes;

-l'atome qui est une proposition (prédicat à 0 argument) ou un prédicat à "n" arguments P(t1,t2,...,tn) où t1,...,tn sont des termes;

-la formule bien faite (fbf) qui est un atome (A ou B...), ou une composition d'atomes (AetB, AouB, nonA, A—>B, (∀X)A, (∃X)A);

-la phrase qui est une fbf où toutes les variables sont liées.

Une règle simple d'inférence est :

si on a∀X F(X) alors on a F(A)

Mais, même sur un exemple simple, la traduction en logique du premier ordre n'est pas toujours triviale (Le Ny 79, Gondran 83).

3/ Le raisonnement : le démonstrateur de théorèmes

La manipulation des prédicats peut se faire essentiellement de deux manières (Nilsson 80, Chouraki 79).

La méthode de déduction par réfutation est utilisée (Nilsson 80). avec des formes clausales. Cette méthode est basée sur le raisonnement par l'absurde. Une clause est une disjonction de littéraux. Un littéral est un atome ou la négation d'un atome. On note +Li l'affirmation du littéral Li et -Li sa négation. Une résolution avec les clauses

C1=+L1-L2-L3..-Li.. -Ln et C2=+Li -L'2...-L'm

permet de construire la résolvante

C1+C2=+L1-L2...-Ln-L'2...-L'm.

Toute phrase peut être mise sous forme de clauses. Pour prouver une proposition, on utilise cette méthode avec les clauses hypothèses et la forme clausale de la négation du but. Le raisonnement est un succés si la résolution fournit la clause vide.

Par exemple avec les hypothèses :

S1 (∀X) (sait-lire (X)⊃ lettré(X))

S2 $(\forall X)$ (dauphin $(X) \supset \neg$lettré(X))

S3 $(\forall X)$ (dauphin$(X) \wedge$intelligent(X))

on cherche à vérifier C: $(\exists X)$ (intelligent$(X) \wedge \neg$sait-lire(X)).
Les clauses correspondant aux hypothèses sont :

C1= $\neg$ sait-lire$(X) \vee$ lettré(X)

C2= $\neg$dauphin$(X) \vee \neg$ lettré(X)

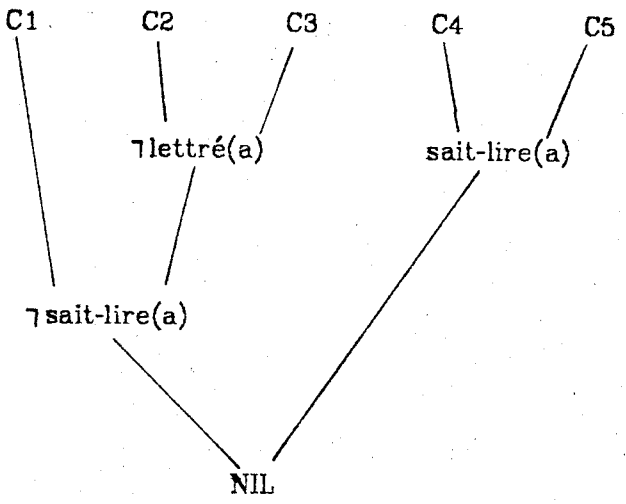
C3= dauphin(a)

C4= intelligent(a)

C5 est la forme clausale de la négation du but :

C5=$\neg$ intelligent$(X) \vee$sait-lire(X)

Le graphe de réfutation suivant montre que C est bien vérifié.



En notant C2=-dauphin(X)-lettré(X) et C3=+dauphin(a) on voit que la résolvante C2+C3, avec la substitution (X/a) est bien -lettré(a) ou lettré(a). Les autres résolutions sont construites de la même facon, jusqu'à obtenir NIL, c'est à dire la clause vide.

Le second principe de raisonnement est le mécanisme classique d'inférence décrit dans un paragraphe suivant sur les systèmes de production. Pour donner une idée sommaire de ce fonctionnement, nous ne citons ici qu'un exemple :
soient les faits initiaux : H, K
et la base de règles :

R1 : A --> E
R2 : H --> A
R3 : EetK --> B
R4 : B --> D
R5 : DetEetK --> C

Le démonstrateur de théorèmes peut déduire des faits nouveaux à partir de la base de connaissances, avec le principe de modus ponens. Par exemple, on déduit :
R2 H --> A donc on a les faits H, K, A
R1 A --> E donc on a les faits H, K, A, E
R3 EetK --> B donc on a les faits H, K, A, E, B

Outre le problème du choix de la règle à utiliser, se pose le problème de l'instanciation des variables d'une règle.

4/ Des applications

QA3 (Barr 81) créé par Green en 1969 est un système questions-réponses qui utilise cette représentation et permet de résoudre des problèmes simples en chimie, en robotique, et en programmation automatique. Les déductions se font selon la méthode de résolution. Par exemple si l'utilisateur donne l'expression : " FeS est un sulfure et un composant gris", QA3 génère la formule logique :

Sulfure(FeS) et Composant(FeS) et Gris(FeS)

QA3 peut alors répondre à une question du genre "est-il exact qu'il n'existe pas de chose grise qui soit un sulfure ?", ce qui se traduit en :

non$\exists$ X/ Gris(X) et Sulfure(X)

Mais la méthode de résolution est trop peu efficace dès que le nombre de faits

dans la base de connaissances devient important.

Le langage PROLOG (Colmerauer 83), créé plus récemment, utilise le calcul des prédicats. Les prédications servent à construire les faits et les règles. Ainsi "Aime (Anne,Roger)" est une assertion en PROLOG, "Aime (x,y) si Aime (y,x)" est une règle comme "Aime (x,y) si Mère (x,y)". Faits et règles utilisent la même description. Les définitions récursives sont permises et les inférences sont automatiques (Dahl 83).

SNARK (Gondran 83) est un système élaboré par Lauriére en 1982 qui utilise cette représentation. Dans SNARK tous les faits sont représentés avec des relations binaires par l'intermédiaire d'un "zombi", c'est à dire une entité n'ayant pas forcément d'existence propre. Par exemple, au fait "Jean donne un livre à Paul", on associe le zombi $a et les trois relations binaires: Don($a)=livre, Donneur($a)=Jean, Receveur($a)=Paul. Un fait est représenté par un triplet: zombi,propriété, valeur (coefficient) où le coefficient (entre 0 et 1) indique la vraisemblance du fait.

Enfin, la logique intervient aussi pour permettre l'insertion d'une certaine intelligence dans les bases de données traditionnelles qui deviennent des "bases de données déductives" (CERT 81).

5/ Avantages et inconvénients

La logique semble être une voie naturelle pour exprimer certains faits (Le Ny 79). Elle dispose de solides bases théoriques. Des logiques non classiques ont été développées, agrandissant le domaine d'utilisation possible (logiques multivalentes, logique non-monotone).
La notation formelle permise est claire, facile à comprendre, flexible, précise et modulaire. On peut exprimer un fait sans se soucier de ses manipulations. Comme le fait remarquer Lauriére (Lauriére 82a/), la véritable modularité est la séparation totale des connaissances et de leurs utilisations, ce qui est atteint avec cette représentation. On peut rentrer en vrac des assertions à l'aide de prédicats (Mylopoulos 80). Cette modularité permet une économie de représentation et les inférences sont naturelles. Beaucoup de représentations peuvent être traduites en logique du premier ordre.

Mais il n'est plus possible de représenter des connaissances procédurales ou des

heuristiques. Une organisation entre les éléments qui constituent la base de connaissances fait défaut. Les assertions sont placées en vrac et si la base devient trop importante, il n'est plus possible de l'utiliser de facon performante. De plus le calcul des prédicats est une représentation très formelle et mathématique et ne permet pas de manipuler facilement des informations incertaines ou incomplètes (Prade 82, 83).

IV/ Réseaux sémantiques

1/ Historique

Pour organiser la connaissance, on utilise souvent en psychologie des réseaux associatifs (Le Ny 79) assimilables à ce que l'on nomme en informatique des réseaux sémantiques (Barr 81). Ces objets connurent une première vogue en informatique dès 1960 grâce à la souplesse de leur sémantique propre. Quilliam en 1968 fut le premier à se servir d'un réseau, ensemble de noeuds liés entre eux. De même, Raphaels en 1968 avec le système SIR, utilise cette représentation. SIR est un système d'interrogations avec une variété de raisonnements très simples :

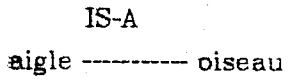
le doigt est une partie de la main
la main est une partie du corps
donc le doigt est une partie du corps

Dans les années 70, Simon utilise un réseau sémantique dans ses recherches pour la compréhension du langage naturel.
Des travaux plus récents rapprochent les réseaux sémantiques des "frames" que nous verrons par la suite.

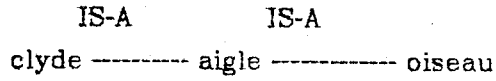
2/ Présentation

Sur un plan formel un réseau sémantique est essentiellement un graphe étiquetté, dont les noeuds représentent des entités, des individus et dont les

arcs sont des instances de relations binaires (Fieschi 84). Il est utilisé pour représenter des connaissances et leurs interdépendances. Des modèles de réseaux différents ont été élaborés sur le principe de ce formalisme (Barr 81, Hendrix 75). Si l'on veut représenter simplement le fait que "les aigles sont des oiseaux", on crée deux noeuds liés avec le lien "IS-A" comme suit:



Si "clyde" est un individu particulier qui est un aigle, on peut écrire:



Dans un domaine où le raisonnement est basé sur une organisation complexe, le réseau sémantique est une représentation naturelle. Un formalisme très utilisé est celui des réseaux sémantiques partitionnés de Hendrix (Hendrix 75). C'est une extension des réseaux sémantiques, où les sommets et les arcs sont regroupés en espaces, eux même regroupés en "vistas". Hendrix a introduit cela pour représenter des situations exigeant la délimitation de sous ensembles d'informations, notamment la portée de quantificateurs.

3/ Types de liens

Dans un réseau sémantique, les noeuds sont le plus souvent de simples prédicats. On distingue parmi eux, génériques et individus ou concepts et individus. Les liens permettent une hiérarchisation des entités selon une parenté de généralisation/spécialisation. Les héritages de propriétés sont alors possibles. Des informations sont héritées pour une entité en provenance des noeuds hiérarchiquement supèrieurs.

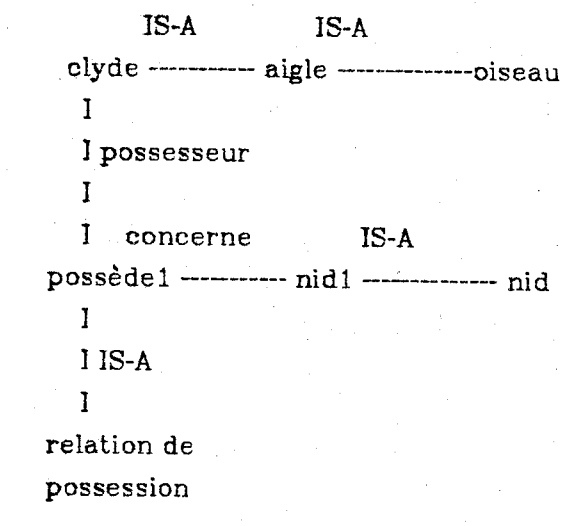
Les liens utilisés entre ces noeuds sont essentiellement de deux types: IS-A et AKO. Ce sont des constructeurs de base. Une étude plus détaillée est donnée dans (Brachman 83).

Les types de liens AKO (A Kind Of) et ISA (Is A) se distinguent selon qu'ils sont entre génériques ou entre génériques et individus. Les relations entre deux génériques peuvent exprimer des liens entre sous-ensembles et sur-ensembles ou entre généralisations et spécialisations (ex: personne - voyageur). Un lien AKO peut aussi servir à une classification (un chien est une sorte de mammifère), à une inclusion conceptuelle ou une restriction de valeurs.

Les relations entre un générique et un individu sont souvent appelées "instanciations", et représentent la relation "élèment de". On exprime ainsi un fait tel que "clyde est un aigle", ou "clyde" est vu comme un individu , c'est à dire un noeud terminal de la hiérarchie, sans plus aucune spécialisation possible.

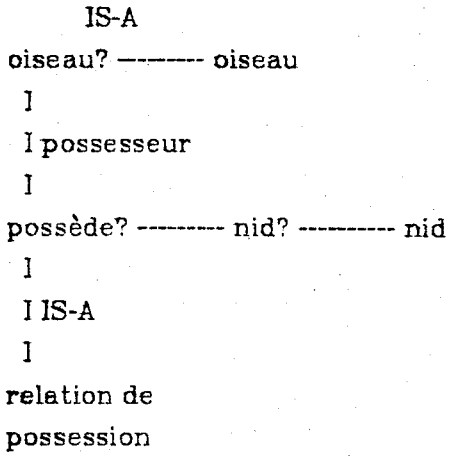
4/ Le raisonnement

Il n'y a pas de sémantique formelle dans les réseaux sémantiques. Une signification est associée à une représentation seulement par la nature des procédures qui la manipulent. Il existe donc différentes procédures pour permettre des inférences.
Le mécanisme de raisonnement le plus utilisé dans les réseaux sémantiques est le pattern-matching (Waterman 78a/) cité au paragraphe précédent. Supposons par exemple que l'on ait le réseau sémantique suivant:



Donc "Clyde" est un aigle qui posséde un nid, "nid1". Pour représenter la

question "Y a t'il un oiseau qui possède un nid ?" le système construit le filtre suivant :



Il essaie alors d'identifier dans la base de connaissances une valeur pour chacune des trois inconnues oiseau?, possède? et nid?, de telle manière que le filtre ainsi instancié représente une situation vraie dans la base. Dans ce cas la réponse sera oui, "Clyde" est un oiseau qui possède un nid.

5/ Des applications

PROSPECTOR est un système expert dans le domaine de la prospection minière. Il utilise deux types de représentation : des règles de production et des réseaux sémantiques partitionnés.

Cette représentation est aussi utilisée dans des langages tels que OWL (Szolovits 77) pour la représentation des connaissances.

PSN (Procédural Semantic Network) est aussi un langage étudié à partir de 1976, utilisant les réseaux sémantiques, pour la représentation des connaissances.

Des systèmes en médecine utilisent aussi cette représentation, comme CASNET ou ABEL (Fieschi 84).

6/ Avantages et inconvénients

Les réseaux sémantiques sont faciles à lire, à comprendre et permettent des représentation graphiques. La base de connaissances est organisée.
Le pattern-matching pour exploiter cette représentation est facile à utiliser.

Mais il manque une sémantique formelle et une terminologie standard aux réseaux sémantiques. Certains types de connaissances (procédurales ou quantifiées) ne peuvent pas être exprimés (Pinson 81). Bien souvent un noeud simple ne suffit pas pour représenter la connaissance et à l'heure actuelle des études sont faites sur des réseaux entre frames (Barr 81). De plus il n'existe pas de procédures qui manipulent rapidement des réseaux sémantiques importants. Les systèmes qui utilisent les réseaux sémantiques ont des difficultés pour fournir des explications sur leur raisonnement.

V/ Les règles de production

1/ Présentation

Les règles sont utilisées dans des systèmes de production, proposés en premier lieu par Post en 1943. Une règle de production est la spécification d'une action conditionnelle de la forme:

si < prédicat > alors <action >
antécédent conséquence

L'ensemble des règles de production est organisé sous forme d'un réseau c'est à dire que la conséquence d'une règle peut être l'antécédent d'une autre.
Les règles de production permettent de représenter la connaissance sous forme de petits modules indépendants. Elles formalisent bien le raisonnement humain, fondé lui aussi sur des règles de type antécédent-conséquent.
Ainsi le système expert MYCIN pour le traitement des infections bactériennes du sang (Lauriére 82b/) contient plus de 200 règles de production. Un exemple en est le suivant:

Régle 85 : si 1/ le site de la culture est le sang
et si 2/ l'organisme est à gram négatif
et si 3/ le patient est un bôte à risque
alors il est possible (0.6) que l'organisme soit le
pseudomonia aeruginosa

Pour permettre la manipulation d'informations incertaines, un coefficient de plausibilité entre 0 et 1 est associé à chaque règle. Cette grandeur mesure la confiance que l'on a dans cet évènement.

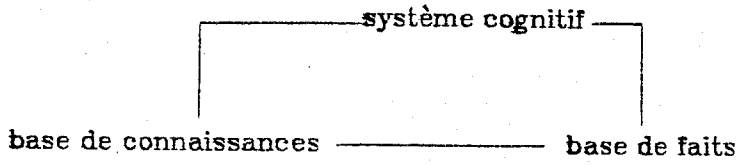
2/ Domaines d'application

Les règles de production sont très utilisées pour les systèmes experts. Mais cette représentation ne convient qu'à certains domaines de connaissances caractérisés par Davis et King en 1979 (Barr 81). Il faut que la connaissance soit diffuse et constituée d'un grand nombre d'éléments. Une situation doit être décrite par des paramètres ayant une faible interaction entre eux. Les processus doivent pouvoir être représentés par un ensemble d'actions indépendantes. Par définition les règles de production ne peuvent pas se référencer entre elles. Le domaine doit être justifiable du schéma d'inférence simple du type modus ponens. Les tâches à résoudre ne doivent pas faire appel à de nombreux sous-buts fortement reliés entre eux. Enfin, les connaissances doivent pouvoir être séparées de la facon dont il faut les utiliser. Cette propriété de modularité peut aussi être une contrainte.

3/ Le raisonnement à l'aide de règles de production :

les systèmes de production

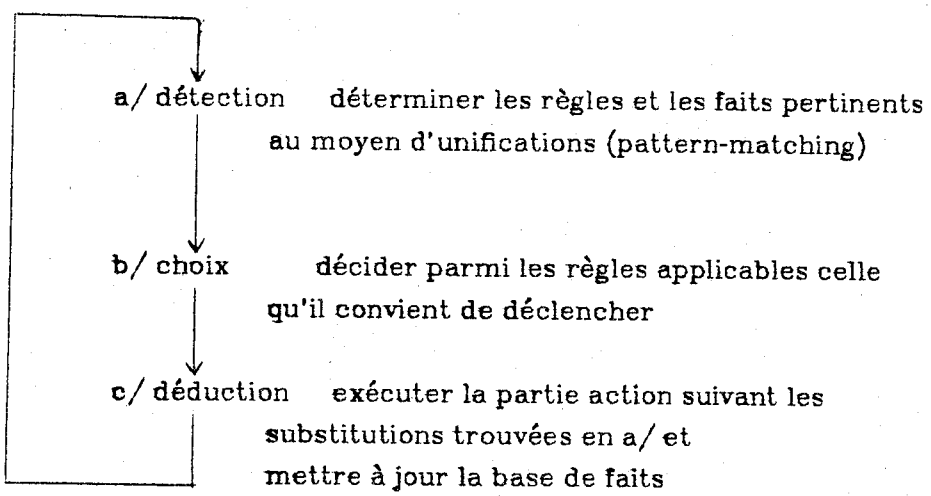
Les systèmes de production manipulent les règles de production selon le mécanisme de modus ponens. Ils servent actuellement aux psychologues pour modéliser certains processus intellectuels (Le Ny 79).
Un système de production comprend une base de faits (ou espace de travail), une base de connaissances et un système cognitif (interpréteur).



La base de connaissances contient toutes les connaissances du domaine sous forme de règles de production nécessaires pour que le système cognitif agisse comme un expert humain.

L'espace de travail contient l'ensemble des faits que le programme a pu déduire à un instant donné. Seul y figure initialement l'énoncé du problème à résoudre.

Le système cognitif, ou plus généralement mécanisme d'exploitation, est la partie clé du système qui contrôle l'ordre des déductions successives. Le travail se fait selon une séquence de cycles élémentaires (Lauriére 82b/):



a/ Détection

En phase de détection le système construit l'ensemble de conflit, constitué des règles activables, en évaluant chaque règle par référence au contenu de la base de faits. Une règle peut contenir des variables. Pour qu'elle soit activable, il faut tenter de les instancier pour unifier les prémisses avec la base de faits. Si celle-ci est importante ainsi que le nombre de variables dans les règles, ce processus, fondé aussi sur le filtrage, devient long et coûteux (Forgy 84,82).

b/ Choix

Dans la phase de choix de l'interpréteur, la stratégie pour la résolution des conflits est importante. Il s'agit alors de choisir la règle à activer parmi les règles activables. Trois grandes familles de structures de contrôle sont proposées dans les systèmes experts : la recherche exhaustive, le choix par évaluation et le contrôle par méta-règles (Barr 81, Lauriére 82b/).

Dans le cas d'une utilisation exhaustive des règles, le contrôle est particuliérement élémentaire, puisqu'il ne tient pas compte de l'évolution de la base de faits. Toutes les règles activables sont utilisées.

Dans le cas d'un contrôle par évaluation, le critère de choix est variable. L'évaluation la plus sommaire est de s'arrêter à la première règle qui convient. On peut aussi sélectionner la règle de coefficient de plausibilité maximum, celle qui utilise les faits les plus récents ou les plus importants.

Dans le dernier cas, des méta-règles permettent au système de raisonner sur le contrôle. Les méta-règles opérent sur les règles objets, c'est à dire les règles de production classiques, et concluent sur leur utilisation selon la situation de travail. C'est le fait qu'elles expriment une "connaissance sur la connaissance" qui en fait des méta-règles.

c/ Déduction

Les règles de production sont utilisées de diverses maniéres. Si une exploitation en mode antécédent (ou cheminement avant) est utilisée alors une règle est applicable lorsque sa partie condition est satisfaite dans la base des faits. Le processus s'arrête quand on a atteint un but cherché ou si plus aucune action ne peut être déclenchée.

Lors d'une exploitation en mode conséquent (cheminement arrière) une règle est applicable lorsque sa partie droite représente un sous-but à atteindre (Descottes 81). L'appliquer consiste à demander de valider sa partie gauche au besoin en engendrant de nouveaux sous-buts. Lorsque le système n'a pas suffisament d'informations pour déclencher une règle et qu'aucun but ne peut être atteint, il interroge l'utilisateur et lui demande de valider un fait si possible.

Certains systèmes proposent un cheminement mixte (Cordier 84a/, Vignard

84a/) qui correspond à un cheminement avant avec des buts intermèdiaires spécifiés par l'utilisateur. Ces indications permettent à chaque cycle de focaliser le raisonnement vers la résolution des buts donnés.

Deux raisons font qu'un système expert doit pouvoir raisonner en chaînage avant et arrière (Gondran 83).
La première est simple: si on ne connaît pas le but de facon détaillée le chaînage avant est naturel. Si au contraire on a un ou plusieurs buts à vérifier, le chaînage arrière permet de restreindre la combinatoire.
La seconde raison est plus complexe: nos connaissances, en dehors des mathématiques et de la logique, sont des hypothèses. Nous utilisons un raisonnement démonstratif pour assurer la validité de nos connaissances mathématiques et des raisonnements plausibles pour le reste. Donc dans certains problèmes, nous devons gérer deux types de connaissances: des connaissances mécanistes, certaines, et des connaissances empiriques, incertaines. Ces dernières sont utilisées en chaînage avant et permettent d'évoquer des buts possibles, alors que les premières sont utilisées en chaînage arrière pour confirmer ou infirmer les buts évoqués.
De plus le chaînage arrière est important dans tous les cas où le système n'a qu'une information incomplète sur le domaine. Il permet de préciser les informations à acquérir.
L'interpréteur doit aussi savoir expliquer son raisonnement,ce qui est une caractéristique essentielle des systèmes experts.

4/ Applications

Les systèmes de production grâce à leur modularité et à leur facilité de manipulation et de compréhension ont été très utilisés pour élaborer des systèmes experts dans différents domaines (Barr 81, Bonnet 82, Cholvy 83, Descottes 81, Dincbas 80a/, Gevarter 82, Laurière 82b/, Lagache 84, Mc Dermott 80). Une étude des systèmes de production est faite par Newell dans (Newell 73). Nous pouvons en citer quelques uns, par exemple DENDRAL qui donne la formule développée d'un corps organique à partir de sa formule brute et de son spectogramme de masse.
Les systèmes experts de la famille MYCIN-TEIRESIAS travaillent pour l'aide au diagnostic et à l'entrée de connaissances dans le domaine des infections bactériennes du sang.
LITHO en géologie du sous sol (Bonnet 82), GARI dans le domaine de la

conception de gammes d'usinage (Descottes 81), PROSPECTOR pour la prospection minière, SICLA en analyse de données (Demonchaux 84),AMEDIA pour le diagnostic de panne dans un réacteur nucléaire (Himbaut 84), SYGAL pour la classification automatique des galaxies(Granger 84a/) sont aussi des systèmes experts utilisant les règles de production. D'autres domaines sont aussi concernés: l'archéologie, le contrôle stochastique (Gomez 84) ou la physique (Dormoy 84).

Des logiciels ont été créés pour permettre d'élaborer le système expert adéquat selon le domaine abordé. Ce sont des moteurs d'inférences nus, avec un environnement de travail plus ou moins complet, tels que EMYCIN, OURCIN, Alouette (Mulet-Marquis 84) ou CRIQUET (Vignard 84a/, b/). Ce dernier regroupe l'essentiel des techniques et outils connus dans les systèmes experts jusqu'alors, afin de permettre un choix non restrictif et d'implémenter le mécanisme véritablement nécessaire. Ce système a déjà été utilisé par divers laboratoires publics et privés.
Une grande partie des systèmes experts créés à ce jour sont des systèmes de production.

5/ Avantages et inconvénients

Les systèmes de production à l'aide de règles sont trés utilisés tout d'abord parce qu'ils sont un moyen simple de représentation des connaissances en accord avec des études faites en psychologie. L'homme exprime habituellement sa connaissance en disant que faire dans une situation donnée. C'est le type de représentation fournie par les règles de production.

Le savoir humain est modulaire et la représentation à l'aide des règles de production permet de conserver cette caractéristique.

La connaissance est séparée des facons dont elle est manipulée et elle peut être donnée en vrac. La base de connaissances est facilement modifiable. Une règle n'incorpore qu'une petite quantité de connaissances. Elle est en théorie indépendante des autres.

La représentation est facile à lire, pour l'utilisateur mais aussi pour le système. La cohérence d'une base de connaissances sous forme de règles de production peut en partie être maintenue (Doyle 79, Ganascia 84).

Le système de production a des facilités d'explications. Il a connaissance de son propre savoir et de ses structures de contrôle, ce qui améliore ses communications avec l'homme. Le système peut ainsi beaucoup mieux expliquer son raisonnement, en fonction de ses connaissances.

Enfin l'expérience montre (MYCIN) qu'un tel système est souple et performant. Les règles de production tiennent compte étroitement des données particulières du cas traité, ce qui est une des raisons de leur efficacité.

Mais il peut être difficile d'exprimer une connaissance à l'aide de règles de production. Comme nous l'avons vu il faut en fait être dans un domaine adéquat. La modularité présentée comme un avantage n'est pas totale. Le mécanisme est souvent contrôlé, pour des raisons d'efficacité, par des artifices divers. L'indépendance des règles s'en trouve diminuée.

Comme le montre F Rechenmann dans (Rechenmann 84b/), la modularité existante entraîne une dispersion des éléments de connaissance constituant chaque règle de production. Les notions de classification hiérarchique et de contexte ne peuvent donc pas être utilisées. L'utilisation de règles de production a été illustrée par H Swaan Arons (Arons 83) pour aider à définir le modèle d'un système physique élémentaire, constitué d'une masse M accrochée à un ressort, compte tenu d'hypothèses a priori, telles que l'existence d'une force de friction ou l'importance de la masse du ressort. Deux règles proposées sont :

R13 : SI le ressort ne satisfait pas la loi de Hooke
ET le ressort est de masse nulle
ET il n'y a pas de force de friction
ET il n'y a pas de force extèrieure
ALORS
le modèle est $dx/dt + f(x)/M = 0$

R14 : SI le ressort ne satisfait pas la loi de Hooke
ET le ressort est de masse nulle
ET il n'y a pas de force de friction
ET il s'applique une force extérieure F(t)
ALORS
le modèle est $dx/dt + f(x)/M = F(t)/M$

Les vingt règles données définissent une classification de modèles possibles. Chaque modèle est une spécialisation différente du modèle général: $d^2x/dt^2 +$

$g(dx/dt) + f(x)/M = F(T)/M$. Pour traduire cette classification, il est normal que des règles aient des prémisses communes et soient d'une structure type :

SI P1 ET P2 ALORS modèle M1
SI P1 ALORS modèle M2

P1 représente un contexte dans lequel M1 et M2 sont des modèles possibles. P2 est une hypothèse discriminante. Comme on le voit avec les exemples de règles R13 et R14, la notion de contexte n'est utilisable qu'en dupliquant les informations communes, ce qui en fait élimine les notions de hiérarchisation et de contexte.

Enfin, cette représentation des connaissances repond très mal au besoin de représenter les objets manipulés, comme dans l'exemple l'entité "modèle".

Enfin si le nombre de règles est très important, sans stratégie particulière, le système devient inefficace.

Mais c'est encore sur de tels systèmes que se font à l'heure actuelle des recherches sur l'apprentissage et la gestion de cohérence (Barr 81).

VI/ Propriétés attendues d'une représentation des connaissances

Il est difficile de caractériser de façon assurée une "bonne représentation". Nous essayons donc tout au plus de donner une liste non exhaustive de caractéristiques et propriétés attendues.

La représentation permet une modélisation de la connaissance telle qu'elle a été définie précédemment, c'est à dire comme une combinaison d'information et de modes d'utilisation.

Toute la connaissance est explicitée. Mais la représentation comme l'expression des connaissances est simple. Une façon de maintenir la simplicité conceptuelle est de conserver une forme de représentation homogène et déclarative afin que les connaissances soient accessibles à tous et au système lui-même. Ce dernier point est essentiel pour faciliter les communications homme-machine et permettre au système d'expliquer son raisonnement. La représentation est modifiable et modulaire.

Les objets manipulés peuvent être décrits avec leur sémantique, c'est à dire leur signification propre en fonction du domaine d'utilisation. Les connaissances sont organisées, au moins hiérarchisées comme dans un réseau sémantique, afin de permettre en particulier l'utilisation de contextes.

Les connaissances de tous domaines sont exprimables, ainsi que des informations incertaines et incomplètes. Des valeurs par défaut peuvent être utilisées pour éviter un blocage du raisonnement par manque d'informations. Les objets représentés peuvent être vus et manipulés sous divers aspects et de différentes manières.

La plupart des représentations citées dans ce chapitre ne vérifient pas tous ces critères. Nous présentons dans le chapitre suivant une représentation mixte, réalisée sous forme de "frames"(Minsky 75) et utilisée dans des langages orientés objets. La représentation centrée objets proposée par la suite s'inspire de cette représentation.

CHAPITRE 2

REPRESENTATION DES CONNAISSANCES A BASE D'OBJETS

Chapitre II/ Représentation des connaissances à base d'objets

PLAN

I/ Notions d'objets

II/ La représentation centrée objets

1/ Présentation générale des frames
- a/ Description
- b/ L'héritage
- c/ Les réflexes
- d/ Utilisations des frames

2/ Problèmes d'utilisation de la représentation centrée objets

3/ Exemples d'utilisations des frames
- a/ CENTAUR
- b/ PATREC
- c/ GUS
- d/ KRL
- e/ FRL

III/ Les langages orientés objets

1/ Origines : SIMULA et SMALLTALK

2/ Autres langages
- a/ Les langages d'acteurs : Plasma, Planner
- b/ CEYX, Orbit
- c/ Les Flavors
- d/ LOOPS
- e/ Mering

IV/ Différences entre langages orientés objets et représentations centrées objets

V/ Avantages des représentations centrées objets

I/ Notions d'objets

Dans le formalisme relationnel classique (Delobel 82) un objet n'existe pas en tant que tel mais seulement comme participant à un ensemble d'énoncés dispersés dans la base de connaissances. Le travail de raisonnement consiste à manipuler ces énoncés afin d'en déduire de nouveaux. C'est le cas dans les systèmes de production. Dans l'exemple donné avec deux règles au chapitre précédent, l'objet "modèle" n'existe pas. Des énoncés dispersés dans les règles permettent de sélectionner le modèle adéquat. Les objets ne sont pas représentés comme des entités à part entière. Ils ne sont décris qu'a travers des propriétés disséminés dans les règles.

A l'opposé, dans le formalisme objets, les entités ont une existence réelle et un espace propre dans la base de connaissances. Les représentations en objets structurés sont nées de la conjonction d'idées de diverses sources. Elle s'inspirent des "schémas" résulats d'études en psychologie par Bartlett, des "frames" de Minsky (Minsky 75), des "scripts" de Schank (Pinson 83), des "objets" utilisés dans les langages orientés objets comme SMALLTALK (Goldbegr 83) ou des types abstraits.

Les entités rassemblent la connaissances associée à un objet. Le terme d'objet recouvre plusieurs notions. Il peut désigner un objet au sens physique, mathématique ou au sens du domaine concerné. Des informations descriptives et au sujet de la dynamique de l'objet décrit sont rassemblées.

Parmi les principales caractéristiques de la représentation à l'aide d'objets, on peut citer les trois suivantes :

-une représentation déclarative qui repose sur la notion d'objet. Le monde est vu comme un ensemble d'objets autonomes, chacun ayant une existence réelle dans la base de connaissances.

-chaque objet comporte à la fois une composante statique et une composante dynamique. On utilise ainsi une puissante combinaison de déclaratif et procédural.

-les notions de hiérarchie d'objets et d'héritage jouent un rôle essentiel dans l'écriture et la manipulation de la base de connaissances.

Cette représentation est utilisée dans deux catégories de logiciels: des

représentations centrées objets, dans lesquelles la dynamique est réduite, ou des langages orientés objets. Nous détaillons ces deux types de produits dans les paragraphes suivants.

II/ La représentation centrée objets

1/ Présentation générale des frames

L'idée des frames est issue des travaux de Minsky (Minsky 75). Ce dernier est parti de l'idée que nous avons en mémoire des structures stéréotypées d'informations et que nous sélectionnons chaque fois qu'une situation nouvelle se présente, l'un de ces stéréotypes, en tentant de le faire coincider avec les données de la situation courante. Le stéréotype identifié complète notre connaissance et indique alors comment réagir aux événements qui peuvent survenir.

La dynamique consiste à diriger la progression du système des frames devenus inadéquates vers d'autres frames quand la situation évolue.

Minsky essaie ainsi d'apporter une théorie unifiée et cohérente regroupant les divers aspects et intérêts des représentations créées jusqu'alors. Mais peu de travaux ont été faits sur la dynamique des frames, ce qui explique les interprétations différentes de cette notion et la difficultés à la définir exactement.

a/ Description

Le frame (Kuiper 75) est une généralisation des réseaux sémantiques dans le cas de "noeuds" complexes. C'est une structure de données composée d'un groupe de "slots" (attributs), chacun décrit par un ensemble de "facets" pouvant désigner un frame ou être un simple identificateur.

La structure se schématise ainsi :

```
(frame-nom
    (slot-1  (facet-11 val-11)
             (facet-12 val-12)
             ...
             (facet-1n val-1n))
    ...
    (slot-p  (facet-p1 val-p1)
             (facet-pq val-pq)))
```

Par exemple, un frame représente le "modèle-physique" manipulé au paragraphe précédent avec les slots: masse-ressort, loi-vérifiée, force-friction et force-traction.

Un frame modélise un concept. Un représentant du concept, ou instance du frame, est obtenu par instanciation, en remplissant les slots avec des valeurs correspondant au cas particulier traité.

b/ L'héritage

Les entités ainsi décrites sont reliées entre elles (Brachman 83a/) et forment le plus souvent une arborescence, parfois un réseau. Les liens hiérarchiques dans l'arborescence sont de type généralisation/spécialisation entre concepts et concept/instance entre un frame et tous ses représentants.

Ces liens permettent d'utiliser un mécanisme d'héritage par lequel un objet hérite des propriétés des entités qui lui sont hiérarchiquement supèrieures.

Le frame définissant le concept "modèle-physique" est:

```
(modele-physique
   (sorte-de (modele))
   (representants (M1 M2))
   (masse-ressort (defaut nulle))
   (loi-verifiee (si-besoin (interroger-utilisateur)))
   (force-friction)
   (force-traction))
```

Le concept "modele-physique" a pour ancêtre "modele". M1 et M2 sont des instances du frame "modele-physique".

A l'aide d'une facette spéciale (defaut), et en l'absence d'informations plus précises, des valeurs par défaut peuvent être données aux slots. Elles sont aussi héritées, ce qui résoud en partie le problème de l'incomplétude, cause d'échec de raisonnement dans bien des cas.

Des liens horizontaux entre entités sont utilisés à l'aide de références imbriquées. Un slot peut avoir pour valeur une indirection vers un autre frame.

c/ Les réflexes

Des facettes rendent les entités actives, à l'aide de réflexes. Ce sont des procédures attachées aux slots, automatiquement activées lors d'un certain type de manipulation sur le slot concerné. Le réflexe "si-besoin" par exemple, désigne la procédure à utiliser pour obtenir la valeur d'un slot si elle manque. Les réflexes "si-ajout" et "si-suppres" indiquent ce qu'il faut faire en cas d'ajout et de suppression de la valeur du slot.

L'attachement procédural est souvent le principal mécanisme d'exploitation de cette représentation. Des problèmes proviennent de la complexité des objets et du réseau qui les lie.

d/ L'utilisation des frames

Dans la plupart des systèmes, un frame spécifie la situation initiale avec les informations disponibles, décrivant ainsi le problème à traiter. Le raisonnement consiste à reconnaître dans le monde des objets connus, l'objet le plus précis et le plus proche de la situation donnée (Bonnet 84b/, Bobrow 77c/). Cet objet permet de compléter la description de la situation initiale et les informations supplémentaires qu'il apporte doivent être les réponses à toutes les questions posées.

Le processus d'identification utilisé est celui de "pattern-matching" ou filtrage, qui permet d'évaluer la proximité de deux frames.

2/ Problèmes d'utilisation de la représentation centrée objets

Cette représentation regroupe des avantages de représentations déclaratives et procédurales. Comme une représentation déclarative, elle est facile à lire et permet d'expliciter un maximum d'informations de tous genres, de façon modulaire et modifiable. L'attachement procédural facilite l'expression d'heuristiques et de connaissances au sujet de la dynamique des objets.
La base de connaissances organisée permet d'utiliser des notions de contextes et de hiérarchie. L'héritage alors possible facilite la spécification des informations et permet un certain accroissement des connaissances.
Les valeurs par défaut (Rich 83) améliorent les performances du système en évitant l'arrêt par manque d'informations.
Le caractère mixte de cette représentation permet de modéliser plus de connaissances que d'autres reprèsentations.
La manipulation d'informations incertaines est possible avec des coefficients de crédibilité comme dans un système de production.

Mais la structure des frames peut paraître complexe. L'insertion directe du procédural sous forme d'appels de programmes dans les réflexes nuit à l'homogénéité et lisibilité de la représentation.

L'héritage et la gestion de cohérence dans une base de connaissances complexe sont aussi difficiles à gérer. Ces problèmes sont déjà abordés dans différents domaines et des solutions plus ou moins efficaces sont proposées (Delobel 82, Doyle 83, Farreny 81).

Le problème essentiel concerne le mode d'exploitation de cette représentation. Le mécanisme d'exploitation classique exposé au paragraphe précédent utilise un processus de pattern-matching trop strict, basé sur des correspondances exactes et syntaxiques entre les noms des slots. Il n'utilise donc qu'une faible partie des informations apportées dans la description, puisqu'il se contente du nom des slots sans considérer leur sémantique, c'est à dire leur contenu et signification propre.

Le pattern-matching rend une réponse binaire, 1 pour une correspondance complète exacte entre les objets mis en correspondance et 0 sinon. Cette information n'est pas assez nuancée pour être réellement utilisable dans un mécanisme censé simuler un raisonnement humain. Aucune imprécision ou incomplétude ne peut être manipulée. Le raisonnement ainsi construit est trop

strict pour permettre une exploitation correcte de cette représentation qui par ailleurs comporte un grand nombre d'informations de tous genres.

3/ Exemples d'utilisations des frames

a/ CENTAUR

La connaissances de ce système est représentée par une combinaison de frames et de règles de production (Aikins 79), tout comme dans YAPS (Allen 83). Le système donne des conseils diagnostics dans le domaine des maladies pulmonaires.

Les frames explicitent les contextes, c'est à dire les faits et préconditions qui sont pris en compte pour décrire la situation et guider l'utilisation de règles de production donnant une connaissance plus fine. L'exploitaion se fait à l'aide d'un processus de filtrage, comme décrit précédemment, avec divers types de règles :

-des règles d'inférences pour obtenir des valeurs pour les slots

-des règles de synthèse permettent une interprétation globale de l'état du sujet.

-des règles de raffinement permettent de construire un diagnostic final.

-des règles d'activation aident à formuler des hypothèses.

-des règles de faits résiduels permettent de vérifier la complétude du diagnostic.

L'interpréteur de CENTAUR exécute les tâches qui figurent dans son agenda suivant l'ordre d'apparition et s'arrête quand celui-ci est vide. L'agenda est mis à jour en fonction des tâches décrites dans les slots.

b/ PATREC

C'est une aide intelligente à la gestion d'une base de données pour saisir des informations de manière souple et répondre à des requêtes de façon perspicace, en réalisant les inférences nécessaires (Mittal 84).

La représentation utilisée correspond aux frames classiques. Les concepteurs ont tenté d'introduire une notion temporelle. Des descripteurs temporels permettent de spécifier pour un événement donné sa date de début, sa durée et ses liens dans le temps avec d'autres événements.

c/ GUS

GUS (Genial Understander System)(Bobrow 77d/) a été conçu en tant qu'interface intelligente capable de soutenir une discussion en anglais dans un domaine donné. L'étude a permis de mettre en évidence les principaux composants d'un système intelligent de compréhension du langage.

Les concepts, sous forme de frames classiques, représentent les divers composants d'un dialogue. Pour conduire le dialogue, le système génère une instance du frame "Dialogue", le plus général. Il cherche ensuite à remplir chacun de ses slots par la méthode "en profondeur d'abord" ce qui lui permet de garder l'initiative dans le dialogue. Le contrôle est totalement réparti. La progression provient des réflexes attachés aux slots qui entraînent la création des instances nécessaires pour construire progressivement la hiérarchie de modules représentant le dialogue.

D'autres systèmes ont été implémentés avec une représentation à base d'objets classiques. NUDGE (Goldstein 80) permet de construire des emplois du temps. INTERNIST (Fieschi 84) est un programme d'aide au diagnostic qui couvre 80% de la médecine interne. PIP (Present Illness Program)(Fieschi 84) contient environ 70 frames décrivant des maladies rénales pour aider au diagnostic dans ce domaine.

Des langages ont aussi été crées avec cette représentation, pour faciliter la spécification des connaissances et l'élaboration de tels systèmes.

d/ KRL

"Knowledge Representation Language" (Bobrow 77c/) est un langage pour la représentation des connaissances, dont une première maquette, KRL-0, a été réalisée en 1976. Les frames sont là encore utilisés tels quels, mais la représentation d'un même concept regroupe plusieurs descripteurs permettant d'avoir plusieurs vues du même objet. Il existe en KRL, sept sortes d'entités: de base, abstraction, spécialisation, individu, manifestation, relation et proposition. Douze types de descripteurs sont utilisables pour spécifier diverses vues de chaque entité.

Le fonctionnement est basé sur le pattern-matching classique dans KRL-0. Mais les concepteurs annoncaient (Bobrow 77c/) pour une prochaine version un processus de filtrage permettant la manipulation de la sémantique des objets et un raisonnement plus nuancé.

e/ FRL

"Frame Representation Language" (Goldstein 80) est utilisé dans le système NUDGE cité précédemment. IL s'inspire de KRL en restant beaucoup plus simple et en respectant de trés prés la description des frames donnée au paragraphe précédent.

Dans NUDGE, cinq types de concepts sont utilisés: activité, personne, place, temps et objet. Il y a environ cent entités hiérarchisées. Dans chaque concept les attributs "commentaires", "valeur", "valeur par défaut" et "contraintes" sont utilisés.

D'autres langages encore ont été implémentés. OWL (Szolovits 77) a été conçu pour représenter l'anglais. Il utilise une représentation à base de réseaux sémantiques avec des noeuds structurés proches des frames. KLONE (Goodwin 79) est aussi plus proche des réseaux sémantiques que des frames. Ce dernier langage dévelope la notion de hiérarchisation des concepts et met en évidence des problèmes particuliers d'héritage dans le cas où une entité a plusieurs ancêtres.

Les systèmes et langages qui manipulent cette représentation en ont surtout utilisé les possibilités de description statique et d'organisation. Mais la dynamique est limitée, basée sur un processus de filtrage trop restrictif.

Nous présentons dans le paragraphe suivant les langages orientés objets, autre famille de logiciels utilisant cette représentation, qui offrent une spécification particulière de la dynamique sans pour autant résoudre les problèmes cités.

III/ Les langages orientés objets

La programmation orientée objets, à l'aide de langages adéquates, constitue un style de programmation au même titre que la programmation procédurale ou fonctionnelle.

1/ Les origines : SIMULA et SMALLTALK

Le concept d'objet provient du langage SIMULA (Briot 83) pour être ensuite vulgarisé par SMALLTALK (Goldberg 83, Cointe 82).

SIMULA est le premier langage a introduire les concepts d'objets et de classes en programmation. Une classe, sous forme d'un modèle de structure de données, décrit un type abstrait. L'utilisateur génère dynamiquement, par instanciation, des exemplaires de cette structure. Les résultats obtenus sont appelés des objets. On appelle attributs les représentations concrètes des caractéristiques et propriétés du modéle défini par la classe. Ce sont des variables et des procédures.
Un objet est donc une entité ayant deux composantes:

-la composante statique caractérisée par les attributs

-la composante dynamique se réduisant à des actions d'initialisations.

SIMULA, pour regrouper des attributs communs, introduit déjà la notion de hiérarchie de structures de données. On peut définir une classe B, déclarée

sous-classe de A, qui hérite donc de celle-ci.

SMALTALLK héritier direct de SIMULA dégage la composante statique des objets de SIMULA pour en faire les uniques entités de son univers. Il intègre également la transmission de messages comme unique moyen de communiquer avec et entre les objets. Ce mécanisme provient des langages d'acteurs (Granger 82).

Un objet SMALLTALK est défini par:

objet = base de connaissances + base de procédures

De façon sommaire, les caractéristiques essentielles de SMALLTALK sont:

-toute entité de l'univers est un objet

-tout objet est l'instance ou représentant d'une classe.

-une classe regroupe les objets ayant des caractéristiques et comportements communs. Une déclaration de classe spécifie les caractéristiques communes à l'aide de variables appelées "champs", et les comportements communs sous forme de procédures appelées "méthodes". On distingue les variables d'instances dont les valeurs sont propres à chaque instance et les variables de classe dont les valeurs associées sont communes à toutes les instances de la classe, spécifiant des connaissances propres à la classe.

-les objets communiquent entre eux par envoi et reception de messages. Un message mentionne le receveur suivi du message proprement dit. Le seul moyen d'accéder aux attributs est la transmission de messages.

SMALLTALK a beaucoup évolué depuis sa création, mais il reste un exemple type de langage orienté objets, que l'on peut en toute généralité caractériser ainsi (Bonnet 84b/):

-langage interactif et extensible

-une structure de données basée sur la notion d'objet décrite précédemment, avec ses caractéristiques et avantages.

-la dynamique est spécifiée dans chaque objet à l'aide de méthodes qui sont des procédures pour la manipulation des objets. Les communications avec et entre

objets ne peuvent se faire que par l'envoi de messages ce qui uniformise la représentation mais favorise le procédural.

2/ Autres langages

a/ Langages d'acteurs: Plasma, Planner

Les langages d'acteurs tels que Plasma (Granger 82), Planner, Act et Formes (Briot 82) ont la notion de messages en commun avec les langages orientés objets. Mais la structure de données utilisée est moins proche de la notion d'objet que de celle de procédure. Le concept d'acteur généralise celui de procédure. Un acteur reçoit des messages et aprés leur traitement envoie des messages à d'autres acteurs qu'il détermine. Le passage de messages est l'unique structure de contrôle. Il est possible de préciser dans le message à quel acteur il faudra envoyer une réponse: c'est le mécanisme de continuation. Le mécanisme de réception des messages est basé sur le filtrage, qui permet à l'acteur de sélectionner les messages acceptables.

b/ ORBIT, CEYX

Des langages tels que ORBIT(Steels 83), CEYX (Hullot 84), GLISP (Gordon 82) et Objvlisp sont des langages orientés objets classiques, c'est à dire correspondant à la définition donnée précédemment, mais qui ont pour interêt principal d'être implémentés à partir d'un dialecte LISP (Maclisp, LeLisp (Chailloux 83), Interlisp, Vlisp). Ils bénéficient de leur facilité de manipulation et d'un environnement de travail souvent riche.

c/ Les Flavors

"Les Flavors" (Weinreb 80) est un langage orienté objets créé en 1979 au Massachusetts Institutes of Technology pour la LISP-Machine. Au contraire de la plupart des langages de la même famille, il permet une organisation des entités plus complexe qu'une simple hiérarchisation.

Un objet peut avoir plusieurs ancêtres précisés dans un attribut particulier, ce qui fait apparaître le problème de l'héritage multiple. Il y a conflit pour une caractéristique ou propriété si celle-ci peut être héritée de plusieurs ancêtres différents. Le problème est à traiter pour les langages orientés objets au niveau structurel pour les attributs et au niveau dynamique pour les méthodes.

Il est résolu pour les attributs en associant une priorité d'héritage à l'ordre de parcours dans la liste des ancêtres.

Dans "Les Flavors", la méthode héritée est une combinaison des méthodes en conflit.
Un "Flavor" est l'équivalent d'une classe de SMALLTALK à laquelle est associée une technique particulière de combinaison des méthodes.
Les méthodes dans les Flavors sont qualifiées à l'aide d'un mot clé choisit parmi: "before", "primary", "after". La technique de combinaison des méthodes la plus souvent utilisée est la suivante:

selon l'ordre de lecture des ancêtres,
-appliquer séquentiellement toutes les méthodes typées "before"

-appliquer la première méthode typée "primary"

-appliquer séquentiellement, dans l'ordre inverse de celui de lecture, toutes les méthodes typées "after".

Le problème du multi-héritage subsiste en fait même aprés divers essais. La solution proposée dans ce langage alourdit la description des objets et ne fournit pas toujours des résultats satisfaisants pour les méthodes héritées.

d/ LOOPS

Enfin, des langages orientés objets, plus récents, traitent le problème du multi-héritage et se rapprochent de diverses manières des représentations centrées objets.

LOOPS (Bobrow 82a/b/) est une extension de Interlisp et réunit quatre styles de programmation:

- programmation procédurale classique

- programmation dirigée par les objets

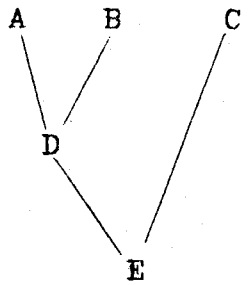
- programmation dirigée par les données à l'aide de valeurs actives

- programmation par les règles

Le langage intègre la notion d'objets, analogues à ceux de SMALLTALK, avec des réflexes propres aux représentations centrées objets. On peut ainsi parler de programmation dirigée par les données, puisque les réflexes activent des procédures suite à des manipulations de données.

Des règles de production peuvent aussi être codées et manipulées comme telles.

Le multi-héritage est pris en compte. Un attribut dans chaque objet spécifie la liste des ancêtres, parcourue de façon classique, en profondeur d'abord puis de gauche à droite. Donc de manière récursive, lors du parcours de la liste des ancêtres, le système considère d'abord les ancêtres de chaque objet, puis son successeur dans la liste. On peut en LOOPS moduler l'héritage en annulant l'aspect récursif dans le parcours de la liste des ancêtres. Soit l'organisation:



Dans l'obet E, les ancêtres sont donnés par (D, C). Sans plus de précision, les ancêtres pris en compte lors de l'héritage sont dans l'ordre: A, B, D, C. Pour éviter le parcours en profondeur, LOOPS permet d'écrire la liste des ancêtres comme suit: ((D), C). Dans ce cas, les ancêtres utilisés sont dans l'ordre: D, C.

Pour résoudre le multi-héritage, la priorité de parcours, selon le chemin indiqué, est utilisée comme priorité d'héritage pour les attributs et les méthodes. Des techniques plus fines permettent de combiner des méthodes, mais non de façon automatique.

LOOPS est un langage avec beaucoup de possibilités, mais trop hétérogène pour être utilisable facilement et efficacement.

Kool (Albert 83) est un langage de même type, mais moins développé.

e/ MERING

Dans Mering (Ferber 84) la frontière devient plus floue entre un concept et ses représentants. Des réflexes sont utilisables et le multi-héritage est aussi traité avec une priorité de parcours.

Une généralisation intéressante de la notion de méthode est faite. Des méthodes classiques permettent l'envoi de messages entre objets et la manipulation de l'objet dans son ensemble. Des applicateurs permettent la distribution des messages aux attributs d'un objet. On se rapproche en cela des représentations centrées objets dans lesquelles les informations sont toutes distribuées sur les attributs, sans informations globales telles que les méthodes dans les objets de SMALLTALK.

Cette étude trés sommaire nous a permis de présenter les caractéristiques essentielles de divers logiciels basés sur la notion d'objet. Nous mettons en évidence dans les paragraphes suivants les principales différences entre ces produits, leurs faiblesses et les améliorations à apporter qui seront les objectifs de notre étude.

IV/ Différences entre langage orienté objets et représentation centrée objets

-Les représentations à base d'objets se développent et sont utiles et utilisées dans les domaines où une base de données dynamique peut être spécifiée. Mais il y a souvent confusion entre langages orientés objets et représentations centrées objets, due en grande partie au manque de développement de ce dernier type de logiciel et à l'apparition de langages tels que LOOPS et Mering qui combinent ces deux types d'outils.

-Dans les langages orientés objets, l'aspect déclaratif est beaucoup moins développé. La description des objets comporte moins de données sous forme déclarative. Les facettes associées aux slots se réduisent la plupart du temps à une valeur et une valeur par défaut.

Les facilités d'expression en langage naturel sont pauvres ou inexistantes.

La dynamique des objets est spécifiée sous forme procédurale à l'aide de méthodes décrivant les actions que l'objet sait faire ou peut subir. Elle concerne l'ensemble de l'objet et non pas chacun de ses attributs. Chaque objet ne peut donc être utilisé que d'une seule manière, celle définie par les méthodes globales. Une telle spécification de la dynamique nuit à l'uniformité, lisibilité et modularité de la représentation et aux possibilités d'explications du système. On retrouve un problème de la représentation procédurale.

Enfin, l'organisation des objets est souvent réduite à une simple arborescence.

-Une représentation centrée objets est plus déclarative. De nombreuses facettes sont utilisées. Certaines, à l'aide de réflexes permettent de distribuer la dynamique sur les attributs. Mais peu ou aucune information ne concerne l'objet dans sa totalité. On peut donc utiliser de façon autonome une partition de la description globale du concept. Plusieurs vues du même concept sont ainsi manipulables.
La notion de messages est totalement absente. Les échanges entre objets ou passage de l'un à l'autre sont à la charge d'un processus global de filtrage.
La représentation est plus uniforme et lisible. La dynamique distribuée sur les attributs est plus explicite, donc plus accessible au système.

Enfin, une organisation plus complexe de la base de connaissances est possible, avec multi-héritage.

-La différence entre ces deux familles de logiciels basés sur la notion d'objet porte essentiellement sur la dynamique. Dans les langages orientés objets, la dynamique concerne l'objet dans son ensemble. Elle est exprimée à l'aide de messages et méthodes sous forme procédurale.

Dans une représentation centrée objets, la dynamique est distribuée sur les attributs à l'aide de réflexes, inexistants dans les langages précédents. Les réflexes ne désignent en théorie que des tâches ponctuelles et permettent de diminuer l'importance du procédural. La dynamique dans la base consiste en la recherche d'un objet satisfaisant certaines caractéristiques. Un processus de filtrage permet cette recherche, en utilisant les informations statiques et dynamiques fournies dans les facettes. Mais, comme nous l'avons vu dans le paragraphe précédent, ce mécanisme n'est pas assez développé pour permettre une utilisation correcte de la représentation.

V/ Avantages des représentations centrées objets

A partir de la description des représentations utilisées en Intelligence Artificielle, nous synthétisons les avantages de la représentation centrée objets décrite dans ce chapitre comme suit:

-elle permet de représenter les objets en tant que tels et non pas sous forme de bribes de connaissances dispersées dans la base.

-c'est une représentation déclarative, modulaire et modifiable dans laquelle des informations de tous genres sont notées de façon explicite et accessible. Ce mode des spécification est naturel dans beaucoup de domaines (Gouze Vignard 84, Barbuceanu 84, Rechenmann 84b/).

-le système a facilement accés à ses connaissances ce qui facilite et améliore ses communications avec l'utilisateur et en particulier la qualités des explications qu'il doit fournir sur son raisonnement.

-l'attachement procédural, avec les réflexes, permet de distribuer la dynamique sur les attributs. Il n'y a plus d'importantes procédures représentant pour le système une séquence difficile à maîtriser et à expliquer en détail. Les réflexes tels que "si-ajout" et "si-suppres", cités précédemment, permettent la propagation dans la base de connaissances des modifications et ainsi la gestion de

cohérence entre les objets.

-toutes les informations, statiques et dynamiques, sont distribuées sur les attributs. La structure des objets est beaucoup plus modulaire et on peut manipuler différents aspects d'un objet, par exemple en considèrant à chaque fois une partition différente de l'ensemble des attributs.

-la base de connaissances est organisée, au minimum hiérarchisée. Le mécanisme d'héritage alors possible facilite la spécification des connaissances. Cette organisation améliore aussi les performances du système et évite l'explosion combinatoire qui apparaît dans les systèmes de production classiques du fait de la non organisation de la base de connaissances. Les valeurs par défaut offrent une autre facilité de description et améliorent le raisonnement.

Mais il subsiste des problèmes. Comme nous l'avons vu au sujet des frames, la représentation n'est pas homogène et déclarative du fait de l'insertion fréquente dans les objets d'appels de programmes. Le langage de description offert reste pauvre.

Le sujet de notre étude est donc en premier lieu d'améliorer cette représentation sur les points suivants:

-avoir une représentation uniformément déclarative et homogène. Les appels de programmes importants sont remplacés par une description du résultat voulu ou du traitement nécessaire. Le système élabore la dynamique à partir de ces informations.

-faciliter la spécification des connaissances en permettant l'utilisation de termes du langage naturel.

Le sujet principal de notre travail concerne l'amélioration de la dynamique et de l'utilisation de cette représentation. Nous étudions pour cela un mécanisme de filtrage qui permette:

-d'implémenter toute la dynamique nécessaire dans la représentation définie auparavant, en particulier pour offrir toutes les techniques d'inférences nécessaires;

-de manipuler des notions imprécises et incomplètes exprimées à l'aide de

termes flous du langage naturel, ainsi que des notions d'importance des éléments dans la description des objets;

-de manipuler un même objet sous différents aspects;

-d'utiliser la sémantique des objets, c'est à dire leur contenu plutôt que de se satisfaire de leur nom;

-un raisonnement nuancé et rende une évaluation de distance entre les objets plutôt qu'une réponse binaire trop stricte;

-des raisonnements par analogie et typologie.

CHAPITRE 3

LA REPRESENTATION CENTREE OBJETS PROPOSEE

Chapitre III/ La représentation centrée objets proposée

PLAN

I/ Les schémas

II/ La dynamique sous forme déclarative

III/ Deux types de schémas : concept et individu

IV/ Organisation des objets

V/ Héritage et multi-héritage

1/ Héritage

2/ Multi-héritage

VI/ Organisation dynamique de la base de schémas

VII/ Différentes vues d'un même schéma

VIII/Gestion de cohérence

1/ Les réflexes

2/ Le mécanisme d'affectation

IX/ Conclusion

Nous présentons dans ce chapitre une représentation essentiellement tirée des travaux de F Rechenmann (Rechenmann 84a/ 85).
Afin de proposer un outil plus souple et facile à manipuler que ceux développés jusqu'alors, nous proposons une représentation simple plus uniforme et déclarative.

I/ Les schémas

Nous définissons un schéma comme une liste à trois niveaux d'imbrication. Au premier niveau se situe le nom du schéma, au deuxième la liste de ses attributs. Chaque attribut est décrit au troisième niveau par une liste de facettes, chacune possédant une valeur. Dans le formalisme LISP, un schéma est représenté par une liste (Winston 81):

```
(nom-schéma
   (attribut 1  (facette 11 valeur 11)
                (facette 12 valeur 12)
                  ...
                (facette 1n valeur 1n))
     ...
   (attribut m  (facette m1 valeur m1)
                  ...
                (facette mp valeur mp)))
```

Les attributs sont choisis par le concepteur et apportent la sémantique de l'objet, c'est à dire sa signification propre, alors que les facettes font partie d'un ensemble prédéfini.

Les informations sont réparties au maximum sur les attributs. Les informations globales associées à une entité sont regroupées dans un attribut particulier nommé SELF. Ces informations, en quantité réduite, concernent par exemple la liste des attributs à utiliser si l'on désire imprimer un objet. Elles définissent une façon par défaut de voir et manipuler un objet, c'est à dire une "vue standard" de l'objet.

Par définition, toute valeur pour toute facette est un filtre, c'est à dire un descripteur plus ou moins précis d'un schéma.
Chaque attribut est typé à l'aide d'une facette "type". Le type d'un objet est

aussi un schéma qui correspond à un noeud dans la hiérarchie des objets en dessous duquel se situent les schémas pouvant être donnés comme valeur à l'attribut concerné.

Un exemple simple nous permet d'introduire plus précisément le formalisme retenu. Cette représentation peut être utilisée en particulier pour développer des logiciels d'aide à la modélisation (Gouze Vignard 84a/).
Le modèle physique qui régit un système élémentaire constitué d'une masse M accrochée à un ressort, compte tenu des forces de friction et de traction et de la masse du ressort sera défini à l'aide du schéma:

```
(Modèle-physique
   sorte-de
     $valeur Universel

   spec
     $valeur spec-modèle-physique

   masse-ressort
     type UN IND MASSE
     $valeur
     val-defaut faible
     si-besoin  (Interroger-utilisateur !masse-ressort)
     importance 1.0

   loi-physique
     type UN IND LOI
     $valeur
     importance 0.5

   force-friction
     type UN D SYMBOLE
     $valeur
     importance 0.4

   force-traction
     type UN D SYMBOLE
     $valeur
     importance 0.4
```

```
commentaires
  type UN IND TEXTE
  $valeur
  importance 0.3

modèle-mathematique
  type UN IND Modèle-math
  $valeur
  importance 0.3   )
```

Les facettes font partie d'un ensemble prédéfini. Leur valeur est toujours un filtre. Les facettes essentielles et les plus utilisées sont:

-type: il donne le "schéma type" de l'attribut et indique ainsi le contenu, c'est à dire la signification propre de l'attribut, c'est à dire sa sémantique.

-$valeur: elle donne la valeur, unique, de l'attribut. Ce peut être un terme désignant de façon précise le schéma valeur. Mais ce peut être aussi une description, sous forme d'un schéma avec plus ou moins d'informations, des objets suceptibles de constituer la valeur de l'attribut.

-val-defaut: elle donne une valeur par défaut, c'est à dire une valeur admissible favorisée. Cette valeur n'est retenue qu'en l'absence d'autre information et lorsque les autres moyens d'obtention de la valeur ont échoués.

-si-besoin:
elle permet d'associer aux attributs des méthodes de calcul des valeurs.

-si-échec, si-succés: pour spécifier les comportements à tenir en cas d'échec ou de succés d'obtention de la valeur pour l'attribut.

-importance: contient un nombre entre 0 et 1 qui donne l'importance relative de l'attribut par rapport à l'objet dans son ensemble.

-domaine: contient une liste en extension des valeurs possibles. Cette facette permet de spécifier des contraintes de façon explicite donc plus accessible et maniable.

-certitude: contient un nombre entre 0 et 1 qui permet à l'utilisateur de forcer une certitude pour une valeur donnée s'il le désire.

-a-verifier: spécifie un prédicat portant sur la valeur afin de la restreindre.

D'autres facettes permettent d'utiliser les réflexes tels que si-ajout, si-modif et si-suppres, décrits dans le chapitre prècédent au sujet des frames.

II/ La dynamique sous forme déclarative

La dynamique est distribuée sur les attributs dans les facettes "$valeur" et "si-besoin" qui décrivent alors les actions à entreprendre pour obtenir la valeur des attributs.

La dynamique est spécifiée de façon procédurale à l'aide de fonctions directement évaluables ou de façon déclarative à l'aide de "schémas traitements".

Ces derniers sont structurellement identiques aux autres. Comme leur nom l'indique, ils décrivent le ou les traitements utilisables pour obtenir la valeur cherchée. Certains attributs décrivent les conditions d'application du traitement et les paramètres nécessaires, d'autres correspondent aux résultats. Les réflexes "si-besoin" de ces derniers attributs activent directement les programmes adéquats. La dynamique est ainsi décrite de façon déclarative et reste accessible et modifiable.

Un exemple, dans le domaine de la modélisation avec des systèmes différentiels, présente simplement ce mode de représentation . Les schémas qui définissent les deux méthodes d'intégration Runge-Kutta d'ordre 4 et de Gear en dessous d'un schéma général d'intégration sont:

```
(Integration
    sorte-de
      $valeur Traitement

    spec
      $valeur (Runge-Kutta4 Gear)

    syst-differentiel
      type UN IND Systeme-diff
      $valeur
      si-besoin (Interroger-utilisateur !syst-diferentiel)
```

```
    importance 0.5

  val-initiale
    type UN D NOMBRE
    $valeur
    si-besoin (Interroger-utilisateur !val-initiale)
    importance 0.5

  type-systeme
    type UN D SYMBOLE
    $valeur
    si-besoin (Interroger-utilisateur !type-systeme)
    importance 1.0

  pas
    type UN D SYMBOLE
    $valeur
    domaine (adaptatif non-adaptatif)
    val-defaut adaptatif
    importance 0.5

  erreur
    type UN D Erreur
    $valeur
    val-defaut 1/2*precision-machine
    importance 0.5

  solution
    type UN D Solution
    $valeur
    importance 0.5)


(Ruge-Kutta-4
  sorte-de
    $valeur Integration

  type-systeme
    $valeur classique
```

```
    solution
      si-besoin (calcul-rk4 !syst-differentiel
                        !val-initiale !pas !erreur)


(Gear
    sorte-de
      $valeur Integration

    type-systeme
      si-besoin (test-rapport-coeff !syst-differentiel)
      domaine (raide)

    solution
      si-besoin (calcul-gear !syst-differentiel
                        !val-initiale !pas !erreur)
```

Les traitements "Runge-kutta-4" et "Gear" sont définis par spécialisations de "Intégration". Pour faciliter les notations, "!val-initiale" fait référence à la valeur de l'attribut "val-initiale" du schéma en cours de manipulation.

Dans l'attribut "type-systeme" du schéma décrivant la méthode de Gear, le réflexe "si-besoin" active un programme qui indique si le système différentiel utilisé est raide ou pas, à partir de tests sur l'importance relative des coefficients dans le système.

Le réflexe "si-besoin" des attributs "solution" dans les deux méthodes désignent les programmes à utiliser avec les paramètres nécessaires.

Pour intégrer un système différentiel S avec les valeurs initiales X à l'aide de la méthode de Gear, il suffit d'activer un traitement d'intégration de Gear avec le filtre:

```
(Gear
    (syst-differentiel $valeur S)
    (val-initiale $valeur X))
```

Ce schéma peut être placé dans la facette "$valeur" de l'attribut "solution" faisant partie du schéma qui décrit un système différentiel donné au parapraphe 4. La dynamique pour intégrer un système différentiel est ainsi spécifiée de

façon déclarative.

Des "schémas prédicats" permettent de spécifier sous forme déclarative des tests donnés en valeur à la facette "a-verifier". Un exemple simple est donné ci-dessous:

```
(Equation-diff
   sorte-de
     $valeur Universel
   spec
     $valeur Equation-diff-lin
   membre-gauche
     type UN IND Var-etat
     $valeur
     importance 0.5
   membre-droit
     type UN IND Membre
     $valeur
     importance 0.5
   variables
     type LISTE IND Variable
     $valeur
     importance 0.5 )


(Equation-diff-lin
   sorte-de
    $valeur Equation-diff
   membre-droit
    a-verifier (Linearite
                (texte $valeur !membre-droit)))


 (Linearire
   sorte-de
    $valeur Traitement
   texte
    type UN D SYMBOLE
    $valeur
    a-verifier (test-linearite !texte ?x)
   succes
    type UN D Booleen
```

$valeur ?x
importance 0.5)

Ce dernier schéma permet d'indiquer de façon déclarative le type de prédicat à utiliser. C'est dans ce schéma que le réflexe "a-verifier" de l'attribut "texte" appelle le programme de calcul symbolique qui va vérifier la linéarité du texte passé en paramètre.

L'idée de base est de retarder l'utilisation du procédural classique en explicitant au mieux auparavant les résultats cherchés et les traitements à utiliser.

Les actions ponctuelles, sans importance et ne nécessitant pas d'explications détaillées sont données sous forme procédurale dans les réflexes.

Les schémas traitement sont utiles pour décrire l'appel de programmes réalisant des tâches importantes et spécifiques au domaine d'application, comme le traitement d'intégration de Gear en modélisation mathématique.

La dynamique est ainsi spécifiée de façon déclarative et uniforme. Elle est facilement modifiable. Le systéme la connaît et peut donc mieux expliquer et maîtriser son raisonnement. Il n'y a plus ou peu d'actions procédurales importantes, longues, inaccessibles et incontrôlables par le système. Comme nous le verrons au paragraphe suivant, le système, à l'aide de recherches successives, élabore la dynamique qu'il évalue la meilleure. Il sait choisir parmi un ensemble de candidats, le schéma traitement le plus adéquat.

Dans le cas du traitement d'intégration, le système est capable de choisir entre la méthode de Runge Kutta d'ordre 4 et la méthode de Gear. Il peut ensuite décrire le traitement utilisé car celui-ci est spécifié sous forme déclarative. Dans la description du schéma, des attributs comme "type-systeme" fournissent les éléments justifiant l'appel du traitement choisi.

Dans les frames classiques, avec la dynamique spécifiée uniquement dans les réflexes sous forme d'un appel direct de programmes, le système ne pourrait qu'indiquer le nom du programme utilisé sans donner plus d'informations sur la signification du traitement et les raisons de son utilisation.

Les schémas traitements sont utilisables dans les facettes "$valeur" et "si-besoin". Le mécanisme d'exploitation que nous décrivons au chapitre suivant sait reconnaître ce type de spécification pour effectuer les recherches et les

évaluations voulues afin de construite la dynamique nécessaire à l'aide des schémas donnés.

III/ Deux types de schémas: concept et individu

Comme nous l'avons vu au sujet des Frames, les objets sont organisés et au minimum hiérarchisés. Nous distinguons deux types de schémas. Les "concepts" et les "individus". Ces derniers sont des "représentants" d'un concept, c'est à dire des noeuds terminaux dans la hiérarchie des entités.

*-On définit des individus en instanciant complétement ou partiellement un concept. Un représentatnt du concept "Modèle-physique" définit précédemment est:

```
(Modèle-physique-1
    est-un
     $valeur Modèle-physique

    masse-ressort
      $valeur nulle

    loi-physique
      $valeur Hooke

    force-friction
      $valeur lineaire)
```

Les informations complémentaires sont héritées de "Modèle-physique". Tout individu est lié au plus à un concept par une relation hiérarchique donnée dans l'attribut "est-un". Inversement, la liste des représentants d'un concept apparaît dans l'attribut "l-instances" de celui-ci.

*-Un concept ne peut être défini que par combinaison et spécialisation de l'existant, c'est à dire d'un ou plusieurs autres concepts. Aucune information ne peut être supprimée. On peut, lors de la création, ajouter des attributs, des facettes ou restreindre des valeurs déjà données dans les concepts utilisés comme ancêtres.
La restriction peut être faite en affectant un type plus spécialisé, en

restreignant le domaine des valeurs possibles ou les conditions associées ou en affectant une valeur à l'attribut dans $valeur.
Ainsi, à partir du même "Modèle-physique", on peut définir les "Modèle-physique-Hooke" qui sont ceux qui décrivent un phénomène vérifiant la loi de Hooke.

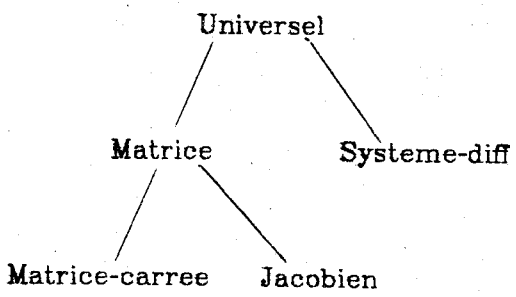
On spécialise l'attribut "loi-physique" en forcant la valeur à "Hooke":

```
(Modèle-physique-Hooke
   sorte-de
     $valeur Modèle-physique

   loi-physique
     $valeur Hooke)
```

Les individus sont aussi nommés "instances". L'ensemble des instances constitue la base de données et l'ensembles des concepts la base de connaissances. Les concepts sont, au moins pour les plus fondamentaux, spécifiés par le concepteur, alors que les instances sont directement ou indirectement le résultat d'une action de l'utilisateur. Pour l'implémentation, il est intéressant de distinguer les instances des autres objets, car elles sont trés nombreuses et sont gérées pour cette raison dans une mémoire relationnelle (Bensaid 84).

IV/ L'organisation des schémas

*-Tout concept s'inscrit dans une hiérarchie taxinomique définie par une relation de généralisation/spécialisation (Brachman 83, Goodwin 79). Ces liaisons sont décrites dans des attributs particuliers, tels que "sorte-de" et "spec" qui indiquent respectivement les ancêtres et les fils du concept concerné. Elles sont aussi appelées "liens verticaux". Un objet "Universel" est racine de toute la hiérarchie. Un exemple simple de hiérarchie est le suivant:

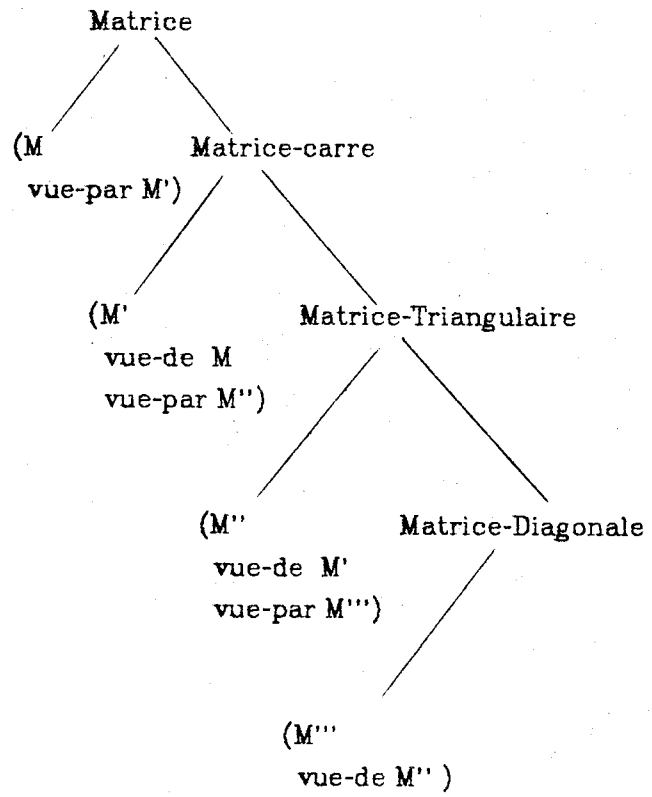


*-Les individus, comme nous l'avons vu au paragraphe précédent, sont des noeuds terminaux dans cette hiérarchie. Des liens verticaux entre individus, spécifiés dans les attributs "vue-de" et "vue-par", permettent de définir avec un ensemble d'individus des vues différentes d'un même objet. Un même représentant peut être vu de différentes façons à l'aide de spécialisations différentes. Dans la hiérarchie suivante:

```
(Matrice
  sorte-de
    $valeur Universel
  spec
    $valeur Matrice-carree

  nb-lignes
    type UN D NOMBRE
    $valeur
    importance 0.5

  nb-colonnes
    type UN D NOMBRE
    $valeur
    imortance 0.5

  expression
    type LISTE IND Vecteur
    $valeur
    importance 0.3 )

(Matrice-carree
  sorte-de
    $valeur Matrice
  spec
```

```
    $valeur Matrice-Triangulaire

  nb-lignes
    a-verifier (Egalite !nb-lignes !nb-colonnes)

  nb-colonnes
    a-verifier (Egalite !nb-lignes !nb-colonnes)


(Matrice-Triangulaire
  sorte-de
    $valeur Matrice-carree
  spec
    $valeur Matrice-Diagonale

  expression
    a-verifier (Moitie-vide  !expression)

(Matrice-Diagonale
  sorte-de
    $valeur Matrice-Triangulaire

  expression
    a-verifier (Etre-diagonale  !expression)
```

Soit M une matrice diagonale. L'individu M, dont le système ne connaît pas a priori la spécialisation est d'abord rattaché au concept "Matrice" le plus général. Spécialiser l'individu M en le rattachant à "Matrice-carree" amène à générer un nouvel individu M' lié au premier par l'attribut "vue-de". M' est une "nouvelle vue" du même représentant M. M' peut aussi être spécialisé, pour donner M", nouvelle vue de M en tant que représentant de "Matrice-Triangulaire". Dans la hiérarchie initiale, les individus sont représentés et liés comme suit:



*-A côté des relations verticales, des liens horizontaux sont aussi utilisés, sous formes de références imbriquées, qui sont nombreuses et dues au typage des attributs. Dans la facette "type", les mots clés UN et LISTE permettent de spécifier respectivement des liaisons (1 : 1) et (1 : n).

Dans le schéma "Systeme-diff" définit par :

```
(Systeme-diff
  sorte-de
    $valeur  Universel
  spec
    $valeur  Systeme-diff-lin
             Systeme-diff-plan
  expression
    type LISTE IND Equation-diff
    $valeur
    importance 0.5
  var-etat
    type LISTE IND Var-etat
```

```
      $valeur
      importance 0.5
    parametres
      type LISTE IND Parametre
      $valeur
      importance 0.5
    dimension
      type UN D NOMBRE
      $valeur
      importance 0.5
      a-verifier (Egalite !dimension
                         (compter-nb-eq !expression))

    jacobien
      type UN IND Jacobien
      $valeur
      importance 0.5

    solution
      type UN IND Integration
      $valeur  (Integration (syst-differentiel $valeur SELF))
      importance 0.4  )
```

Le mot clé SELF fait référence au schéma courant lui-même.

L'attribut "jacobien" fait référence par un lien (1 : 1) au schéma:

```
(Jacobien
  sorte-de
    $valeur Matrice
  spec
    $valeur Jacobien-lineaire
  expression
    type UN D Expression
    $valeur
    si-besoin (calcul-formel-jacobien !systeme)
    importance 0.5

  val-point
    type  LISTE D NOMBRE
    $valeur
```

```
    importance 0.5

  valeur
    type UN D NOMBRE
    $valeur
    importance 0.5
    si-besoin (Eval-jacobien !expression !val-point)

  systeme
    type UN IND Systeme-diff
    $valeur
    importance 0.5 )
```

Dans le schéma décrivant une équation:

```
(Equation
    sorte-de
      $valeur Universel
    spec
      $valeur Equation-lineaire
              Equation-explicite
    membre-gauche
      type UN IND Variable
      $valeur
      importance 0.5

    membre-droit
      type UN IND Expr-arithm
      $valeur
      importance 0.5

    variables
      type LISTE IND Variable
      $valeur
      importance 0.4 )
```

l'attribut "variables" définit un lien de type (1 : n), puiqu'il pointe sur les "n" objets qui représentent les différentes variables de l'équation.

*-En mathématiques, des liens horizontaux particuliers sont nécessaires, pour associer à un même concept différentes "représentations" obtenues par des

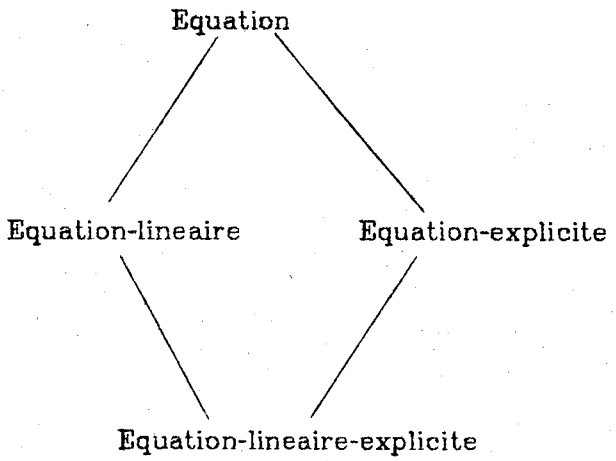
modifications structurelles et de valeurs. Ce lien est décrit dans les concepts avec des attributs particuliers, "representation-de" et "representation-par" dont le réflexe "si-besoin" contient le traitement à utiliser pour construire la nouvelle représentation.

L'organisation générale des schémas, à l'aide de liens verticaux et horizontaux de différentes natures est un graphe.

V/ Héritage et multi-héritage

1/ L'héritage

Un des intérêts de l'organisation des schémas tient à l'utilisation de mécanismes d'héritage par lesquels un schéma recoit totalement ou partiellement la connaissance associée à ses ancêtres. Par exemple, avec différents types d'équations, on peut avoir l'organisation suivante :



L'héritage est partiel si des attributs sont spécialisés. Ce mécanisme facilite la définition de nouveaux schémas en évitant la recopie d'informations redondantes et permet une mise à jour plus sûre et rapide. Les modifications apportées à un schéma sont propagées par héritage, ce qui assure une certaine cohérence de la base d'objets.

2/ Le multi-héritage

La possibilité pour un schéma d'avoir plusieurs ancêtres peut être source de problèmes . Dans l'exemple donné, le schéma "Equation-lineaire-explicite" pose le problème dit de l'héritage multiple. Ce schéma doit hériter de deux schémas "Equation-lineaire" et "Equation-explicite".

Un problème d'héritage est à traiter en cas de conflit entre plusieurs attributs hérités. Nous dirons que deux attributs sont en conflit s'ils ont même type, ou s'ils ont même nom et même type.

Il faut alors établir une priorité d'héritage. De façon classique, l'héritage se fait d'abord de bas en haut puis de gauche à droite (Bobrow 82a/, Goodwin 79, Kuiper 75), comme nous l'avons déjà vu pour LOOPS (ChII, III-2-d). Dans notre exemple, le schéma "Equation-lineaire-explicite" hérite d'abord des schémas "Equation-lineaire" et "Equation" dans cet ordre, puis du schéma "Equation-explicite".

Lors de conflit entre deux attributs, différentes solutions sont possibles. De façon simple, on peut ne considérer que le premier attribut rencontré, dont on hérite en totalité. Le conflit est en fait ignoré.

"L'héritage au niveau des facettes" consiste à faire la fusion des descriptions des deux attributs en conflit, en tenant compte de la priorité d'héritage citée ci-dessus.

Enfin, un "héritage au niveau des valeurs" complète l'héritage précédent en combinant aussi les valeurs des facettes $valeur et val-defaut.

Si on ne considère que les deux attributs suivants, provenant de deux ancêtres d'un même schéma:

```
masse-ressort
   type UN IND MASSE
   $valeur
   val-defaut faible
   si-besoin (Interroger-utilisateur !masse-ressort)
   importance 0.5
```

masse
type UN IND MASSE
$valeur
val-defaut nulle
domaine (nulle faible moyenne)
importance 0.6

Ces deux attributs sont en conflit lors de l'héritage car ils ont même type. Avec la première solution, on hérite de la description du premier. Avec la seconde technique, l'attribut dans le concept généré par héritage est décrit par :

masse-ressort
type UN IND MASSE
$valeur
val-defaut faible
domaine (nulle faible moyenne)
si-besoin (Interroger-utilisateur !masse-ressort)
importance 0.5

Les valeurs des facettes, dans ce cas, sont héritées du premier attribut selon l'ordre de parcours des ancêtres.

La troisième solution fournit la même description structurelle avec pour valeur par défaut une combinaison de "faible" et "nulle".

La première solution, beaucoup trop restrictive, est à rejeter.
Pour conserver l'avantage des descriptions à l'aide de termes flous comme nous le verrons au chapitre suivant, la troisième solution a été choisie.

Les valeurs floues au niveau des valeurs par défaut permettent d'indiquer de façon simple un ensemble de valeurs admissibles, au lieu d'une seule valeur exacte typique. Par exemple, comme nous le verrons au chapitre V, une valeur par défaut égale à "grand" pour un ordre de grandeur permet de spécifier un intervalle de valeurs admissibles plutôt qu'une valeur exacte numérique à utiliser par défaut.

La solution de multi-héritage choisie permet, en fusionnant aussi les valeurs et valeurs par défaut des attributs en conflit, de conserver cet avantage de description dans les concepts générés par héritage. La fusion des valeurs est fondée sur les techniques de combinaisons des valeurs floues exposées au chapitre suivant. Une vérification de cohérence est faite dans le cas de multi-

héritage, afin de s'assurer que l'on n'instancie pas un attribut avec une combinaison de valeurs héritées mais totalement disjointes.

Dans les langages orientés objets cités précédement, comme LOOPS (Bobrow 82a/b/) et Les Flavors (Weinreb 80), un problème important se posait pour le multi-héritage des méthodes. La méthode héritée est une combinaison plus ou moins efficace des méthodes définies sur les ancêtres. Mais le résultat obtenu n'est jamais totalement satisfaisant. Dans notre cas, la dynamique est spécifiée de façon déclarative et répartie sur les attributs. Il n'y a plus de problème d'héritage d'informations globale concernant l'objet dans son ensemble. La solution exposée ci-dessus pour le multi-héritage structurel est donc suffisante pour gérer tous les problèmes d'héritage.

L'héritage multiple accroît encore les possibilités de spécification des connaissances, en permettant par exemple des points de vue multiples d'un même concept. Le concept "Equation-lineaire-explicite" correspond à la combinaison de deux points de vue différents d'une même équation.
Pour améliorer les possibilités d'expression du concepteur, on peut offrir des outils comme ceux de LOOPS, permettant de limiter le parcours en profondeur ou d'imposer un chemin particulier d'héritage.

VI/ Organisation dynamique de la base de schémas

Les schémas sont organisés en graphe. L'organisation reste dynamique. Pour l'implémentation, cela oblige, au contraire de la plupart des langages orientés objets, à ne figer aucune référence. Les liens entre schémas ne sont résolus que lors des interrogations et les valeurs obtenues dynamiquement, par recherches ou applications de traitements, ne sont pas mémorisées définitivement.

Une telle organisation dynamique est coûteuse en temps mais facilite la spécification des connaissances, la réorganisation de la base et reste en accord avec des études psychologiques mettant en évidence l'aspect évolutif de la mémoire et son organisation dynamique (Le Ny 79).

Cet aspect facilite la propagation des informations, puisque aucune référence n'est figée. Il n'y a pas en fait de mise à jour nécessaire par propagation.

On peut aussi restructurer la base, en modifiant la liste des ancêtres des schémas. Des problèmes de cohérence sont alors à prendre en compte, mais ils sont d'autant plus faciles à résoudre qu'aucun lien n'est figé. Nous ne traitons pas ce problème, sujet d'étude surtout dans le domaine des bases de données (Delobel 82, Nicolas 81).

VII/ Différentes vues d'un même objet

*-Les informations sont réparties sur les attributs. Si certaines sont globales et concernent le schéma dans son ensemble, elles sont regroupées dans l'attribut SELF. Ce dernier définit une vue unique "standard" de l'objet.

Mais la structure des schéma est modulaire. Il y a donc autant de façons de voir un "concept" donné qu'il y a de combinaisons possibles de ses attributs. La notion de vue peut être manipulée par l'utilisateur et le système.

Le premier, lors d'une impression d'un objet, peut donner la liste des attributs à utiliser.

Grâce aux coefficients d'importance de chaque attribut, le système sait, lors de l'exploitation, quels sont les attributs qui constituent la vue à utiliser pour chaque concept. Ce sont ceux dont le coefficient propre d'importance est supérieur à une borne fixée auparavant. Par exemple, le schéma "Modèle-physique" peut être vu comme suit, avec une borne d'importance fixée à 0.4:

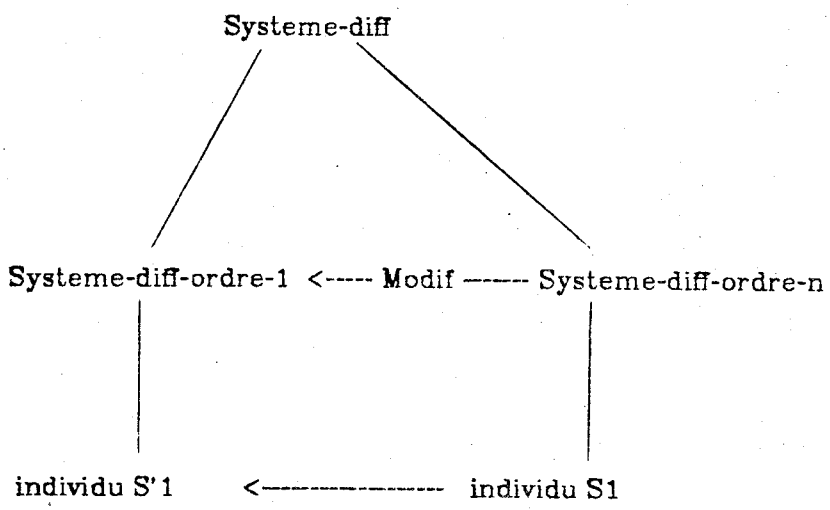
```
(Modèle-physique
   sorte-de
     $valeur Universel
   masse-ressort
     type UN IND MASSE
     $valeur
     val-defaut faible
     importance 1.0

   loi-physique
     type UN IND LOI
     $valeur
     importance 0.5
```

```
force-friction
  type UN D SYMBOLE
  $valeur
  importance 0.4

force-traction
  type UN D SYMBOLE
  $valeur
  importance 0.4 )
```

et ne décrire plus qu'un modèle purement physique, sans aucune référence à l'aspect mathématique. On manipule ainsi des vues différentes d'un même concept.

*-Nous avons vu au paragraphe précédent qu'un ensemble d'instances peut correspondre à un ensemble de vues d'un même individu. Chacune est une spécialisation différente du même représentant. Cette notion de vue est utilisée alors entre individus, c'est à dire entre des éléments terminaux de la hiérarchie.

*-Des liens horizontaux particuliers, définis au paragraphe 4, permettent de spécifier des "représentations " différentes d'une même entité, concept ou individu. Cette notion est nécessaire en mathématique où la classification se fait selon des liens hiérarchiques de généralisation/spécialisation mais aussi selon des liens horizontaux de cette sorte. Par exemple, un modèle différentiel d'ordre "n" peut être ramené à un modèle d'ordre 1, plus important en taille. Chacun est néanmoins une représentation du même système d'équations différentielles.



VIII/ Gestion de cohérence

1/ Les réflexes

Toujours selon le même principe, les éléments pour la gestion de cohérence sont répartis sur les attributs. Le maintien de la cohérence de l'ensemble de la base, c'est à dire des relations entre schémas, est assuré par l'emploi de réflexes, avec les facettes "si-ajout", "si-supp" et "si-modif" qui décrivent les actions à éxecuter lors d'ajout, de suppression et de modification de la valeur de l'attribut. Ces actions sont spécifiées à l'aide d'une procédure. La facette "a-verifier" permet d'introduire des contraintes d'intégrité sous la forme de "schémas prédicats", ou avec des textes de prédicats directement évaluables. Ces contraintes doivent être vérifiées lors de l'instanciation de l'attribut, mais servent également au cours de l'identification par filtrage. Le schéma suivant décrit un système différentiel en utilisant les facettes citées pour la gestion de cohérence.

```
(Systeme-diff
   sorte-de
      $valeur  Universel
   spec
      $valeur  Systeme-diff-lin
               Systeme-diff-plan
   expression
      type LISTE IND Equation-diff
      $valeur
      importance 0.5
      si-modif (affect !dimension  (compter !expression))

   var-etat
      type LISTE IND Var-etat
      $valeur
      importance 0.5
      a-verifier (egalite !dimension (compter !var-etat))

   parametres
      type LISTE IND Parametre
      $valeur
      importance 0.5
```

```
dimension
   type UN D NOMBRE
   $valeur
   importance 0.5
   a-verifier (Schema-Egalite
            (att1 $valeur !dimension)
            (att2 $valeur ?x
               si-besoin (Schema-Compter
                     (att1 $valeur!!expression)
                     (resultat $valeur ?x))))

jacobien
   type UN IND Jacobien
   $valeur
   importance 0.5
   si-besoin (Jacobien (systeme $valeur !expression))

solution
   type UN IND Solution-diff
   $valeur
   importance 0.4  )
```

Plusieurs "!" permettent de résoudre des références imbriquées. Ainsi, "!!expression" et "!expression" font référence au même attribut "expression" du système différentiel mais à deux niveaux d'imbri c ation différents.

Lors d'ajout ou de suppressions d'équations, la dimension du système, égale au nombre d'équations, doit être mise à jour. Cette propagation est indiquée dans la facette "si-modif" de l'attribut "expression" sous forme procédurale. La vérification équivalente est donnée dans l'attribut "dimension" sous forme déclarative avec un "schéma prédicat" pour l'égalité et un "schéma traitement" pour compter le nombre d'équations dans le texte du système. Mais, ces traitements et prédicats ne nécessitent pas d'explications ou détails. Il est donc préférable de les spécifier directement sous forme procédurale.

2/ Le mécanisme d'affectation

Lors de l'affectation d'un attribut, le système exécute une suite de tests.

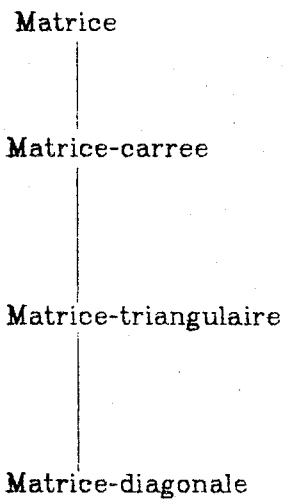
Le premier est basé sur la sémantique de l'attribut donné par son type, dans la facette "type" dont la valeur un schéma.

Si la valeur fournie pour l'instanciation est un concept, il doit être hiérarchiquement infèrieur au schéma qui est donné comme type de l'attribut instancié.

Si la valeur fournie est un individu, le même test est pratiqué. S'il est négatif, avant de signaler une incohérence, le système tente de spécialiser l'individu donné comme valeur jusqu'à un niveau satisfaisant pour maintenir la cohérence. Le schéma ci-dessous représente un traitement d'inversion de matrices.

```
(Inversion-matrice
   sorte-de
     $valeur Inversion

   expression
     type UN IND Matrice-carree
     $valeur
     importance 0.5

   inverse
     type UN IND Matrice-carree
     $valeur
     importance 0.5
     si-besoin (Inversion1 !expression) )
```

Si la valeur donnée pour instancier l'attribut "expression" est un individu de type Matrice, le système essaie de le spécialiser, c'est à dire de le faire descendre dans la hiérarchie suivante, au moins jusqu'au concept "Matrice-carree", nécessaire pour permettre l'inversion.



Aprés ce contrôle, le système vérifie que la valeur respecte les contraintes spécifiées, s'il y en a, dans les facettes "Domaine" et "a-verifier".

Cette gestion de cohérence est essentielle pour permettre une utilisation correcte des concepts. Les possibilités d'expression sont assez souples et importantes pour permettre de donner des règles de cohérence différentes, variant d'un objet à l'autre et même d'un utilisateur à l'autre.

Mais la répartition des informations sur chaque attribut apporte souvent de la redondance. Des contraintes portant sur plusieurs attributs doivent être dupliquées dans chacun d'eux. Un exemple en est donné avec le schéma Matrice-carree, dans les attributs "nb-lignes" et "nb-colonnes":

```
(Matrice-carre
  sorte-de
    $valeur Matrice
  spec
    $valeur Matrice-triangulaire

  nb-lignes
    type UN D NOMBRE
    $valeur
    importance 0.5
    a-verifier (egalite !nb-lignes !nb-colonnes)

  nb-colonnes
    type UN D NOMBRE
```

$valeur
imortance 0.5
a-verifier (egalite !nb-lignes !nb-colonnes)

expression
type LISTE IND Vecteur
$valeur
importance 0.3)

Une même information peut être utilisée comme contrainte et comme méthode d'inférence. Elle est alors dupliquée dans les réflexes "a-verifier" et "si-besoin". Le problème de la cohérence est important, et reste à résoudre sur beaucoup de points (Doyle 79). Il est à traiter en se rapprochant des études faites en bases de données (Delobel 82).

IX/ Conclusion

Cette représentation est en constant développement. Des améliorations par rapport à la description donnée plus haut ont déjà été précisées dans (Rechenmann 85). Les principaux apports de cet outil peuvent être résumés ainsi:

-une représentation déclarative et uniforme qui permet une organisation efficace des connaissances.

-aide à la gestion de cohérence de la base d'objets en particulier en utilisant une notion de type pour chaque attribut.

-simplification et unification des outils d'expression de la dynamique. Elle est distribuée sur les attributs sous forme déclarative. Certains attributs apportent la description des traitements utilisables sur le schéma manipulé. Les traitements, pour les plus importants, sont aussi décrits sous forme de schémas.

-diversité des mécanismes d'inférences spécifiés à l'aide d'une représentation déclarative. Il suffit de donner à l'aide d'un filtre la description du résultat voulu. Le système opère les recherches et active les traitements nécessaires pour l'obtenir.

Par ailleurs, il est intéressant de se situer par rapport à des études de même

nature qui visent à spécifier des systèmes informatiques capables d'apprentissage. Notre objectif ici n'est pas de spécifier un tel système. Nous mettons simplement en évidence des caractéristiques essentielles de la représentation des connaissances présentée dans les paragraphes précédents, qui sont utiles et nécessaires pour implémenter un tel système.

La notion même de schéma a été apportée en premier lieu par des études de psychologie (Winston 75). Ces études ont de même montré la nécessité de l'organisation des connaissances, traduite ici comme nous l'avons vu sous forme d'un graphe de schémas. Une base dynamique est aussi nécessaire pour permettre la réorganisation de l'ensemble suite à l'acquisition de nouvelles connaissances. Cet aspect évolutif de la mémoire est reconnu nécessaire, efficace et pratiqué chez l'être humain (Le Ny 79).

La base de schémas proposée, avec les notions d'organisation, de dynamique, de multi-héritage et son mécanisme d'exploitation permet la génération de concepts, point important pour permettre l'apprentissage. Il s'agit de créer, selon différentes techniques et raisonnements, de nouveaux objets à partir de ceux qui existent (Nagel 83).

Nous présentons dans le chapitre suivant le mécanisme d'exploitation qui avec la manipulation de valeurs floues et de la sémantique des objets permet la génération de concepts.

CHAPITRE 4

LE MECANISME D'EXPLOITATION A BASE DE FILTRAGE FLOU

Chapitre IV/ Le mécanisme d'exploitation à base de filtrage flou

PLAN

I/ Définition de la dynamique dans une base de schémas
- 1/ Recherches de schémas valeurs
- 2/ Recherches et activations de schémas traitements

II/ Les stratégies de filtrage

III/ Le filtrage
- 1/ Le filtrage classique
- 2/ Le filtrage flou
- 3/ Le filtrage réalisé

IV/ Présentation de la théorie des ensembles flous
- 1/ Les ensembles flous
- 2/ Opérations sur les ensembles flous
- 3/ Mesures de possibilités et de nécessités
- 4/ Manipulations de ces notions
- 5/ Utilisation des ensembles flous

V/ Spécification du flou et organisation de l'univers de discours
- 1/ L'univers de discours
- 2/ Les valeurs floues
- 3/ Les univers locaux de discours
- 4/ Autres définitions possibles

VI/ Manipulations des valeurs terminales
- 1/ Evaluation de distances
- 2/ Combinaison de valeurs

VII/ Le processus général de filtrage réalisé
- 1/ Processus élémentaire
- 2/ Utilisation des réflexes
- 3/ Manipulation des variables
- 4/ Traitement des référemces imbriquées

VIII/Le filtrage structurel

I/ Définition de la dynamique dans une base de schémas

Comme le montrent les travaux de F Rechenmann (Rechenmann 85), l'absence d'utilisation, dans la représentation proposée, d'autres représentations, telles que les règles de production et leurs mécanismes d'exploitation, offre une unité du formalisme, une simplicité mais une multiplicité des mécanismes d'inférence constituant la dynamique de la base.

On peut distinguer sept manipulations élémentaires qui définissent l'essentiel de la dynamique.

Au niveau des schémas, une sélection rend l'ensemble des entités vérifiant certaines caractéristiques spécifiées sous forme d'un filtre.
Une classification permet de situer et d'insérer un schéma dans la hiérarchie des objets constituant la base.
Le mécanisme d'instanciation consiste à créer un individu à partir d'un concept choisi, en affectant une valeur aux attributs voulus.

Au niveau des attributs, la dynamique concerne plus précisément l'obtention d'une valeur. Les diverses infèrences se font uniquement à partir de la description du résultat voulu sous forme de filtre.

Il y a inférence par héritage si la valeur est prélevée dans la description d'un des ancêtres de l'entité concernée.
Lors d'une inférence par filtrage, le système recherche un schéma particulier qui est la valeur voulue ou qui est identifiable au schéma traité et permet de le compléter.
L'inférence peut aussi être pratiquée à l'aide de traitements. Ces derniers sont décrits par des schémas, comme nous l'avons vu au chapitre précédent, ou avec des fonctions directement évaluables. La valeur obtenue résulte souvent de l'exécution d'algorithmes externes.
Enfin, en dernier lieu, la valeur peut être obtenue à partir des informations apportées dans la facette "val-defaut", si les recherches précédentes ont été vaines.

L'essentiel de la dynamique concerne donc l'obtention d'une valeur pour un attribut, avec s'il y a lieu des effets de bord. Elle est spécifiée de façon déclarative avec des filtres. Ceux-ci décrivent les résultats voulus et les traitements nécessaires. Le mécanisme d'exploitation effectue les recherches et élabore la dynamique voulue.

Si le type de l'attribut traité contient le mot clé LISTE, tous les schémas trouvés admissibles sont rendus, sinon une fonction de choix permet de sélectionner une entité parmi les schémas candidats. De façon simple, on choisit souvent l'entité la plus proche du filtre utilisé.

Des manipulations plus complexes, comme des analogies, peuvent aussi être construites avec ces mécanismes élémentaires comme nous le verrons dans les derniers paragraphes de ce chapitre.
Les sélections, classifications, instanciations et inférences utilisent à la base un mécanisme de recherche fondé sur le filtrage flou, sujet de ce chapitre. Nous définissons deux types de recherches.

1/ Recherches de schémas valeurs

Dans ce cas, un filtre plus ou moins complexe donné en valeur à la facette "$valeur" désigne le schéma adéquat. Pour l'obtenir il suffit d'une recherche sans activations de traitements.

Par exemple, dans le schéma décrivant un système physique, la recherche du système mathématique associé se fait à partir des composants requis . L'attribut "modèle-mathématique" désigne simplement le modèle associé au modèle physique courant.

```
....
modèle-mathématique
  type UN IND Modèle-math
  $valeur (Modèle-math  (modèle-physique
                        $valeur SELF))
  importance 0.5
....
```

Avec une définition très simple d'un Modèle-mathématique :

```
(Modèle-math
   sorte-de
      $valeur Universel
   texte-modèle
      type UN IND SYMBOLE
      $valeur
      importance 0.5
```

modèle-physique
type UN IND Modèle-physique
$valeur
importance 0.5)

Ce cas correspond à une inférence directe à l'aide de règles de production.

Des filtres imbriqués peuvent aussi être utilisés pour spécifier ce qui correspond à une règle d'infèrence ayant des conditions dépendantes (Rechenmann 85).

Lors d'une infèrence par filtrage, on peut aussi s'intéresser non pas au schéma rendu par l'opération de recherche, mais à la valeur de l'un de ses attributs. Cette valeur est obtenue grâce à une variable intermédiaire placée dans la facette "$valeur" de l'attribut pour lequel on effectue les recherches. Dans le filtre, placé dans la facette "si-besoin", la variable intermédiaire est donnée comme valeur à l'attribut adéquat. Un exemple d'une telle spécification est donné au paragraphe 8-1 du chapitre précédent, dans le schéma "Systeme-diff" avec le "schéma-compter".

Dans tous les cas, on ne fait que spécifier à l'aide de filtres les caractéristiques essentielles des valeurs cherchées et le système effectue les recherches et élabore la dynamique nécessaire pour trouver ou construire le résultat voulu.

2/ Recherches et activations de schémas traitements

L'obtention des valeurs nécessite alors l'utilisation d'un ou plusieurs schémas traitement spécifiés dans la facette "$valeur" ou dans le réflexe "si-besoin" de l'attribut concerné. Des effets de bord peuvent ainsi être pratiqués.

Le système commence par rechercher, comme nous le verrons au paragraphe suivant, le schéma traitement adéquat si celui-ci n'est pas donné de façon précise dans le filtre manipulé.

Puis, le système recherche, parmi les instances de ce schéma traitement un individu qui représente une précédente utilisation du traitement sur des objets identiques aux paramètres actuels.

Si la recherche est un succès, l'instance trouvée est la valeur voulue.

Sinon, le système génère une instance du schéma traitement avec en paramètres les objets actuels nécessaires. A la création de ce nouvel individu, les réflexes "si-besoin" attachés aux attributs du schéma traitement utilisé activent les programmes voulus qui fournissent les valeurs nécessaires pour instancier l'individu.

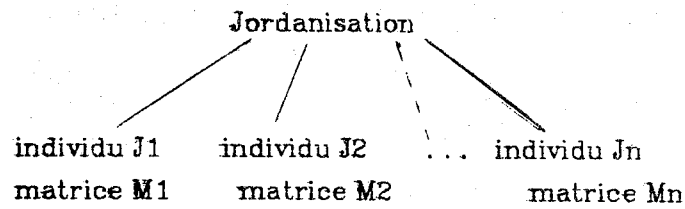
Avec le traitement de "Jordanisation" définit ci-dessous:

```
(Jordanisation
    sorte-de
      $valeur Traitement
    matrice
      type UN IND Matrice-carree
      $valeur
      importance 0.5
    val-prop
      type UN IND Vecteur
      $valeur ?valprop
      importance 0.5
      si-besoin  (jordan !matrice ?valprop
                          ?vecprop ?succes))
    vec-propre
      type UN IND Matrice-carre
      $valeur ?vecprop
      importance 0.5
    succes
      type UN D Booleen
      $valeur ?succes
      importance 0.5   )
```

pour avoir la matrice de Jordan associée à une matrice M décrite par le schéma M-X, il suffit de faire appel à un traitement "Jordanisation" avec le filtre suivant:

```
(Jordanisation
    (matrice type UN IND Matrice-carree
         $valeur M-X ))
```

Ce filtre désigne précisément le schéma traitement à utiliser. Si la hiérarchie des représentants déjà créés à cet instant du traitement de Jordanisation est:



le système conclu que le traitement n'a jamais été utilisé auparavant sur la matrice M-X. Il génère alors une instance du concept "Jordanisation" avec en entrée le schéma M-X pour valeur de l'attribut "matrice". Le réflexe "si-besoin" activé à la génération de l'individu permet d'instancier les attributs "val-prop", "vec-propre" et "succes" par propagation des instanciations des variables résultats ?valprop, ?vecprop et ?succes du programme activé.

Une telle expression de la dynamique permet de limiter l'espace de recherche et d'éviter le problème d'explosion combinatoire commun par exemple avec les systèmes de production.

Dans des langages orientés objets comme KOOL (Albert 83), LOOPS (Bobrow 82,a/ b/) et MERING (Ferber 84), utilisant les notions de frames et de règles de production, les réflexes peuvent avoir en valeur une liste de règles. Dans notre cas, une expression équivalente est obtenue en donnant une liste de filtres.

Les schémas prédicats sont manipulés selon une méthode identique. Un individu d'un schéma prédicat n'est créé que si le test qu'il représente est vérifié pour les paramètres passés dans le filtre. Lors de l'évaluation de prédicats, c'est donc l'obtention d'un individu qui signifie que le prédicat est vérifié pour les objets manipulés.

Ce mode de spécification de la dynamique offre une puissance d'expression équivalente aux systèmes habituels, mais de façon plus déclarative, modulaire et modifiable et en limitant l'espace de recherche.

Deux stratégies de filtrage permettent d'effectuer ces recherches.

II/ Les stratégies de filtrage

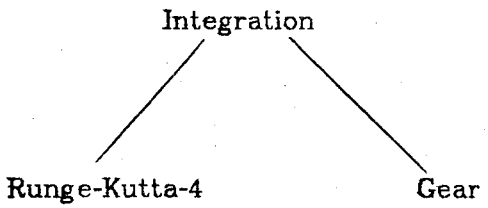
Les inférences sont formulées à l'aide de filtres, c'est à dire de schémas incomplets. Le système effectue des recherches à partir de ces informations.
Deux stratégies de filtrage sont utilisées pour effectuer les recherches et ainsi permettre les manipulations citées au paragraphe précédent.

La première stratégie est un processus de "filtrage à-plat". Ce processus compare un schéma filtre avec une liste de schémas provenant souvent d'un même niveau dans la hiérarchie des objets. Il rend pour chaque schéma testé une évalution de distance avec le filtre, variant entre 0 et 1.

La deuxième stratégie est un processus de "filtrage en profondeur", combinaison de filtrages à-plat. Une opération de filtrage à-plat au niveau "n" permet de sélectionner un schéma identifiable au filtre tel qu'il est instancié à ce niveau. On choisit le schéma le plus proche du filtre selon les distances rendues par le processus de filtrage à ce niveau.
Des informations apportées par ce schéma, et non utilisées pour la mise en correspondance, complétent le filtre. Les valeurs des attributs dans le filtre, mais aussi la structure de ce dernier c'est à dire la liste des attributs qu'il contient, peuvent évoluer. De façon équivalente, mais en utilisant une représentation supplémentaire sous forme de règles de production avec leur mécanisme d'exploitation, le filtre peut être complété en activant un ensemble de règles d'infèrence associées au schéma retenu au niveau "n". Il faut dans ce but spécifier, pour chaque schéma de la base, un ensemble de règles d'infèrence qui permettent de compléter le filtre en fonction des informations qu'il contient à l'instant du traitement (Granger 84b/).

Le processus est itéré au niveau "n+1" avec le filtre complété et les schémas fils de l'entité retenue au niveau "n". L'évolution s'arrête lorsque le système a trouvé un schéma estimé suffisament proche du filtre, ou lorsque il n'y a plus de schémas utilisables.

Les schémas d'intégration définit au chapitre précédent constituent une hiérarchie de schémas trés simple :

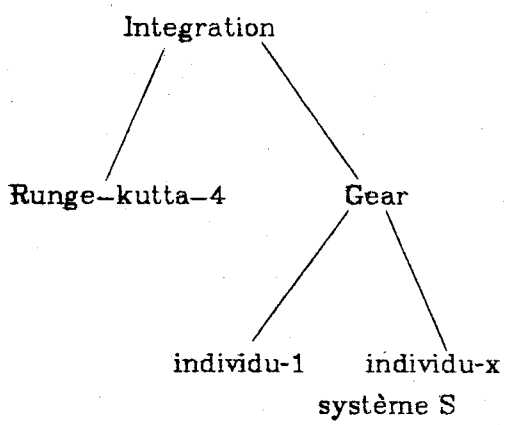


On peut laisser le choix de la méthode à utiliser à la charge du système en activant l'intégration du système différentiel S avec le filtre :

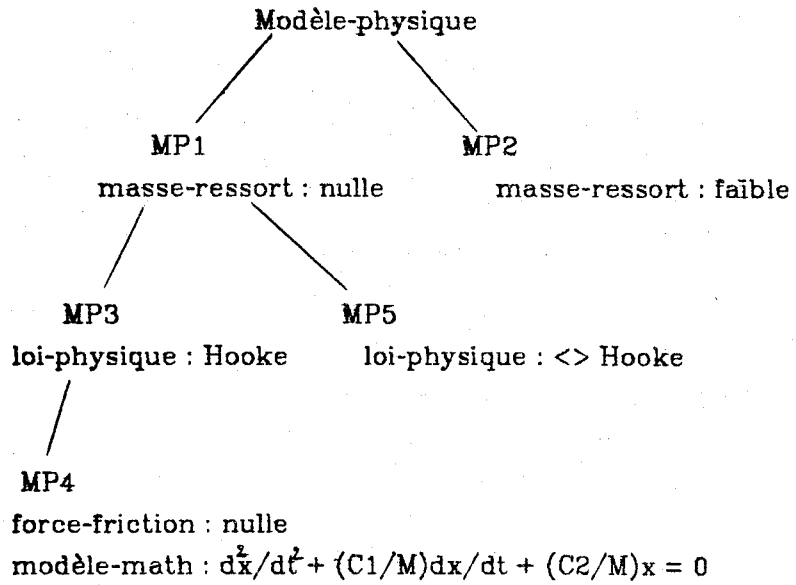
(Integration
(syst-differentiel $valeur S))

Le mécanisme d'exploitation recherche avec un filtrage en profondeur la méthode adéquate parmi les schémas Runge-Kutta-4 et Gear, puis l'active comme nous l'avons vu au paragraphe précédent.

Si le système différentiel traité S est raide, un nouvel individu du traitement Gear est créé:



Avec le schéma "Modèle-physique" donné précédemment, une hiérarchie obtenue par spécialisation est décrite par:



On ne précise dans chaque concept que la variation de l'attribut qui permet la spécialisation. Si le filtre initial est:

((masse-ressort nulle)(loi-physique Hooke)(force-friction faible))

et en supposant la correspondance possible entre les termes "nulle" et "faible" au sujet de la force de friction, le filtrage en profondeur associe MP4 au filtre et fournit le modèle mathématique voulu :

$$d\dot{x}/dt^2 + (C1/M)dx/dt + (C2/M)x = 0.$$

Deux étapes de filtrage à plat ont permis des identifications successives avec MP1 et MP3 pour aboutir à MP4.
Les distances obtenues successivement avec les différents schémas sont :

MP1	0.87
MP2	0.83
MP3	0.94
MP5	0.0
MP4	0.9

Ce processus est construit comme un parcours d'arbre, avec retour-arrière automatique en cas d'échec. Il y a échec si aucun schéma n'est estimé suffisament proche du filtre au dernier niveau d'une sous-arborescence de la hiérarchie des schémas.

L'utilisateur n'a ainsi que peu d'informations à fournir et il n'a pas ou peu à répondre à des interrogations du système.

III/ Le filtrage

1/ Le filtrage classique

Cette technique classique en Intelligence Artificielle permet d'extraire d'une base l'ensemble des entités ou faits correspondant à un filtre (Charniack 83) (Waterman 78 b/) en manipulant des informations qualitatives et quantitatives. Le filtre et le fait sont des schémas. La notion de correspondance est à définir de façon plus ou moins précise (Cohen 77). Les procédés ordinaires de filtrage sont fondés sur une conception rigide de la similarité entre descripteurs du filtre et du fait. La similarité globale est calculée comme une combinaison logique des similarités atomiques binaires.

Aucune imprécision ou incomplétude n'est permise dans l'expression des descripteurs. Un langage restreint et strict est imposé pour la spécification des connaissances.

Les processus d'identification n'utilisent pas ou peu la sémantique des objets. Ils se contentent d'une association entre les objets basée sur la correspondance exacte et syntaxique du nom des attributs. On n'utilise alors aucune référence au contenu des attributs, c'est à dire à leur signification propre ou sémantique.

Les processus rendent des distances binaires (0 ou 1) et ne permettent pas au système de tenir un raisonnement nuancé. Des solutions admissibles sont ainsi rejettées, faute de nuances, ce qui peut être la cause d'un échec global.

2/ Le filtrage flou

Un processus de filtrage flou (Cayrol 82, Umano 79, Giles 80) sait manipuler une valeur imprécise pour un attribut et rend une note entre 0 et 1 qui est une évaluation de distance entre le filtre et le fait. Ces techniques ont été initialement utilisées pour le traitement du langage naturel (Winston 75). L'utilisation de valeurs floues permet l'expression d'incertitudes et d'incomplétudes (Dubois 80, Prade 82, Zadeh 83). L'expression des connaissances et donc les communications homme-machine en sont facilitées. Par exemple, en réponse à une interrogation du système sur la taille d'une personne, l'utilisateur avec un système habituel doit rendre une valeur précise, telle que 1.75 ou 1.80. Avec le langage proposé, il lui suffit de fournir une valeur floue, comme "grand", pour indiquer au système un ensemble de valeurs "plus ou moins" admissibles. Ces facilités de description sont nécessaires en particulier dans une représentation orientée objets pour les valeurs par défaut des attributs. Celles-ci (Yager 84) désignent par défaut un ensemble de valeurs possibles, c'est à dire un concept, plutôt qu'une valeur exacte typique. Une telle description est plus robuste et moins restrictive, et correspond exactement à l'utilisation des valeurs floues.

3/ Le filtrage réalisé

Le premier objectif était de réaliser un mécanisme qui permette d'exploiter correctement la représentation centrée objets proposée. Cet objectif est atteind simplement avec un filtrage élémentaire ordinaire et les deux stratégies de filtrage citées dans le paragraphe précédent. Ces techniques permettent d'utiliser les mécanismes d'inférence présentés dans le premier paragraphe. Le

filtrage sait manipuler les schémas traitements comme ils sont définis plus haut. Dans un premier temps, le filtrage réalisé permet donc d'effectuer les recherches et de construire les schémas nécessaires pour obtenir les résultats voulus.

Dans un deuxième temps, afin de faciliter dans certains domaines la description des schémas et pour spécifier un mécanisme d'exploitation plus nuancé et puissant, nous avons réalisé un processus de filtrage flou. Ce dernier permet de manipuler à la fois des valeurs exactes et des valeurs floues définies de différentes façons.

Il est l'élément principal de la dynamique de la base de schémas et s'intègre parfaitement dans cette représentation. La hiérarchie des schémas constitue un univers de discours dans lequel sont définies les valeurs floues utilisées. Une partie de notre étude a porté sur les manipulations nécessaires des valeurs floues lors du multi-héritage. La combinaison de ces valeurs permet de conserver l'avantage d'une telle spécification pour les concepts générés par héritage.

L'utilisation de termes imprécis du langage naturel sous forme de valeurs floues facilite la spécification des connaissances. Cette possibilité est importante car bien souvent nos connaissances ne sont pas suffisantes pour exprimer une information de façon aussi précise et exacte que l'exigent les systèmes informatiques.

Les termes flous désignent des ensembles de valeurs admissibles plutôt qu'une valeur exacte. L'association des schémas ne se fait donc plus à partir de correspondances exactes de valeurs, mais avec des notions d'inclusions d'ensembles.

Pour utiliser au mieux la sémantique des objets, le système fonde l'association entre attributs essentiellement sur leur type, c'est à dire la description de leur contenu et donc leur sémantique. Avec le concept de modèle physique, l'association entre les attributs décrivant la masse du ressort n'est pas fondée sur le nom des attributs qui peut être "masse", "masse-ressort" ou "poids" mais sur la désignation de leur contenu, c'est à dire le type qui doit être MASSE.

Le raisonnement tenu est beaucoup plus souple et nuancé. Les réponses rendues au sujet des correspondances entre schémas sont moins strictes et correspondent souvent à une approximation sous forme d'une distance variant entre 0 et 1.

Le processus général mis au point est constitué de deux étapes principales : le filtrage de structure et le filtrage atomique. La première n'identifie que les structures des objets alors que la seconde utilise les valeurs terminales atomiques. Des variantes dans le raisonnement sont possibles en utilisant les informations de façon plus ou moins poussée.

Divers aspects d'un même objet peuvent être utilisés en limitant la structure utile des schémas.

Comme nous le verrons au paragraphe suivant, la définition des termes flous est simple mais assez souple pour permettre plusieurs spécifications différentes.

Enfin, le processus permet la manipulation simultanément des notions d'incomplétude et d'imprécis avec les valeurs floues, d'importance des attributs dans la description et de certitude au sujet de certaines valeurs fournies. Pour cela des facettes "importance" et "certitude" peuvent être associées à chaque attribut, comme dans le schéma "modèle-physique" donné au chapitre précédent.

Le problème nouveau est de définir des termes flous que le système sache manipuler. A cette fin, nous utilisons la théorie des ensembles flous (Prade 82, 83, 84, Zadeh 79, 83) et en particulier la notion de distributions de possibilités.

IV/ Présentation de la théorie des ensembles flous

Dans les études que l'être humain mène sur son mode de raisonnement, on a longtemps entretenu une confusion entre "raisonnement usuel" et "logique" (Kayser 79). Pourtant de nombreux exemples montrent à l'évidence qu'il nous est naturel et familier de raisonner en termes de mesures. C'est un aspect que le formalisme logique tend à réduire sinon à nier (Israel 83). Les modèles logiques, en général, donnent une représentation idéalisée de nos connaissances, ce qui les rend parfois inaptes à modéliser certains de nos raisonnements. Nous faisons des évaluations, pour faire évoluer notre opinion en fonction des informations recues, qui ne s'effectuent pas sur une échelle à deux valeurs: vrai ou faux. Une échelle plus fine est nécessaire pour prendre en compte des imprécisions et incertitudes, qui sont des informations spécifiées à l'aide de termes flous. Plusieurs solutions ont été proposées: les fonctions de certitude (Shortliffe 75), les plausibilités (Kayser 79) et la logique floue (Zadeh 79). Nous présentons ici

simplement les outils extraits de cette dernière théorie (Dubois 80), nécessaires à notre travail.

1/ Les ensembles flous

La nécessité de disposer d'une échelle plus fine qu'une échelle à deux valeurs a amené Zadeh à introduire et à définir les ensembles flous. Un ensemble flou A dans un référentiel U (univers de discours) est caractérisé par sa fonction d'appartenance μ_A qui indique si un élément e de U appartient plus ou moins à A: μ_A : U -->[0 1]. μ_A(e) représente le degré d'appartenance de e à A. Par exemple, si X et Y désignent deux masses, on peut écrire :

$$\mu_{masse}(X) = 0.9 \quad et \quad \mu_{masse}(Y) = 0.2$$

Le concept "lourd" est une notion floue. La transition entre l'appartenance et la non appartenance ne se fait pas de manière brutale, mais graduellement: on définit ainsi un sous ensemble flou.

2/ Opérations sur les sous-ensembles flous

Les opérations entre les sous-ensembles s'expriment par des opérations sur leur fonction d'appartenance (Dubois 80). Si A et B sont des sous-ensembles flous du référentiel U et μ_A, μ_B leurs fonctions d'appartenance, on définit:

-l'inclusion : $A \subset B$ si et seulement si

$$\forall e \in U, \mu_A(e) < \mu_B(e)$$

sous forme d'une relation d'ordre

-l'intersection : $A \cap B$ est défini par

$$\forall e \in U, \mu_{A \cap B}(e) = Min(\mu_A(e), \mu_B(e))$$

-l'union : $A \cup B$ est défini par

$$\forall e \in U, \mu_{A \cup B}(e) = Max(\mu_A(e), \mu_B(e))$$

-pseudo-complémentation : A' pseudo-complémentaire de A défini par

$$\forall e \in U, \mu_{A'}(e) = 1 - \mu_A(e)$$

3/ Mesures de possibilité et de nécessité

Le concept de possibilité a aussi été introduit par Zadeh. Une mesure de possibilité μ sur un univers U est un ensemble de fonctions de P(U) dans [0 1] où P(U) est l'ensemble des parties de U avec :

$$\Pi(\phi) = 0 \quad et \quad \Pi(U) = 1$$

$$\forall A \subset P(U) \quad et \forall B \subset P(U) \quad \Pi(A \cup B) = Max(\Pi(A), \Pi(B))$$

Une mesure de possibilité Π est construite à partir d'une distribution de possibilités "p", fonction de U dans [0 1].

$$\forall A \subset P(U) \quad \Pi(A) = Sup_{u \in A}(p(u))$$

On a $\Pi(\{u\}) = p(u)$.
De facon duale, on définit une mesure de nécessité N comme suit:

$$\forall A \subset P(U) \quad N(A) = 1 - \Pi(A') \text{ ou A' = complémentaire de A}$$

Un ensemble de propriétés intéressantes sont liées à ces notions (Dubois 80, 85). On définit une "proposition floue" comme une proposition écrite "X est Y" où X est une variable prenant ses valeurs dans un univers de discours U et Y un sous-ensemble flou de U qui exprime la compatibilité entre les valeurs de U et le concept Y (Fieschi 84). Les distributions de possibilités permettent de représenter et de manipuler ces propositions floues. Chacune, comme "X est Y", induit une distribution de possibilités qui correspond à la fonction d'appartenance μ_Y du sous-ensemble flou Y :

$$\text{Possibilité (X etre-masse 85.0) = p(85) = } \mu_Y(85) = 0.8$$

La mesure de possibilité permet de traduire une certaine similitude entre deux propositions floues.

4/ Manipulations de ces notions

Toutes sortes de manipulations ont été définies avec ces notions. Ainsi, pour permettre des inférences, le Modus Ponens Généralisé et le Modus Tollens Généralisé ont été définis comme des extensions du Modus Ponens et du Modus Tollens de la logique classique (Dubois 80,Prade 82). De même, diverses techniques permettent de combiner ces valeurs. Notre sujet n'est pas ici de détailler ces mécanismes, présentés dans (Dubois 80,82, 85, Prade 82, 84, Zadeh 84).

5/ Utilisations des ensembles flous

Nous n'utilisons de cette théorie que la notion de sous-ensembles flous. Chaque terme flou est défini sous forme d'un tel ensemble, caractérisé par sa fonction d'appartenance. On peut ainsiheuristiquesruire des distributions de possibilités pour des propositions floues. Comme nous verrons par la suite, ces fonctions d'appartenance sont elles-mêmes spécifiées sous forme de fonctions mathématiques très simples. Une grande souplesse de codage, avec des facilités de modifications, en découlent. Les calculs de distance pratiqués n'utilisent que les fonctions d'appartenance, mais permettent la manipulation à la fois de valeurs exactes et floues. Des techniques simples et ad hoc de combinaisons de valeurs ont été définies pour permettre la manipulation simultanément des notions d'imprécis, d'incertain et d'importance des attributs dans les descriptions des schémas.

V/ Spécification du flou et organisation de l'univers de discours

1/ L'univers de discours

L'utilisation d'un type pour chaque attribut est essentielle pour la représentation des connaissances et pour le processus de filtrage. Par définition, toute valeur d'un attribut est un schéma ou un filtre désignant un schéma qui se situe dans une famille d'entités désignée par le type de l'attribut. S'il n'y est pas, il doit pouvoir y être ramené, sinon il y a erreur.

Chaque valeur définit un lien horizontal. Le type permet de désigner des "valeurs terminales", qui ne sont pas des références imbriquées au véritable sens du terme. On utilise pour cela, dans le type des attributs, les mots clés IND pour "indirection" et D pour "direct".

Pour les valeurs terminales, le type de l'attribut avec le mot clé "D", désigne un schéma qui contient la définition des valeurs floues utilisables au sujet du domaine local concerné. Ce schéma est un "univers local " de discours, dans lequel des termes flous sont définis relativement les uns par rapport aux autres.

La hiérarchie des schémas constitue un univers de discours complet sous forme d'une hiérarchie d'univers locaux. Le typage des attributs permet donc aussi de spécifier le vocabulaire disponible dans chaque domaine.

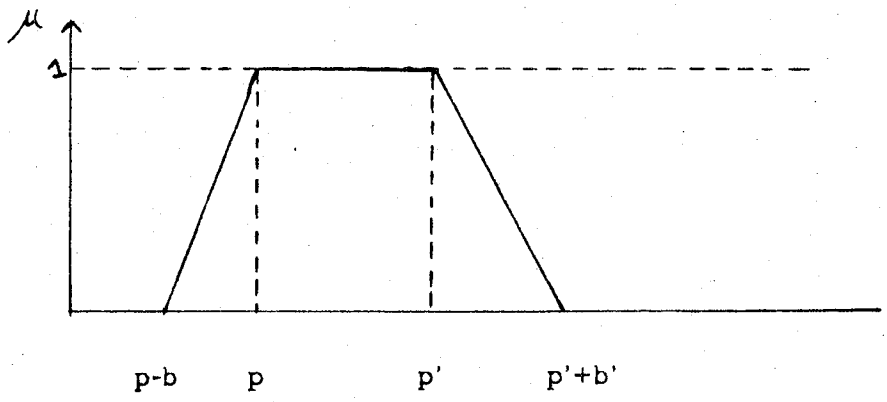
2/ Les valeurs floues

Les valeurs terminales sont définies numériquement sur une échelle spécifique. Par exemple, dans l'univers des MASSE de volumes, les valeurs "lourd", "moyen", "plume" sont définies relativement les unes par rapport aux autres sur une échelle allant de 0 à 140 kilogrammes.

Parmi les univers locaux, on distingue les repèrables et les non repèrables. Dans un univers repèrable, une erreur ε est donnée pour permettre un filtrage flou avec des nombres. Les valeurs numériques x et y sont identifiables si $x \in [y-\varepsilon, y+\varepsilon]$. Dans les univers du second type, les valeurs ne peuvent pas être des nombres et l'échelle locale est choisie arbitrairement. Par exemple, les univers locaux GRANDEUR et MASSE sont repèrables, au contraire des univers QUALITE et COULEUR.

Les valeurs terminales sont exactes ou floues. Une valeur terminale exacte est un nombre dans un univers repèrable. Dans un univers non repèrable, c'est un terme défini par un nombre choisi dans l'échelle locale. Une valeur terminale floue est définie par une distribution de possibilités (Dubois 80), elle-même spécifiée à l'aide d'une fonction trapézoidale (Cayrol 82).

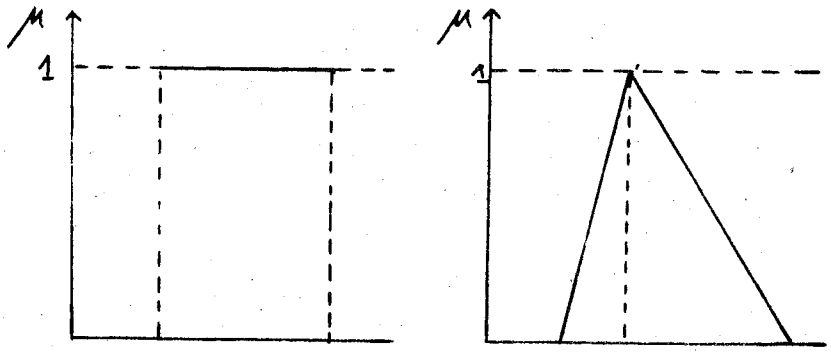
De façon générale, une distribution est donnée sous la forme: μ(p p' b b')



Une fonction trapézoidale μ(p p' b b') définit la distribution:

$$\mu(x) = \begin{cases} 1 \ si \ x \quad [p \ p'] \\ 0 \ si \ x \leqq p - b \ \ ou \ \ x \geqq p' + b' \\ \frac{1}{b} * x + (1 - \frac{p}{b}) \ si \ p - b \leqq x \leqq p \\ -\frac{1}{b'} * x + (1 + \frac{p'}{b'}) \ si \ p' \leqq x \leqq p' + b' \end{cases}$$

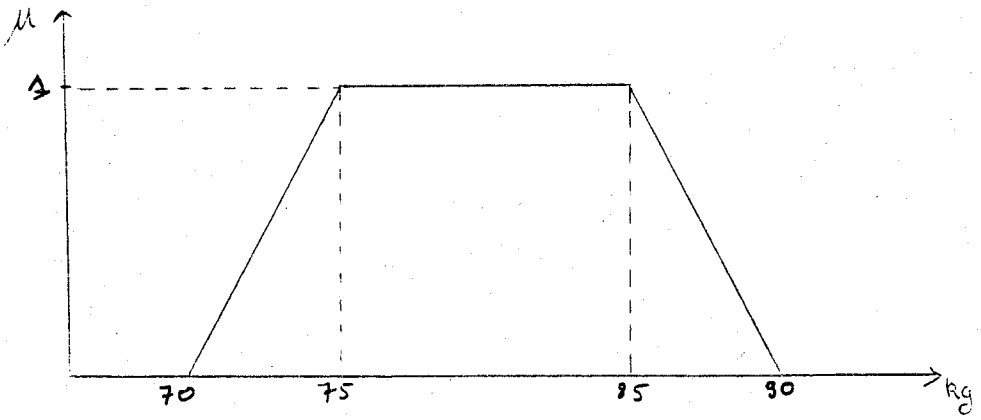
On peut utiliser des distributions particulières, telles que:



(p p' 0 0) (p p b b')

ce qui permet de se rapprocher des notions plus habituelles de codages avec des intervalles de valeurs .

Dans le domaine des MASSE, sur une échelle de 0 à 140, le terme "plutot-lourd" peut être défini avec la fonction : $\mu(75\ 85\ 5\ 5)$



3/ Les univers locaux de discours

Les univers locaux SYMBOLE et NOMBRE permettent respectivement un filtrage exact sur des valeurs ou un filtrage flou avec des nombres. Le premier permet de typer n'importe quelle chaîne de caractères avec laquelle on utilisera un filtrage syntaxique exact. Dans le second cas, la donnée d'une erreur ε est suffisante. Ces informations sont placées dans des attributs particuliers des schémas. Par exemple, dans le modèle physique précédemment utilisé, l'attribut "masse-ressort" fait référence par son type au schéma MASSE qui peut être défini ainsi:

```
(MASSE
   sorte-de
      $valeur Universel
   type-univers
      type UN D SYMBOLE
      $valeur mesurable
      importance 0.5
   borne-inf
      type UN D NOMBRE
      $valeur 0.0
      importance 0.5
   borne-sup
```

```
    type UN D NOMBRE
    $valeur 140.0
    importance 0.5
erreur
    type UN D NOMBRE
    $valeur 4.0
    importance 0.5
l-termes-flous
    type LISTE D SYMBOLE
    $valeur ( (lourd (85 100 5 5))
              (plutot-lourd (75 85 5 5))
              (leger (40 50 5 5)))
    importance 0.5 )
```

Les attributs "borne-inf" et "borne-sup" donnent la définition de l'échelle numérique choisie dans l'univers local concerné, alors que "erreur" indique l'erreur à utiliser dans ce domaine pour filtrer les nombres. L'attribut "l-termes-flous" donne les définitions des termes flous utilisables dans cet univers local. La distinction entre univers repèrable et non repèrable est utilisée pour le calcul des distances. De façon fondamentale, des valeurs numériques ne peuvent pas être utilisées dans un univers non repèrable. Une définition correcte des termes du vocabulaire dans chaque univers local conditionne le bon fonctionnement du processus général de filtrage.

4/ Autres définitions possibles

D'autres définitions des termes flous sont utilisables. Dans SPHINX (Fieschi 84), le vocabulaire d'un univers local est spécifié sous forme d'une hiérarchie de termes. La distance entre termes est calculée en utilisant la différence de profondeur entre ceux-ci dans la hiérarchie. Les problèmes de spécifications proviennent de la difficulté à situer un terme par rapport aux autres dans la hiérarchie définissant le vocabulaire.

Dans notre cas, la spécification à l'aide de fonctions trapézoidales est simple et facilement modifiable. Mais, on ne peut définir un terme que de façon continue et non pas comme la réunion de plusieurs sous-ensembles flous. On est aussi obligé de toujours se rapporter à une échelle numérique. Cette référence n'est pas un vrai problème selon des études de psychologie (Le Ny 79, Fraisse 69). Quelque soit le mode de spécification, le problème subsistera pour définir les termes flous. Ce problème est indépendant du mode de définition et inhérent à

la nature des termes et du problème.

VI/ Manipulations des valeurs terminales

Nous exposons ici deux types de manipulations. Les unes nécessaires à l'évaluation d'une distance entre deux valeurs terminales (Cayrol 82, Ishizuka 82), les autres pour des combinaisons de valeurs lors du multi-héritage. Le terme de distance n'est pas à prendre ici au sens mathématique. On utilise ce terme au niveau du matching atomique et global. Dans ce dernier cas, l'évaluation rendue n'est pas une distance car elle ne respecte pas en particulier l'axiome de symétrie. L'interprétation de l'évaluation n'est pas habituelle puisque dans notre étude, une valeur nulle signifie que les éléments mis en correspondance sont disjoints. Lors du filtrage atomique, les valeurs terminales sont de même type, c'est à dire définies dans le même univers local.

1/ Evaluation de distances

Si le type des deux valeurs manipulées est SYMBOLE, la distance est 1 pour une égalité stricte, 0 sinon.

Si le type est NOMBRE, avec une erreur ε précisée dans la structure des schémas, la distance entre les deux valeurs numériques x et y est:

si $x \in [y-\varepsilon, y+\varepsilon]$ alors 0
sinon si $\varepsilon < 1$ alors $1 - |x - y|$
sinon $1 - \frac{|x - y|}{\varepsilon}$

Un exemple codé avec l'univers local NOMBRE est donné ci-dessous. La fonction "distance1" calcule la distance entre deux valeurs.

```
: (affec 'NOMBRE '%erreur '$valeur 0.5)
= ()
```

```
: (impression 'NOMBRE 3 '(%erreur))
NOMBRE

    attribut : %erreur

      valeur : 0.5
      type : (UN D NOMBRE)
      import : 0.3

= t

: (distance1 5.8 () 6.2 () 'NOMBRE)
= 0.6
```

Dans un univers local repèrable, des calculs identiques sont possibles. Dans l'univers MASSE par exemple :

```
: (impression 'MASSE 3 '(%erreur))
MASSE

    attribut : %erreur

      valeur : 4.
      type : (UN D NOMBRE)
      import : 0.3

= t
: (distance1 35.6 () 38.2 () 'MASSE)
= 0.35
```

Dans les autres cas, le calcul de la distance varie selon le type des valeurs terminales utilisées. Dans notre cas, des valeurs floues et exactes peuvent être manipulées simultanément, avec une notion d'importance.

-Avec deux valeurs terminales floues x et y définies par les distributions de possibilités μ_X et μ_Y, la distance est le degré d'intersection de (Ishizuka 82).

Afin d'éviter une discontinuité entre les évaluations de distances calculées avec deux valeurs exactes et deux valeurs floues, on utilise une fonction de tolérance qui permet d'étendre la distribuion de possibilité de la valeur X provenant du filtre. On utilise pour cela la distribution $\mu_\varphi(0\ 0\ \varepsilon\varepsilon)$ associée à l'erreur ε. La

distibution $\mu_{X \circ \varphi}$ définie par:

$$\mu_{X \circ \varphi}(u) = Sup_{v \in U} \min(\mu_X(v), \mu_\varphi(v,u))$$

remplace alors μ_X.

D'un point de vue pratique, on uilise pour $\mu_{X\ \varphi}$ la distribution définie par:

soient μ_X (p p' b b')

μ_φ (0 0 ε ε)

$\mu_{X'} = \mu_{X \circ \varphi}$ = (p p' b+ε b'+ε)

Le degré d'intersection de $\mu_{X'}$ et μ_Y est donné par:

$$J(x,y) = \frac{\max_{u \in I}(\mu_{X' \cap Y}(u))}{\min(max_{u \in I}\mu_{X'}(u), \max_{u \in I}\mu_Y(u))}$$

avec $\mu_{X'}$ = (p p' b1 b2) μ_Y = (q q' c1 c2)

I(X') = [p - b1, p' + b2] I(Y) = [q - c1, q' + c2]

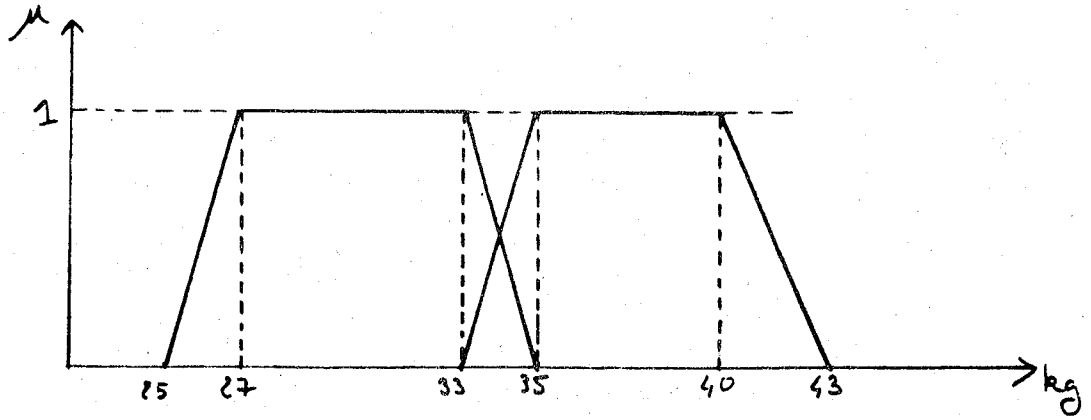
$I = I(X' \cap Y) = I(X') \cap I(Y)$

$\mu_{X' \cap Y}(u) = \min_{u \in I}(\mu_{X'}(u), \mu_Y(u))$

Un exemple simple facilite la présentation:

soient deux termes flous x = poids-plume avec μ_X=(27 33 2 2)
y = au-dessus-30 avec μ_Y=(35 40 2 3)
et ε = 0

I(X') = [25 35] I(Y) = [33 43] I = [33 35]



$\max(\mu_{X' \cap Y}(u)) = 0.5$ pour u = 34

$\min(\max \mu_{X'}(u), \max \mu_Y(u)) = 1$

d'où $J(x,y) = 0.5 = \mu_{X'}(34) = \mu_Y(34)$

La maquette détaillée au chapitre 5 a été réalisé de facon modulaire. Il est en particulier très facile de changer les fonctions utilisées pour l'évaluation des distances entre les valeurs terminales.

Le calcul exposé ici, à l'aide d'un degré d'intersection, peut être changé ou amélioré, par exemple en le combinant avec un degré d'inclusion des fonctions de distribution. Un tel coefficient est donné par (Ishizuka):

$$I(x,y) = \frac{\min_{u \in I}(1, 1 - \mu_{X'}(u) + \mu_Y(u))}{\max_{u \in I}(\mu_{X'}(u))}$$

Dans le même univers local MASSE, mais avec des valeurs floues, un exemple de calcul de distance est:

```
: (impression 'MASSE 3 '(%l-termes-flous))
MASSE
    attribut : %l-termes-flous

        valeur : ((nulle-faible 1.25 3.25 0.75 0.75) (lourd 85. 100. 5. 5.)
        (plutot-lourd 70. 80. 5. 5.) (leger 40. 60. 5. 5.) (faible 2. 5. 1.
        1.) (nulle 0. 1. 0. 1.))
        type : (LISTE D SYMBOLE)
        import : 0.3

: (distance1  'faible () 'nulle () 'MASSE)
= 0.5
```

-Avec une valeur floue x définie par μ_X et une valeur terminale exacte y.

num(y) = y si c'est un nombre
la valeur associée comme définition sinon

distance = μ_X(num (y))

-Avec deux valeurs terminales exactes x et y, les formules utilisées sont celles

décrites dans le cas de deux valeurs numériques, en employant alors comme valeurs num(x) et num(y).

Ce dernier cas est difficile à traiter car il y a peu d'informations. Les coefficients calculés n'ont pas de signification et ne servent qu'à une évaluation de distance relative.

Pour permettre la manipulation d'expressions arithmétiques avec des valeurs floues, les opérateurs arithmétiques (<, <=, <>, =, >=, >) ont été redéfinis. Nous n'utilisons ici que des heuristiques très simples mais suffisantes a priori, sachant que des outils de la logique floue permettront si nécessaire d'implémenter un mécanisme plus élaboré en particulier avec la notion d'opérateurs flous. Dans notre étude, soit x une valeur terminale, traduite de la manière suivante si elle apparaît dans une expression arithmétique :

trad (x) = num (x) comme définit ci-dessus

$= \frac{p + p'}{2}$ si x est définie par la fonction (p p' b b')

Ainsi l'expression (x <= y) est évaluée sous la forme (trad(x) <= trad(y)). Il en est de même avec <, >=, >. Les opérateurs = et <> utilisent en plus l'erreur ε précisée dans l'univers local.

Ainsi (x = y) est évaluée sous la forme $\text{trad}(x) \in [\text{trad}(y) - \varepsilon, \text{trad}(y) + \varepsilon]$, et (x <> y) sous la forme $\text{trad}(x) \notin [\text{trad}(y) - \varepsilon, \text{trad}(y) + \varepsilon]$. Ces expressions sont instanciées et évaluées lors du filtrage de structure.

Toujours avec l'univers local MASSE pour typer l'attribut "%masse-ressort" dans le schéma "Modèle-physique-ressort", la traduction de termes flous est donnée par :

```
: (traduc 'nulle)
= 0.5
: (traduc 'faible)
= 3.5
: (>= faible nulle)
= 1.
: (>= faible 10.0)
= 0.
```

2/ Combinaisons de valeurs

En cas de conflit entre attributs hérités lors du multi-héritage, il est nécessaire de pouvoir combiner les valeurs, pour conserver l'avantage de description avec des valeurs floues. Ceci est pratiqué pour les valeurs (facette $valeur)et les valeurs par défaut (facette val-defaut) de deux attributs en conflit. Là encore, les modes de combinaisons utilisés sont heuristiques, simples, tiennent compte des coefficients d'importance et permettent une gestion de cohérence des valeurs manipulées lors de l'héritage.

Au niveau des valeurs (facette $valeur):

-si les importances des attributs sont différentes, on hérite de la valeur du plus important

-sinon:

si l'intersection des valeurs est vide alors

on signale une incohérence potentielle et on hérite du premier

sinon

si ce sont 2 valeurs exactes (donc égales) alors on hérite de cette valeur

si ce sont 2 valeurs floues alors on hérite de leur intersection

sinon on hérite de la valeur exacte.

On définit l'intersection de deux valeurs floues spécifiées respectivement avec (p p' b b') et (q q' c c') , par (i j d d') et:

(i j) = intersection des segments (p p') et (q q')

d = moyenne de b et c

d' = moyenne de b' et c'

L'intersection est reconnue vide si (i j) est vide. Dans ce premier cas, si deux attributs ont des valeurs disjointes, le système signale une erreur potentielle car il est apparamment incorrect de vouloir hériter en provenance d'un même domaine, pour un même sujet, de deux valeurs totalement disjointes. Une gestion de cohérence au niveau des valeurs héritées est ainsi possible.

Un exemple de combinaison de valeurs floues au niveau des facettes "$valeur" est donné avec les deux attributs suivants:

: (impression 'Modèle-physique-ressort 1 '(%masse-ressort))
Modèle-physique-ressort

```
    attribut : %masse-ressort

       valeur : nulle
       type : (UN D MASSE)
       import : 1.
       defaut : nulle
       si-besoin : (interroger-utilisateur 'masse-ressort 'U '
       Modèle-physique-ressort)

= t

: (impression 'Modèle-physique-bloc 1 '(%masse-bloc))
Modèle-physique-bloc

    attribut : %masse-bloc

       valeur : faible
       type : (UN D MASSE))
       import : 1.
       defaut : faible
       domaine : (nulle faible moyen)

= t

: (conflit1% 'Modèle-physique-ressort '%masse-ressort
        'Modèle-physique-bloc '%masse-bloc)

= (nulle-faible 1. 2. 0.5 1.)
```

On affecte maintenant dans l'un des deux attributs en conflit une valeur disjointe de l'autre. Le processus de multi-héritage fournit la valeur du premier attribut et signale une incohérence potentielle:

```
: (affec 'Modèle-physique-ressort '%masse-ressort '$valeur 'lourd)
= ()

: (conflit1% 'Modèle-physique-ressort '%masse-ressort
        'Modèle-physique-bloc  '%masse-bloc)

attention : pb pour l'heritage de valeur d'attributs de classe
= lourd
```

Au niveau des valeurs par défaut des attributs en conflit, quelque soit leur type (exact, flou), on génère une valeur floue par un calcul de barycentre . Soient:

V1 = la valeur par défaut du premier attribut si c'est une valeur exacte ou la moyenne de m1 et m2 si c'est une valeur floue définie par (p p' b b') avec m1 = p - b et m2 = p' + b'.

V2 = même chose pour le second attribut

C1 = le coefficient d'importance du premier attribut s'il est non vide, 0.5 sinon.

C2 = la même chose pour le second attribut.

X1 = la moyenne de b et b' si la valeur par défaut du premier attribut est définie par (p p' b b'), une erreur choisie auparavant sinon.

X2 = la même chose pour le second attribut.

La valeur floue générée est : (G - d, G + d, Y , Y), où:

G = barycentre (V1 V2 C1/C1+C2 C2/C1+C2)

d = la valeur V1 ou V2 associée à l'attribut d'importance maximum

Y = barycentre (X1 X2 C1/C1+C2 C2/C1+C2)

Avec les mêmes schémas et attributs en conflit que ci-dessus, un exemple de combinaison de valeurs par défaut est:

```
: (conflit2% 'Modèle-physique-ressort '%masse-ressort
        'Modèle-physique-bloc  '%masse-bloc)

= (nulle-faible 1.25 3.25 0.75 0.75)
```

La combinaison des mêmes valeurs floues "nulle" et "faible" est plus large au niveau des valeurs par défaut que pour les valeurs. Une valeur par défaut est mieux spécifiée et plus utile avec une valeur floue. C'est pourquoi la combinaison de valeurs par défaut doit rendre de préférence une valeur floue. L'utilisation de calculs barycentriques permet de manipuler de façon simple les notions d'imprécis apportées par les valeurs floues et les notions d'importance liées aux attributs.

VII/ Le processus général de filtrage réalisé

1/ Le processus de filtrage élémentaire

Le processus général prend en entrée essentiellement deux schémas. Le premier est considéré comme le filtre à compléter s'il y a lieu. S'il n'y a pas plus d'informations, le système utilise les "vues standards" de chaque schéma.

Ces vues sont constituées d'informations apportées dans l'attribut SELF, mais aussi d'informations par défaut distribuées sur les attributs. C'est le cas pour les importances relatives des attributs, qui sont initialisées avec une valeur par défaut de 0.5.

Mais afin de moduler les vues utilisées, on peut spécifier les paramètres suivants:

-une borne d'importance, entre 0 et 1, indiquant à partir de quelle valeur du coefficient d'importance, un attribut est important pour le filtrage.

-une borne de filtrage, entre 0 et 1, indiquant en dessous de quelle distance minimum le filtrage est un échec.

-des importances pour chaque attribut du filtre et du fait, ce qui permet d'utiliser des aspects particuliers de chaque entité.

Les techniques de filtrage à-plat et de filtrage en profondeur sont basées sur un processus élémentaire qui se décompose en trois étapes: le filtrage hiérarchique, le filtrage structurel et le filtrage atomique. Dans la première étape, le système vérifie que le filtre se situe bien en dessous du fait, dans la même hiérarchie. Lors de la seconde étape, seule la structure des schémas, c'est à dire le nom des attributs et leur type, est manipulée. En dernier lieu, des distances sont évaluées entre les valeurs terminales des schémas. Les différentes étapes doivent être utilisées dans cet ordre. Mais des filtrages partiels peuvent être pratiqués en n'utilisant que la première ou la seconde étape.

Cette structuration du processus est nouvelle. Elle permet un raisonnement par analogie en fondant l'association entre objets sur les attributs et les valeurs.

Un raisonnement par typologie, aussi possible, permet d'associer des objets en ne mettant en correspondance que leurs structures, c'est à dire le nom et le

type de leurs attributs, sans utiliser leurs valeurs.

2/ Utilisation des réflexes

Les réflexes "si-besoin", s'il y en a dans le premier schéma, c'est à dire celui vu comme filtre, sont utilisés pour obtenir des valeurs. Ils activent les programmes adéquats et rendent les valeurs nécessaires pour instancier les schémas. Afin de conserver une base d'objets dynamique, les informations obtenues par recherches ne sont pas mémorisées définitivement, sauf sur demande explicite de l'utilisateur. C'est le cas en particulier au cours d'un raisonnement par analogie ayant pour but de compléter la description du filtre.

3/ Manipulation des variables

Les variables, s'il y en a, sont aussi traitées et instanciées lors du filtrage de structure. Elles peuvent apparaître dans les deux schémas, ce qui implique l'utilisation de procédures d'unification.

Elle ne sont utilisables que comme valeurs d'une facette. Une variable comme nom d'attribut n'aurait aucune signification sémantique. De plus, l'association entre attribut se fait essentiellement sur leur type et non pas sur leur nom.

Les instanciations obtenues sont propagées dans tout le schéma source, afin en particulier d'instancier et de vérifier les restrictions sur les variables. Les résultats des réflexes sont ainsi recueillis et placés où il le faut. Par exemple, avec le traitement de diagonalisation :

```
(Diagonalisation
    sorte-de
      $valeur Traitement
    matrice
      type UN IND Matrice-carree
      importance 0.5
    val-prop
      type UN IND Vecteur
      $valeur ?valprop
      importance 0.5
```

```
    si-besoin (diagonal !matrice ?valprop
                      ?vecprop ?succes))
  vec-prop
    type UN IND Matrice-carree
    $valeur ?vecprop
    importance 0.5
  succes
    type UN D Booleen
    $valeur ?succes
    importance 0.5)
```

Les résultats ?valprop, ?vecprop et ?succes du programme de diagonalisation permettent d'instancier l'individu "Diagonalisation" concerné.

4/ Traitement des références imbriquées

Toute valeur donnée pour une facette est par définition un filtre, donc une référence à un schéma. Pour chacune, sauf si elle est reconnue comme valeur terminale, le système génère un appel récursif au processus général de filtrage. Cet appel n'est généré que si les types donnés pour ces références sont des types identifiables.

VIII/ Le filtrage structurel

Les structures des schémas, constituées du nom des attributs et de leur type, sont mises en correspondance. En cas de réussite, le processus rend une liste de couples dans laquelle chaque attribut important du filtre est mis en correspondance avec un attribut du fait. Des règles simples régissent cette étape :

R1 : tout attribut important du fait est mis en correspondance avec un attribut du filtre.

R2 : tout attribut important du filtre est mis en correspondance avec un attribut du fait.

R3 : un attribut est important si son coefficient propre d'importance est supérieur ou égal à la borne minimum d'importance précisée dans la structure de l'objet ou à l'appel du processus de filtrage.

R4 : on n'associe à chaque attribut du filtre qu'un attribut au plus du fait.

R5 : des attributs sont mis en correspondance si et seulement si :
ils ont des types identifiables et même nom
ou ils ont des types identifiables.

R6 : Deux types T1 et T2 sont identifiables si et seulement si T1 est égal à T2 ou hiérarchiquement inférieur.

R7 : il y a erreur si deux attributs ont même nom et des types différents.

L'utilisation essentiellement du type des attributs pour les associer permet donc de fonder la correspondance surtout sur leur contenu, c'est à dire leur sémantique (Reimer 83) plutôt que sur leur syntaxe, c'est à dire leur nom.

Deux types identifiables ne sont pas toujours exactement égaux. Une distance "d-types" entre types est calculée comme suit :

soient deux types T1 et T2 identifiables

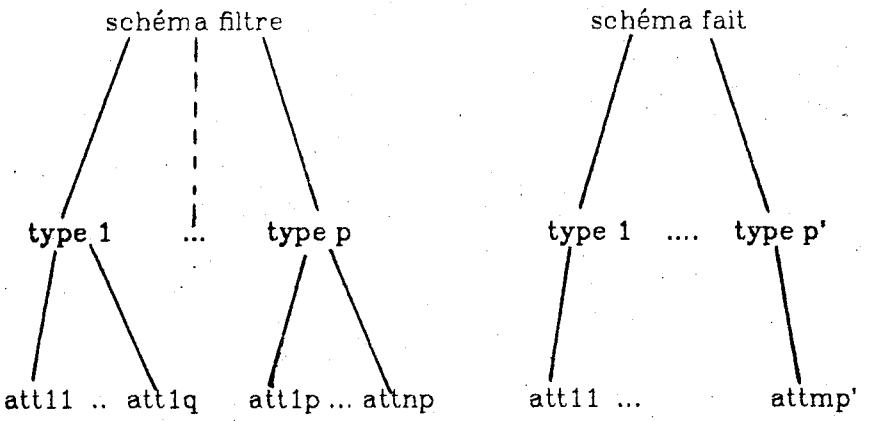
si T1 est hiérarchiquement supérieur à T2 alors d-types = 0
sinon
si T1 = T2 alors d-types = 1
sinon N = niveau(T1) -niveau (T2)

d-types = 1/ (N + 1)

où niveau(T) désigne le niveau de l'univers local T dans la hiérarchie des schémas.

Il n'est pas nécessaire ici d'essayer toutes les associations possibles entre chaque attribut du filtre et du fait. Elles sont basées sur la sémantique, traduite par le type des attributs, et doivent pouvoir se faire de façon unique pour éviter cette recherche.

On utilise une structure arborescente pour chaque schéma:



On détermine pour chaque attribut du filtre les attributs du fait de types identifiables. S'il n'y en a aucun et que l'attribut concerné du filtre est important, c'est un cas d'échec global.

Si pour un attribut "atti" du filtre il y a plusieurs attributs utilisables, le système recherche parmi eux un attribut de même nom que "atti". En cas d'échec, l'attribut le plus important et non déjà utilisé dans la liste des candidats est choisi.

Plusieurs attributs d'un même schéma ne doivent pas avoir le même type, ce qui dénote un manque d'information sémantique.

Dans l'exemple de modèle physique, utilisé jusqu'alors, l'attribut "masse-ressort" est mieux typé avec MASSE qu'avec NOMBRE, ce qui reste possible. Le type MASSE apporte plus d'informations sur la signification de l'attribut.

Cette spécification des connaissances oblige à accorder une grande importance au typage, ce qui peut paraître contraignant, mais permet une utilisation plus fine et plus performante des schémas.

Une spécification incorrecte des types peut être la cause de mauvaises associations entre attributs lors du filtrage structurel. Mais ces incorrections seront mises en évidence lors du filtrage atomique, par l'impossibilité d'associations entre les valeurs. Seuls les types SYMBOLE et NOMBRE peuvent être utilisés plusieurs fois dans un schéma. L'association entre attributs ainsi typés se fait par leurs noms.

Cette technique évite ainsi la complexité d'une recherche dans une arborescence d'associations très coûteux et peu performante dans la plupart des systèmes.

La notion d'importance relative d'un attribut par rapport à l'objet est primordiale dans cette étape. Un échec du filtrage structurel seulement au niveau des attributs reconnus importants entraîne un échec global. Les autres attributs sont négligés pour l'association entre les schémas. Mais ce sont eux qui apportent les informations permettant de compléter le filtre à chaque étape lors d'un filtrage en profondeur.
Un tel procédé avec la possibilité de faire varier les coefficients d'importance des attributs permet de manipuler divers aspects des objets. Des filtrages incomplets sont possibles, en ne retenant qu'une liste partielle d'attributs à considérer comme importants, sur lesquels on fonde l'association des deux schémas. Là encore le raisonnement est nuancé puisque les importances varient entre 0 et 1, et sont utilisées pour calculer la distance finale rendue.

IX/ Le filtrage atomique

Si le filtrage structurel n'est pas un échec, on cherche alors à calculer une distance entre les valeurs des attributs mis en correspondance. Une distance atomique "d-att" est calculée pour chaque couple. Le calcul de cette distance se fait selon la nature des valeurs terminales et les techniques exposées au

paragraphe VI-1 de ce chapitre. La valeur rendue "d-att" est interprétée comme ceci:

- si d-att = 0 alors les valeurs sont disjointes
- si d-att = 1 alors les valeurs sont égales
- si 0 < d-att < 1 alors les valeurs sont voisines

Le filtrage atomique rend pour chaque couple d'attributs formé lors du filtrage structurel, dont les types sont distants de $d-types_i$, une paire de valeurs numériques imp_i , $d-att_i$ où imp_i est l'importance de l'attribut du filtre et $d-att_i$ est la distance entre les valeurs des attributs. La distance globale D entre le filtre et le fait est donnée par:

$$D = \frac{\sum_i imp_i * d-types_i * d-att_i}{\sum_i imp_i}$$

Le fait est retenu si D est supérieure à une borne minimum de filtrage précisée au préalable.

Le processus de filtrage exposé peut donc être considéré comme un problème d'optimisation de la fonction:

$$F(X/Y_j) = \frac{\sum_{u \in A} imp(u) * d-types(u) * d-att(u)}{\sum_{u \in A} imp(u)}$$

où A = liste des attributs du filtre X

et Y_j le schéma utilisé parmi Y1 Y2 ... Yp les p schémas testés

Le tout en ayant fixé une vue utile pour chaque schéma manipulé.

Un exemple de filtage complet, avec les schémas définis plus haut, est donné ci-dessous. On utilise en premier lieu une borne d'importance classique, égale à 0.5, et aucun paramètre dans le processus de filtrage. Les schémas utilisés ont peu de valeurs.Tous les coefficients d'importance non précisés sont égaux à 0.5. L'association se fait essentiellement selon la structure.

```
(Modèle-physique-ressort
   sorte-de
```

```
      $valeur Modèle-physique
    masse-ressort
      type UN IND MASSE
      $valeur nulle
    loi-physique
      type UN IND LOI
    force-friction
      type UN D SYMBOLE
    force-traction
      type UN D SYMBOLE
    commentaires
      type UN IND TEXTE
    modèle-mathématique
      type UN IND Modèle-math)


  (Modèle-physique-bloc
    sorte-de
      $valeur Modèle-physique
    masse-bloc
      type UN IND MASSE
      $valeur  faible
    vitesse-initiale
      type UN D NOMBRE
    force-frottement
      type UN D SYMBOLE
    loi
      type UN IND LOI
    pente
      type UN D NOMBRE
    commentaires
      type UN IND TEXTE
    modèle-mathématique
      type UN IND Modèle-math)
```

: (match 'Modèle-physique-ressort 'Modèle-physique-bloc)

matching elementaire entre Modèle-physique-ressort et Modèle-physique-bloc

resultats du filtrage structurel :

associations filtre-fait : ((%l-termes-flous . %l-termes-flous)
(%propcarac . %propcarac) (%erreur . %erreur) (%borne-sup . %borne-sup)
(%borne-inf . %borne-inf) (%type-U . %type-U) (vue-par . vue-par)
(vue-de . vue-de) (super . super) (precedent . precedent)
(%modèle-math . %modèle-math) (%commentaires . %commentaires)
(%force-traction)(%force-friction) (%loi-physique . %loi)
(%masse-ressort . %masse-bloc))

verifications de ces associations dans le sens fait-filtre :
((%l-termes-flous . %l-termes-flous) (%propcarac . %propcarac)
(%erreur . %erreur) (%borne-sup . %borne-sup) (%borne-inf . %borne-inf)
(%type-U . %type-U) (vue-par . vue-par) (vue-de . vue-de)
(super . super) (precedent . precedent) (%modèle-math . %modèle-math)
(%commentaires . %commentaires) (%pente . %pente)
(%vitesse-initiale . %vitesse-initiale)
(%force-frottement . %force-frottement)(%loi . %loi)
(%masse-bloc . %masse-bloc))

distance de filtrage structurel : 1.

resultats du filtrage atomique :

distance entre les element des couples :
((%l-termes-flous . %l-termes-flous) (%propcarac . %propcarac)
(%erreur . %erreur) (%borne-sup . %borne-sup) (%borne-inf . %borne-inf)
(%type-U . %type-U) (vue-par . vue-par) (vue-de . vue-de)
(super . super) (precedent . precedent) (%modèle-math . %modèle-math)
(%commentaires . %commentaires) (%force-traction) (%force-friction)
(%loi-physique . %loi) (%masse-ressort . %masse-bloc))
((0.3 1.) (0.3 1.) (0.3 1.) (0.3 1.) (0.3 1.) (0.3 1.) (0.3 1.)
(0.3 1.) (0.3 1.) (0.3 1.) (0.3 1.) (0.3 1.) (0.5 1.) (1. 0.75))
= (Modèle-physique-ressort
Modèle-physique-bloc 9.509803921568627e-1 ())

Les deux schémas dans ces conditions sont identifiables et distants de 0.95. Des attributs du filtre, tels que "%force-traction" et "%force-friction", n'ont pas été associés. Mais ils n'etaient pas importants dans la vue utilisée du filtre.

On peut utiliser une vue complète du filtre en imposant une borne d'importance plus petite, égale à 0.4 :

: (match 'Modèle-physique-ressort 'Modèle-physique-bloc 0.4)

matching elementaire entre Modèle-physique-ressort et
Modèle-physique-bloc

resultats du filtrage structurel :

associations filtre-fait : ((%l-termes-flous . %l-termes-flous)
(%propcarac . %propcarac) (%erreur . %erreur) (%borne-sup . %borne-sup)
(%borne-inf . %borne-inf) (%type-U . %type-U) (vue-par . vue-par)
(vue-de . vue-de) (super .super) (precedent . precedent)
(%modèle-math . %modèle-math) (%commentaires . %commentaires)
(%force-traction . ****) (%force-friction . ****)
(%loi-physique . %loi) (%masse-ressort . %masse-bloc))

verifications de ces associations dans le sens fait-filtre :
((%l-termes-flous . %l-termes-flous) (%propcarac . %propcarac)
(%erreur . %erreur) (%borne-sup . %borne-sup) (%borne-inf . %borne-inf)
(%type-U . %type-U) (vue-par . vue-par) (vue-de . vue-de)
(super . super) (precedent . precedent) (%modèle-math . %modèle-math)
(%commentaires . %commentaires) (%pente . %pente)
(%vitesse-initiale . %vitesse-initiale)
(%force-frottement . %force-frottement)(%loi . %loi)
(%masse-bloc . %masse-bloc))

distance de filtrage structurel : 0.

Le filtrage est un échec dés la premiére étape, car les attributs du filtre "%force-traction" et "%force-friction" sont importants dans la vue utilisée et n'ont pas pu être associés à un attribut du fait.

X/ Exploitations différentes d'un même objet

et divers types de raisonnements

Le processus d'associations décrit permet des raisonnements par analogie et typologie (Carbonell 83, Nagel 83), comme nous les avons définis au paragraphe VII-1.

Un raisonnement du premier type est obtenu avec un processus complet: filtrage hiérarchique, structurel et atomique. On utilise la sémantique des objets, et la manipulation de valeurs floues permet des correspondances plus souples. C'est donc un raisonnement plus puissant que ceux qui fondés uniquement sur des identifications syntaxiques exactes des noms d'attributs, comme c'est le cas dans la plupart des systèmes informatiques.

Un raisonnement par typologie est obtenu en ne tenant compte que de la structure des schémas, donc en n'utilisant qu'une étape de filtrage structurel.

L'utilisateur peut modifier la "vue standard" de chaque schéma en spécifiant une borne minimum de filtrage, une borne d'importance, et un coefficient d'importance pour chaque attribut. Un schéma peut donc être manipulé sous différents aspects. Un aspect particulier est défini par une partition de l'ensemble global des attributs, constituée par ceux qui sont importants dans cette vue, c'est à dire qui ont une importance propre supèrieure à la borne minimum d'importance. Ainsi, le schéma décrivant le "Modèle-physique" peut être vu comme un concept purement physique en ne tenant compte que des quatre premiérs attributs, ou comme un concept mathématique en utilisant le dernier attribut, ou encore comme un modèle physique particulier en considérant plus particuliérement, par exemple, le type de loi vérifiée par le ressort (attribut "loi"). On peut identifier des modèles physiques entre eux en ne tenant compte que de cet attribut, ce qui permet bien une typologie sur un critère particulier.

Le filtrage structurel et la possibilité d'utiliser différentes vues d'un même schéma permettent des "filtrages partiels" (Hayes-Roth 84). Ce processus a pour but de mettre en évidence les ressemblances entre concepts mais aussi et surtout les différences. Le résultat du filtrage structurel précise aussi les attributs importants des schémas qui n'ont pas pu être mis en correspondance avec d'autres. Ils sont la cause des différences entre objets.

En reprenant les exemples de modèles physiques utilisés au chapitre précédent, on peut définir deux schémas, l'un pour décrire l'oscillation d'un ressort et l'autre le glissement d'un bloc sur une pente.

Dans le premier cas, on a:

```
(Modèle-physique-ressort
   sorte-de
     $valeur Modèle-physique
```

```
masse-ressort
  type UN IND MASSE
  val-defaut faible
  si-besoin (Interroger-utilisateur !masse-ressort)
  importance 1.0

loi-physique
  type UN IND LOI
  importance 0.7

force-friction
  type UN D SYMBOLE
  importance 0.5

force-traction
  type UN D SYMBOLE
  importance 0.5

commentaires
  type UN IND TEXTE
  importance 0.5

modèle-mathématique
  type UN IND Modèle-math
  importance 0.5   )
```

Puis pour le second phénoméne physique :

```
(Modèle-physique-bloc
  sorte-de
    $valeur Modèle-physique

  masse-bloc
    type UN IND MASSE
    val-defaut faible
    importance 1.0

  vitesse-initiale
    type UN D NOMBRE
    val-defaut 0.0
    importance 0.5
```

```
force-frottement
  type UN D SYMBOLE
  importance 0.5

loi
  type UN IND LOI
  importance 0.5

pente
  type UN D NOMBRE
  importance 0.5
  si-besoin (Interroger-utilisateur !pente)

commentaires
  type UN IND TEXTE
  importance 0.5

modèle-mathématique
  type UN IND Modèle-math
  importance 0.5   )
```

Un filtrage structurel avec ces deux schémas et une borne de filtrage égale à 0.4 nous montre que seuls les attributs: masse-ressort, masse-bloc, loi, loi-physique et modèle-mathématique sont associables sous la forme:

((loi, loi-physique)(masse-ressort, masse-bloc)(modèle-mathématique, modèle-mathématique)(commentaires, commentaires)(force-friction) (force-traction)(vitesse-initiale)(force-frottement)(pente))

Les attributs parasites sont ainsi localisés, puisque ce sont ceux pour lesquels aucune association n'est possible.

Des raisonnements par typologie successifs, en faisant varier les coefficients d'importance pour éliminer les attributs parasites, permettent de trouver une "vue commune" à plusieurs objets. La génération de concepts par abstraction est ainsi possible.

XI/ Conclusion et puissance du raisonnement

Le filtrage proposé permet la manipulation de valeurs floues pour faciliter la spécification des connaissances et améliorer la puissance du raisonnement. Il s'intègre dans une représentation centrée objets et sait manipuler la dynamique exprimée de façon déclarative. L'univers de discours utilisé correspond à la hiérarchie des concepts dans son ensemble.

Le processus manipule la sémantique des objets , l'imprécis sous forme de valeurs floues et l'importance des attributs.

Il permet un raisonnement nuancé et rend une évaluation de distance entre les objets mis en correspondance.

Des modes d'exploitation différents de la base d'objets sont permis et ainsi des raisonnements divers et nuancés sont utilisables, tels que des raisonnements par analogie et typologie.

Ce dernier type de mécanisme est facilité par les possibilités de manipuler un même schéma sous différents aspects. Le processus élaboré permet l'appentissage du système par génération de concepts (Ritchie 84, Winston 80).

La génération par spécialisation se fait simplement avec l'inférence par héritage propre à la représentation centrée objet. On génére un objet en fusionnant plusieurs entités données comme ses ancêtres.

La génération par abstraction, ou généralisation, se fait comme nous l'avons vu au paragraphe précédent, avec un filtrage partiel qui permet de rejeter les attributs parasites et de ne conserver que ceux qui sont communs aux schémas manipulés.

Avec l'exemple du paragraphe précédent, on peut générer par abstraction le modèle physique manipulant des objets avec masse et régit par une loi spécifique à trouver:

```
(Modèle-physique-g
    sorte-de
      $valeur Modèle-physique

    spec
      $valeur Modèle-physique-ressort
```

Modèle-physique-bloc

masse
  type UN IND MASSE
  importance 1.0

loi
  type UN IND LOI
  importance 0.5

commentaires
  type UN IND TEXTE
  importance 0.5

modèle-mathématique
  type UN IND Modèle-math
  importance 0.5)

Ces mécanismes permettent de construire des raisonnements complexes souvent utilisés par l'homme pour organiser ses connaissances et même créer de nouveaux éléments par abstractions successives avec de plus en plus de raffinement.

On peut:

-soit pratiquer des filtrages partiels successifs en imposant des bornes de filtrage et d'importance de plus en plus élevées mais en conservant les mêmes vues pour les schémas utilisés. On extrait ainsi des abstractions de plus en plus précises mais de même nature.

-soit pratiquer des filtrages partiels successifs en conservant les bornes mais en faisant varier les vues des schémas utilisés. Les abstractions obtenues peuvent être de natures différentes.

-soit pratiquer des filtrages successifs en faisant tout varier simultanément.

De tels mécanismes sont utiles dans une représentation des connaissances centrée objets. Ils offrent des possibilités nouvelles et nécessaires pour exploiter correctement cette représentation très riche en informations de toutes sortes.

Enfin, se pose le problème essentiel qui est de valider l'usage du filtrage flou et des termes du langage naturel. Beaucoup d'exemples donnés dans les chapitres précédents sont issus d'un domaine de modélisation mathématique rigoureux où l'utilisation de valeurs floues n'a pas lieu. Le but était de montrer comment le mécanisme d'exploitation effectue les recherches, élabore la dynamique et manipule les traitements à partir uniquement de descriptions.

Mais quelques exemples nous montrent aussi l'utilisation de valeurs floues pour la description d'un phénomène physique. On doit reconnaître des facilités d'expressions et constater que les nuances du raisonnement, qui constituent sa puissance, sont fonction de la bonne définition des valeurs floues.

Les facilités de spécifications à l'aide de termes du langage naturel ne peuvent être utilisées qu'aprés avoir défini un vocabulaire utile sous forme d'univers locaux et de valeurs floues. Ces définitions peuvent être relativement difficiles. Mais des expériences précédentes, en médecine (Fieschi 84) et dans les bases de données (Dubois 85a/), tendent à prouver l'utilité de ces outils.

Dans beaucoup de domaines de modélisation, avant de travailler à un niveau de modélisation rigoureux, l'expert doit pouvoir décrire le phénomène étudié à l'aide de termes du langage naturel pour utiliser sans problèmes vis à vis de la machine les connaissances imprécises et incertaines dont il dispose. S'il n'en est pas ainsi, l'expert ne doit fournir que des valeurs exactes. Il peut alors, en choisissant une valeur au hasard dans un ensemble de valeurs admissibles a priori, construire un modèle erroné.

C'est pour cette raison et pour permettre à un système informatique de travailler au niveau sémantique du domaine d'application que nous introduisons des valeurs floues et le filtrage flou dans EDORA, qui est un système intelligent d'aide à la modélisation mathématique en biologie décrit au chapitre suivant.

Quoiqu'il en soit, si des problèmes insolubles au premier abord se posent pour la définition des valeurs floues, on peut toujours utiliser le mécanisme de façon classique, basé sur une logique binaire sans nuances.

CHAPITRE 5

LE SYSTEME EDORA

UNE UTILISATION DE LA REPRESENTATION CENTREE OBJETS

Chapitre V/ Le système EDORA :

une utilisation de la représentation centrée objets

PLAN

I/ Le domaine d'application

1/ Introduction
2/ Des systèmes existants d'aide à la modélisation

II/ Présentation du projet EDORA

1/ Introduction
2/ Utilisation d'une représentation centrée objets
3/ Conceptualisation

a/ Niveau sémantique
b/ Niveau de représentation physique
c/ Niveau de modélisation mathématique
d/ Niveau traitement

4/ Exemples de spécifications de schémas

III/ Réalisation de la maquette

1/ Schéma d'ensemble
2/ Langages de programmation utilisés
3/ Présentation du langage CEYX
4/ La représentation des connaissances implémentée
5/ Le filtrage réalisé

I/ Le domaine d'application

1/ Introduction

La modélisation mathématique permet dans de nombreux domaines de mieux exploiter les données, plus rapidement et plus efficacement. Il existe déjà des logiciels pour faciliter la spécification de modèles mathématiques, en temps continu sous la forme d'équations différentielles comme en temps discret sous la forme d'équations récurrentes (Rechenmann 84b/).

Mais la tendance à l'heure actuelle dans ce domaine est à l'augmentation du nombre des outils offerts et non pas à l'amélioration de l'aide réelle apportée. A côté de la connaissance du "comment" sous forme de programmes pour utiliser ces outils, il manque la connaissance du "quand" et du "pourquoi" (Vivet 84). L'approche par la modélisation mathématique reste donc peu courante car elle réclame une expertise supplémentaire que les systèmes informatiques n'apportent pas encore.

2/ Des systèmes existants d'aide à la modélisation

Les logiciels existants appartiennent à diverses catégories. Au niveau le plus bas, de nombreuses bibliothéques de programmes scientifiques (Harwell, Nag, IMSL) proposent entre autres des méthodes d'intégration et d'identification. Elles sont difficiles d'accès et d'utilisation.

Il existe des logiciels plus développés, pour lesquels la richesse en méthodes et surtout l'étendue du domaine d'application sont des critères fondamentaux d'évaluation. C'est le cas des logiciels ACSL (ACSL 75), CSSL, CSMP (CSMP 72), MLP (Spriet 82) ou même les logiciels dits d'économétrie comme TROLL, EPS ou MODULECO (Oudet 80, Vignard 83), qui sont utilisables dans toute situation où la modélisation se fait à l'aide d'équations récurrentes.

Les facilités d'accès et d'utilisation s'améliorent, même pour des utilisateurs qui ne possèdent pas obligatoirement la connaissance et l'expertise requises. Mais aucune aide sémantique n'est apportée par ces systèmes, pas même pour le choix des outils adéquats. Quelques exceptions existent, en biologie notamment.

Citons Biomod (Ganer 71), Saam, Cosmos (Hamrouni 79) qui offrent des langages spécialisés d'écriture de modèles. En général ces logiciels sont mal ou sous employés.

Globalement, l'aide apportée est jugée insuffisante à plusieurs points de vue :

Des outils sont disponibles mais sans aucun "mode d'emploi". Il n'y a pas en fait réellement aide à la modélisation. Les connaissances essentielles du "quand" et "pourquoi" ne sont pas fournies. Le logiciel devrait être au moins capable d'indiquer si une méthode est inadaptée, compte tenu du contexte de travail et de justifier sa réponse.

La nature itérative du processus d'analyse-modélisation n'est pas prise en compte. Il n'y a pas de gestion de cohérence des différentes versions du modèle en cours d'élaboration. Cet état de fait est regretable car un modèle ne vaut que par les hypothèses qu'il traduit et il est donc fondamental de retrouver les raisons qui ont conduit à sa construction et à sa définition.

Ces systèmes négligent aussi l'aspect sémantique du domaine d'application, c'est à dire toutes les connaissances ou interprétations spécifiques du domaine. Ils ne sont donc pas en mesure d'apporter une aide quelconque efficace durant la phase de formalisation. Ces deux dernières critiques font que l'utilisateur n'est en aucune manière assisté dans sa démarche heuristique de modélisation.

D'un point de vue plus technique, le manque de langage spécialisé, de facilités d'expression et le manque d'extensibilité sont aussi des inconvénients importants.

Il paraît donc intéressant d'incorporer aux logiciels de modélisation la connaissance liée aux méthodes offertes, aux objets manipulés ainsi qu'aux domaines d'application. Pour cela, une approche systèmes experts est faite dans EDORA (Gouze Vignard 84b/c/), en insistant sur la nécessité d'une prise en compte de la sémantique du problème et de l'expertise humaine.

II/ Présentation du projet EDORA

1/ Introduction

Le système EDORA (Equations Différentielles Ordinaires et Récurrentes Appliquées) est l'objet d'un projet de recherche au centre INRIA de Sophia Antipolis. La raison d'être de EDORA est de rendre l'approche de la modélisation mathématique accessible dans divers domaines, en particulier en biologie. C'est un système informatique intelligent dont le but est de soutenir la démarche heuristique du modélisateur, en lui apportant les méthodes mathématiques nécessaires et en lui montrant les possibilités et limites de ces outils. EDORA n'est pas une simple collection d'outils mais il n'impose pas non plus une démarche unique de modélisation. C'est un système interactif qui utilise la sémantique du problème à la fois pour améliorer son efficacité et pour se rapprocher de l'utilisateur.

2/ Utilisation d'une représentation centrée objets

Dans les chapitres précédents, nous avons mis en valeur les principaux avantages d'une représentation des connaissances centrée objets.

La connaissance est donnée de façon explicite et donc plus accessible au système lui même, ce qui améliore ses possibilités d'explications et de communications avec l'homme. La représentation est lisible, modulaire et modifiable. Les utilisations possibles du procédural et déclaratif simultanément, et de termes du langage naturel améliorent encore la puissance d'expression de cette représentation. Enfin l'organisation des objets est un atout majeur pour la spécification et l'utilisation des connaissances.

De façon plus intuitive, on ressent le besoin dans certains domaines tels que celui de la modélisation mathématique, de décrire les objets manipulés par exemple ici les modèles, équations, variables, méthodes.

Il est plus utile de disposer du schéma "Modele-physique" suivant:

```
(Modele-physique
   sorte-de
     $valeur Modele

   masse-ressort
     type UN IND MASSE
     $valeur nulle
     importance 1.0

   loi-physique
     type UN IND LOI
     $valeur Hooke
     importance 0.5

   force-friction
     type UN D SYMBOLE
     $valeur lineaire
     importance 0.4

   force-traction
     type UN D SYMBOLE
     $valeur nulle
     importance 0.4

   modele-mathematique
     type UN D SYMBOLE
     $valeur (dx/dt + g(dx/dt)/M + C2/M*x = 0)
     importance 0.3    )
```

que des règles de production suivantes, extraites d'une application du système CRIQUET (Vignard 84b/), qui le décrivent :

```
REGLE Physique NUMERO 2
SI ressort avoir force-retour lineaire
ALORS ajoutfait (ressort satisfaire loi Hooke)
COEFF : 0.8

REGLE Physique NUMERO 4
SI ressort avoir deplacement petit
ALORS ajoutfait (ressort avoir force-retour lineaire)
COEFF : 0.7
```

REGLE Physique NUMERO 10
SI m//M << 1
ALORS ajoutfait (non (ressort avoir masse))
COEFF : 0.9

REGLE Physique NUMERO 17
SI ressort satisfaire loi Hooke
ET non (ressort avoir masse)
ET ressort avoir force-friction
ET force-friction etre lineaire
ET non (ressort avoir force-traction)
ALORS ajoutfait (model etre (dx/dt + g(dx/dt)/M + C2/M*x = 0))
COEFF : 0.8

Les objets, tels que le modéle, sont souvent manipulés sous différents aspects. Le modèle peut être utilisé d'un point de vue physique et d'un point de vue mathématique, pour l'identifier et pour le simuler. Il faut donc avoir accès à ses propriétés physiques et mathématiques. Ces points de vue ne sont pas indépendants et les règles de production sont mal adaptées à ce genre de situation. Pour toutes ces raisons (Rechenmann 84b/), une représentation des connaissances centrée objets paraît plus adéquate.

EDORA regroupe trois aspects importants en informatique et rarement apparents simultanément.

La définition d'une base d'objets orientée vers la modélisation est un aspect fondamental du projet.

L'aspect algorithmique est aussi important comme dans tous travaux de modélisation. Le système sait gérer une bibliothèque d'algorithmes numériques et formels.

Enfin, comme un système expert, il peut stocker et utiliser des heuristiques et connaissances spécifiques au domaine de modélisation. Il sait répondre à des questions de l'utilisateur, par exemple sur les possibilités d'utilisation d'un traitement dans la situation donnée. Il doit aussi pouvoir fournir une méthodologie de modélisation mathématique à partir de la description du problème à traiter selon des termes propres à l'utilisateur.

L'utilisation du mécanisme d'exploitation décrit dans le chapitre précédent est

intéressant dans ce domaine comme dans tous ceux où l'on utilise une représentation des connaissances centrée objets. On peut ainsi utiliser des raisonnements nuancés en utilisant la sémantique des objets et du domaine, ce qui est un objectif du système EDORA.
La possibilité de spécification des connaissances à l'aide de termes du langage naturel facilite le travail de modélisation pour des experts biologistes qui ne sont pas informaticiens et préférent utiliser leur propre vocabulaire aprés l'avoir défini au système.

3/ Conceptualisation

La spécification des connaissances dans le domaine de la modélisation mathématique en biologie a permis de construire des schémas, répartis en quatre sous-ensembles interconnectés (Gouze Vignard 84c/). Nous emploierons le terme de niveau pour désigner ces sous ensembles qui selon les cas correspondent à différents niveaux d'appréhension du système global.

a/ Le niveau sémantique

C'est usuellement le niveau d'entrée dans le système. L'utilisateur définit les objets biologiques qu'il souhaite manipuler avec leurs caractéristiques pertinentes dans la situation concernée (ex: une espèce posséde un fort taux de croissance).

La signification propre de chaque objet, dépendante de l'utilisation, est ainsi facilement donnée au système. On évite de limiter l'expression à un formalisme purement mathématique, vide de toute signification spécifique au domaine.

Parmi les objets sémantiques, on distingue les objets élémentaires, les relations et les situations.
Les objets élémentaires sont les variables biologiques utilisées, par exemple les espéces.
Les relations, notées "types de schémas", représentent des processus biologiques classiques, c'est à dire des liens possibles entre les objets élémentaires.
Un objet situation correspond à une situation de travail biologique particulière. C'est un type de schéma, instancié avec des objets élémentaires particuliers.

b/ Le niveau de représentation physique

Le biologiste choisit alors un mode de représentation physique qui lui permet de formaliser le modèle biologique et d'en écrire les équations.

Les objets élémentaires sont représentés par des variables physiques et les processus biologiques par des processus physiques. La représentation physiques de la situation rassemble les variables et les processus physiques.

Ce niveau offre une interface claire entre le phénomène biologique et une expression mathématique formelle sous forme d'équations. Le biologiste peut encore à ce niveau indiquer dans les schémas des renseignements qualitatifs tels que l'ordre de grandeur des variables et les vitesses des processus.

c/ Le niveau de modélisation mathématique

A la représentation physique correspond une représentation mathématique sous forme d'un système différentiel.

Aux trois types d'objets du niveau sémantique correspondent les variables, types de modèles et modèles du niveau de modélisation mathématique. D'autres objets sont aussi utilisés à ce niveau.

Le système vérifie la cohérence entre les équations et la situation biologique initiale. Mais les objets mathématiques sont dégagés de toute connotation physique. Les équations construites sont manipulées de façon rigoureuse.

d/ Le niveau traitement

On retrouve l'aspect algorithmique cité précédemment, ou encore les schémas traitements introduits au chapitre III. Le système effectue les traitements mathématiques qui lui sont ordonnés, mais il peut aussi en proposer compte tenu du contexte, en expliquant son choix et en documentant les traitements

avancés.

Parmi les traitements utilisables, on peut citer:

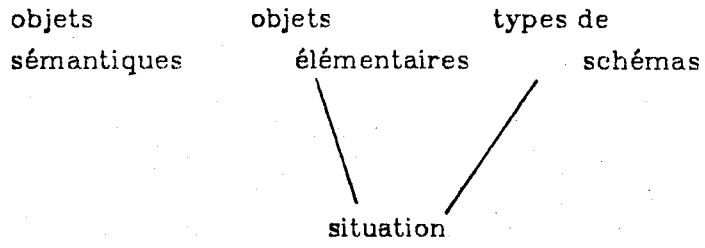
-l'intégration numérique.
-l'identification.
-l'analyse de sensibilité permettant d'apprécier l'influence des paramètres sur l'évolution du système.
-l'étude des points stationnaires et de leur stabilité.
-la recherche de solutions périodiques.
-la visualisation graphique d'une variable en fonction d'une autre.

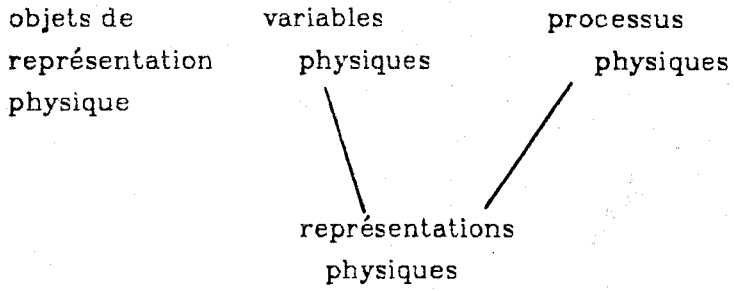
Ces objets traitements définissent la dynamique des objets en général. Ils s'appliquent surtout aux objets du niveau mathématique.

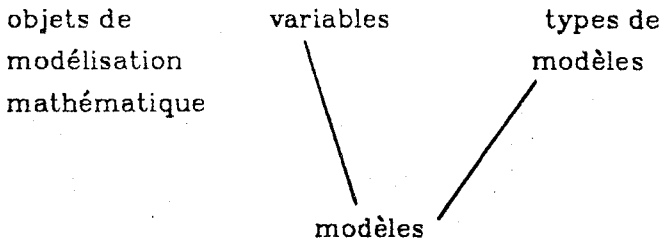
La classe des traitements est aussi divisée en plusieurs familles. Chacune peut contenir plusieurs éléments. La famille des traitements d'intégration comporte par exemple les méthodes de Runge-Kutta d'ordre 4 et la méthode de Gear.

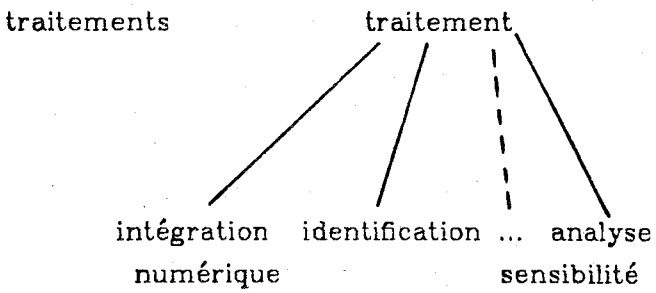
Le maintien effectif de la séparation entre les trois premiers niveaux est suggéré par la technique de modélisation des biologistes. Ces derniers travaillent plutôt au niveau sémantique qu'au niveau mathématique (Gouze Vignard 84b/c/).

La figure1 ci-dessous donne une vue simplifiée de la structure d'ensemble de la base d'objets:


objets sémantiques
objets élémentaires
types de schémas
situation


objets de représentation physique
variables physiques
processus physiques
représentations physiques


objets de modélisation mathématique
variables
types de modèles
modèles


traitements
traitement
intégration numérique
identification
...
analyse sensibilité

4/ Exemples de spécifications de schémas

Les objets sont décrits à l'aide de schémas. Comme nous l'avons vu, la sémantique des objets est apportée dans la description par les attributs. Pour préciser la modélisation des connaissances faite à l'aide des schémas dans le domaine abordé, nous donnons ci-dessous quelques exemples de schémas extraits de la maquette réalisée. Les exemples fournis sont les premiers résultats d'une phase de formalisation en cours au moment de la rédaction de cette thèse.

On ne cite que des schémas apparaissant au niveau sémantique. Des schémas des niveaux mathématique et traitement ont été utilisés comme exemples dans les chapitres précédents.

Les conventions d'écritures ont été données au chapitre III:

-SELF désigne le schéma courant.

-ATT désigne l'attribut courant.

-VAL désigne la valeur courante, c'est à dire celle de la facette manipulée dans l'attribut courant.

-!nom-attribut désigne la valeur de la facette "$valeur" de l'attribut "nom-attribut" dans le schéma courant.

-enfin, une fonction classique d'accès aux valeurs des facettes est:

(obtatt nom-schema nom-attribut nom-facette)

pour obtenir la valeur de "nom-facette" dans l'attribut "nom-attribut" du schéma "nom-schéma".

On décrit la racine de l'arborescence des objets sémantiques.

```
(OBJET-SEMANTIQUE
    sorte-de
        $valeur  UNIVERSEL
    spec
        $valeur  (OBJET-ELEMENTAIRE
            PROCESSUS
            SITUATION))
```

Les objets élémentaires sont détaillés.

```
(OBJET-ELEMENTAIRE
    sorte-de
        $valeur   OBJET-SEMANTIQUE
    spec
        $valeur   (VARIABLE-DE-SORTIE-BIOLOGIE
              VARIABLE-FORCANTE-BIOLOGIE
              VARIABLE-D-ETAT-BIOLOGIE)

    domaine
          type     UN D SYMBOL
          $valeur  dynamique-des-populations-marines
          import    0.5)


(VARIABLE-D-ETAT-BIOLOGIE
    sorte-de
        $valeur   OBJET-ELEMENTAIRE
    spec
        $valeur   POPULATION

    nom type UN D SYMBOLE
         import    1.0

    dynamique
        type      UN D SYMBOLE
        domaine   (discrete continue)
        si-besoin
             (INTERROGER-UTILISATEUR  !dynamique)
          a-verifier
            (if (equal VAL 'continue)
              (>= (obtatt !representation-math
                        'ordre '$valeur) 'grand)
               t)
          import     0.5

    assertions
        type LISTE D SYMBOLE
         import   0.4
```

```
    representation-math
        type UN IND VARIABLE-D-ETAT-MATH
        import   0.4)


(VARIABLE-DE-SORTIE-BIOLOGIE
    sorte-de
        $valeur   OBJET-ELEMENTAIRE
    spec
        $valeur   POPULATION

    nom type UN D SYMBOLE
        import   1.0

    dynamique
        type     UN D SYMBOLE
        domaine  (discrete continue)
        si-besoin
            (INTERROGER-UTILISATEUR !dynamique)
        a-verifier
            (if (equal VAL 'continue)
              (>= (obtatt !representation-math
                          'ordre '$valeur)  'grand)
               t)
        import   0.4

    assertions
        type LISTE D SYMBOLE
        import   0.4

    representation-math
        type UN IND VARIABLE-DE-SORTIE-MATH
        import   0.4

    variables-d-etat-observees
        type LISTE IND VARIABLE-D-ETAT-BIOLOGIE
        import   0.5

    variables-forcantes-observees
        type LISTE IND VARIABLE-FORCANTE-BIOLOGIE
```

```
        import   0.5

    relation-avec-ces-variables
        type UN INF FONCTION
        import   0.5)


(POPULATION
    sorte-de
            $valeur  (VARIABLE-DE-SORTIE-BIOLOGIE
                VARIABLE-D-ETAT-BIOLOGIE)

    nom-pop
        type UN D SYMBOLE
        $valeur
          import   1.0

    ordre-de-grandeur
        type UN IND ORDRE-DE-GRANDEUR
        defaut   grand
        si-besoin
           (INTERROGER-UTILISATEUR !ordre-de-grandeur)
        a-verifier
                (> VAL 'petit)
        import   0.6

    habitat
        type      UN IND HABITAT
        import   0.4

    regime
        type UN IND REGIME
        import   0.4

    assertions
        type LISTE D SYMBOLE
        import   0.4

    unite-de-mesure
        type     UN D UNITE
```

```
        si-besoin
            (INTERROGER-UTILISATEUR !unite-de-mesure)

        a-verifier
                  (not(null VAL))
        import    0.5

    domaine-de-variation
        type      UN D INTERVALLE
        defaut    [0, grand]
        si-besoin
            (INTERROGER-UTILISATEUR !domaine-de-variation)
        a-verifier
                  (>= (borne-sup VAL) 'grand)
        import    0.5

    structure
        type UN D SYMBOLE
        domaine  (homogene, avec-classes-d-age)
        si-besoin
            (INTERROGER-UTILISATEUR !structure))


(VARIABLE-FORCANTE-BIOLOGIE

    sorte-de
        $valeur  OBJET-ELEMENTAIRE
    spec
        $valeur  (TEMPERATURE ECLAIREMENT)

    nom
        type UN D SYMBOLE
        import    1.0

    ordre-de-grandeur
        type UN D ORDRE-DE-GRANDEUR
        defaut    grand
        si-besoin
            (INTERROGER-UTILISATEUR  !ordre-de-grandeur)
           import   0.6
```

```
assertions
     type UN D SYMBOLE
     import   0.4

representation-math
     type UN IND VARIABLE-D-ENTREE-MATH
     import   0.5

domaine-de-variation
     type      UN D INTERVALLE
     defaut    [0, grand]
     si-besoin
        (INTERROGER-UTILISATEUR  !domaine-de-variation)
     import   0.5

unite-de-mesure
     type      UN D UNITE
     si-besoin
        (INTERROGER-UTILISATEUR !unite-de-mesure)
     import   0.5
```

Les processus biologiques sont à leur tour définis comme autres objets sémantiques.

```
(PROCESSUS
    sorte-de
        $valeur   OBJET-SEMANTIQUE

    spec
        $valeur  (MORTALITE
             PREDATEUR-PROIE
             COMPETITION
             CROISSANCE)


    nombre-d-objets
        type UN D ENTIER
        a-verifier
                 (> VAL 0.0)
        import   0.6
```

```
type-des-objets
     type LISTE IND OBJET-ELEMENTAIRE
     domaine (VARIABLE-D-ETAT-BIOLOGIE
              VARIABLE-FORCANTE-BIOLOGIE)
     import   0.5

nom-du-processus
     type UN D SYMBOLE
     import   0.5

graphe
     type UN IND PROCESSUS
     import   0.5)


(PREDATEUR-PROIE
    sorte-de
          $valeur    PROCESSUS

    nombre-de-predateurs
         type UN D ENTIER
         import   0.6

    nombre-de-proies
         type UN D ENTIER
         import   0.6

    autolimitation
         type UN D BOOLEEN
         import   0.4)
```

Enfin, des schémas définissent des situations de travail en rassemblant des objets élémentaires et processus, c'est à dire l'essentiel des informations sémantiques.

```
(SITUATION
    sorte-de
          $valeur    OBJET-SEMANTIQUE

    spec
```

```
            $valeur     (DYNAMIQUE-D-UN-OBJET
                        PROCESSUS+OBJETS)

      objets-elementaires
            type LISTE IND OBJET-ELEMENTAIRE
            import   0.5

      nombre-d-objets
            type UN D ENTIER
            si-besoin
                        (compter !objets-elementaires)
            import   0.5

      nombre-de-processus
            type UN D ENTIER
            si-besoin
                        (compter !processus)
            import   0.5

      processus
            type LISTE IND PROCESSUS
            import   0.5

      variables-d-etat
            type LISTE IND VARIABLE-D-ETAT-BIOLOGIE
            import   0.5

      variables-forçantes
            type LISTE IND VARIABLE-FORCANTE-BIOLOGIE
            import   0.5)


(PROCESSUS+OBJETS
      sorte-de
            $valeur     SITUATION

      spec
            $valeur     (INTERACTION1
                        INTERACTION2)
```

```
    vitesse-qualitative-du-processus
        typeUN D VITESSE
        import   0.5

    representation-graphique
        typeUN IND OBJET-PROCESSUS-GRAPHIQUE
        import   0.4)


(INTERACTION1
    sorte-de
          $valeur    PROCESSUS+OBJET
    spec
        $valeur  (EXPONENTIEL
            GOMPERTZ
            LOGISTIQUE
            SATURATION
            EN-CLOCHE
            LINEAIRE)

    caracterisation
        type     UN D SYMBOLE
        a-verifier
                 (equal (compter !variables-d-etat) 1.0)
        import   0.5

    croissance
        type UN D SIGNE
        import   0.5

    ordre-de-grandeur
        type UN D ORDRE-DE-GRANDEUR
        import   0.5

    parametres
        type     LISTE IND PARAMETRE
        import   0.5

    coefficients
        type LISTE IND COEFFICIENT
```

```
        import    0.5

    influence-des-variables-forçantes
        type LISTE IND COURBE-DE-REPONSE
        import    0.5

    formalisme
        type UN IND UNIVERSEL
        domaine (FLUX REACTION-CHIMIQUE)
        import    0.5

    expression-math
        type UN IND EXPRESSION-MATH
        import    0.5

    dynamique-de-x
        type UN IND EXPRESSION-MATH
        defaut    (* !croissance !expression-math)
        import    0.4)


(INTERACTION2
    sorte-de
            $valeur    PROCESSUS+OBJET

    spec
        $valeur  (PREDATEUR-PROIE
             BILINEAIRE
             MICHAELIENNE)
    caracterisation
        type      UN D SYMBOLE
        a-verifier
                  (equal (compter !variables-d-etat) 2)


    lineaire-en-x1
        type      UN D BOOLEEN
        import    0.5

    lineaire-en-x2
```

type UN D BOOLEEN
import 0.5

signe-de-l-interaction-de-1-sur-2
type UN IND SIGNE
import 0.5

importance-de-cette-interaction-1
type UN D IMPORTANCE
defaut faible
import 0.5

signe-de-l-interaction-de-2-sur-1
type UN D SIGNE
impoprt 0.5

importance-de-cette-interaction-2
type UN D IMPORTANCE
defaut faible
import 0.5

parametres
type LISTE IND PARAMETRE
import 0.5

coefficients
type LISTE IND COEFFICIENT
import 0.5

influence-des-variables-forçantes
type LISTE IND COURBE-DE-REPONSE
import 0.5

formalisme
type UN IND UNIVERSEL
domaine (FLUX REACTION-CHIMIQUE)
import 0.5

expression-math-de-la-vitesse
type UN D EXPRESSION-MATH
a-verifier

```
                (> VAL 0.0)

dynamique-de-x1
     type UN D EXPRESSION-MATH
     defaut   (* !signe-de-l-interaction-de-2-sur-1
              !expression-math-de-la-vitesse)
     import   0.5

dynamique-de-x2
     type     UN D EXPRESSION-MATH
     defaut   (* !signe-de-l-interaction-de-1-sur-2
                !expression-math-de-la-vitesse)
     import   0.5)
```

Cette liste d'objets du niveau sémantique n'est évidemment pas exhaustive. Beaucoup de schémas utilisés pour typer des attributs n'ont pas été définis tels que: BOOLEEN, UNITE, HABITAT, REGIME, INTERVALLE et d'autres encore. Mais dans tous les cas, les noms de schémas et d'attributs ont été choisis pour être significatifs et faciliter la lecture et compréhension des schémas.

Des réflexes sont utilisés , en particulier une fonction d'interrogation vers l'utilisateur apparaît souvent dans la facette "si-besoin".

Des univers locaux sont utilisés: ORDRE-DE-GRANDEUR, VITESSE et IMPORTANCE. Dans le premier, les valeurs floues définies sont: petit, appréciable, grand et très-grand. Ces valeurs seront définies numériquement comme cela a été décrit au chapitre précédent. Les échelles choisies varieront avec le phénoméne étudié et le contexte de travail.

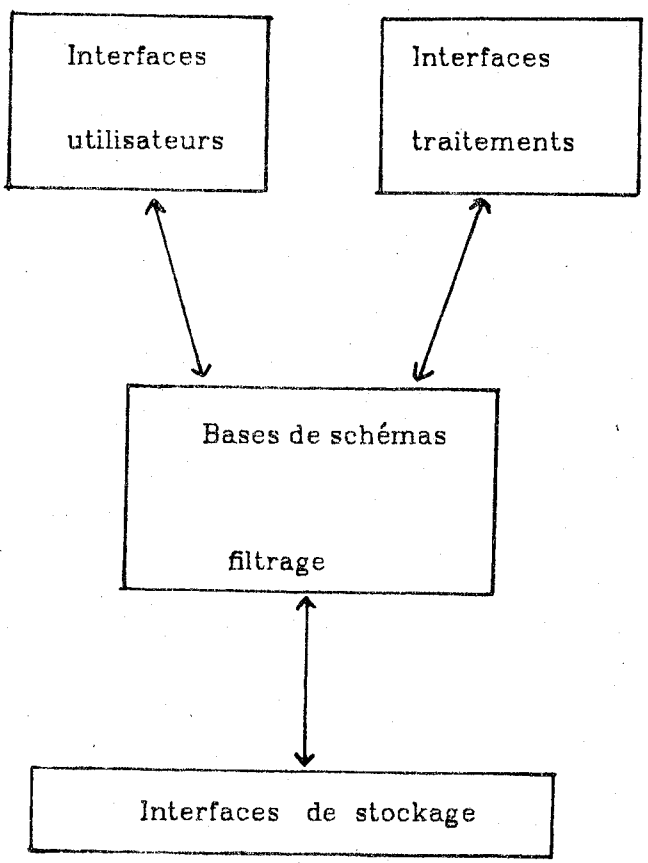
L'état d'avancement actuel de la formalisation des connaissances dans le projet EDORA permet de confirmer la réelle utilité des termes du langage naturel dans la description des schémas. Cela est en particulier vérifié au niveau sémantique de description. La pratique et des expèriences futures permettront d'améliorer le codage et les outils utilisés.

III/ Réalisation de la maquette

L'étude faite est de nature trés générale. Le sujet n'était pas de spécifier et d'implémenter une représentation des connaissances et un mécanisme d'exploitation pour une application spécifique, telle que la modélisation mathématique. Le système informatique réalisé doit pouvoir être utilisé dans divers domaines, comme celui de l'aide à la modélisation mathématique, où la formalisation des connaissances est à faire avec les notions d'objets.

1/ Schéma d'ensemble du système

Le schéma d'ensemble du système informatique est centré autour d'une base d'objets, seul moyen de communications :



Les interfaces utilisateurs, non encore développées, devront permettre l'utilisation de langages spécialisés différents, propres à chaque classe d'utilisateurs.

Les interfaces traitements gèrent les bibliothéques d'algorithmes. Les principaux problèmes sont techniques et concernent l'interfacage entre différents environnements.

Le système de stockage des individus, tels qu'ils ont été définis au chapitre III, est constitué d'un système de gestion de base de données relationnelle écrit en LeLisp (Bensaid 84).

Notre propos ici concerne uniquement la base d'objets et plus précisément les mécanismes de filtrage permettant son exploitation.

2/ Les langages de programmation utilisés

La maquette, comportant la représentation des connaissances par schémas ainsi que les mécanismes d'exploitation associés, ont été réalisés avec LeLisp (Chailloux 83) comme langage de base. En tant que langage fonctionnel, interprété, interactif et souple, il facilite les étapes de développement et de mise au point. Les manipulations de symboles sont aisées. De plus LeLisp est lisible, puisqu'il posséde toutes les structures de contrôle habituelles, et il est l'un des dialectes Lisp les plus performants.

Pour construire le langage de spécification des schémas, nous avons aussi utilisé CEYX, un langage orienté objets intégré à LeLisp. Les performences de CEYX (Hullot 84), sa définition comme langage orienté objets et le fait qu'il soit intégré à LeLisp nous ont paru intéressants. De plus, nous espérions ainsi pouvoir utiliser les techniques d'héritage entre objets propres à ce langage.

3/ Présentation de CEYX

a/ Introduction

Nous ne donnons qu'une bréve présentation du langage CEYX, plus précisément décrit dans (Hullot 84).

On retrouve en CEYX les notions d'objets, de classes, de transmissions de messages et d'héritage. Un objet se caractérise par une structure de type "record" Pascal. Il possède un certain nombre de champs (ou attributs) dans lesquels on peut stocker des informations sur l'objet. Il possède aussi des méthodes appelées en CEYX "sémantiques" , définissant les comportements possibles de l'objet. Un objet est toujours l'instance d'une classe. Les classes peuvent être organisées selon une hiérarchie suivant laquelle champs et sémantiques sont hérités.

b/ Définition d'une classe

On définit une classe par son nom et un certain nombre de champs:

(deftclass <classe> <champ 1> ... <champ n>)

On peut définir une sous-classe d'une classe déjà existante:

(deftclass {<sur-classe>}:<classe> <champ 1> ... <champ n>)

Dans ce cas, les champs <champ 1> ... <champ n> représentent de nouveaux champs qui s'ajoutent à ceux hérités de la <sur-classe>.

Exemple:

```
? (deftclass Modele nom modele-math)
= #:Tclass:Modele

? (deftclass {Modele}:Modele-physique masse-ressort
```

```
?                          loi-physique
?                          force-friction
?                          force-tration)
= #:Tclass:Modele:Modele-physique
```

On définit ainsi la classe Modele qui possède les champs "nom", "modele-math". La classe Modele-physique est une sous-classe de la première. Elle possède aussi les champs "nom" et "modele-math" avec en plus les champs "masse-ressort", "loi-physique", "force-friction" et "force-traction".

c/ Création d'instances

Un objet représentant d'une classe est créé avec la fonction omakeq:

(omakeq <classe> <champ 1> <valeur 1> ... <champ n> <valeur n>)

ce qui permet aussi d'affecter la valeur <valeur i> au champ <champ n>.

Exemple:

```
? (setq M1 (omakeq Modele-physique nom 'M1
?                  masse-ressort 'nulle
?                  loi-physique 'Hooke
?                  force-friction 'lineaire
?                  force-traction 'nulle))
= #(#:Tclass:Modele:Modele-physique . #[M1 () nulle Hooke lineaire nulle])
```

d/ Fonctions d'accès aux champs

On accède aux champs de l'objet à l'aide de fonctions d'accès créées automatiquement lors de la définition de la classe. Ces fonctions permettent de consulter et de modifier la valeur d'un champ.

Exemple:

de consultation:

```
? ({Modele-physique}:masse-ressort  M1)
= nulle
```

d'affectation:

```
?({Modele-physique}:force-traction  M1  'F)
= F

? ({Modele-physique}:force-traction  M1)
= F
```

e/ Définition de sémantiques

Les sémantiques définissent le comportement d'un objet à la réception d'un message donné. On définit une sémantique comme une fonction LISP de nom <classe>:message.

(de {<classe>}:<message> (objet arg1 ... arg n)
...)

La sémantique <message> est alors définie pour tout objet de la classe <classe>. Les arguments sont composés de l'objet receveur et des paramètres facultatifs.

On envoie un message à un objet avec la construction "send".

(send <message> <objet> <arg 1> ... <arg n>)

Exemple:

```
? (de  {Modele-physique}:alourdir (modele  masse)
?      (print  "Masse  nouvelle :  "
?          ({Modele-physique}:masse-ressort  modele  masse)))
= #:Tclass:Modele:Modele-physique:alourdir
```

On définit ainsi la sémantique "alourdir" avec un argument "masse" qui change

la masse du ressort dans le modèle-physique voulu.

L'envoi de message se fait par:

```
? (send 'alourdir M1 '50g)
Masse nouvelle : 50g
= 50g

? ({Modele-physique}:masse-ressort M1)
= 50g
```

f/ Mécanisme d'héritage

Un objet d'une classe hérite des champs définis au niveau des sur-classes et de toutes les sémantiques. Seul le mono-héritage est possible. L'objet sait donc répondre à tous les messages que peuvent recevoir les objets des classes hiérarchiquement supérieures, en plus des messages attachés à sa propre classe. Pour cela, la construction "send" effectue une recherche, en profondeur d'abord, dans l'arbre des classes pour trouver la sémantique à activer.

4/ La représentation implémentée

Le langage de spécification implémenté à l'aide de LELISP et CEYX permet de construire des schémas tels qu'ils ont été définis au chapitre III.

Il est apparu quelques restrictions dues au langage CEYX.

Du point de vue descriptif, il est possible de construire des schémas à trois niveaux d'imbrication avec des classes et instances de CEYX. Mais, on doit pouvoir affecter une valeur à un attribut dans n'importe quel schéma de la hiérarchie. Or, on ne peut en CEYX affecter une valeur qu'aux champs d'une instance et non pas d'une classe. Nous avons donc dû distinguer les affectations d'attributs dans un concept et un individu. Les implémentations utilisées selon les cas sont différentes.

Nous n'avons pas ou peu utilisé les sémantiques de CEYX, puisque contrairement au langages orientés objets, nous n'utilisons pas de méthodes mais des réflexes qui permettent de distribuer la dynamique sur les attributs. L'activation des réflexes est gérée par les fonctions d'accès, certaines étant définies à l'aide de sémantiques.

La principale restriction concerne la dynamique de la base d'objets. CEYX n'offre qu'une organisation statique des classes et instances. Lors de la définition d'une classe (deftclass) ou d'une instance (omakeq), toutes les informations héritées sont physiquement dupliquées dans le nouvel objet. Les recherches d'informations héritées sont ainsi évitées ce qui améliorent les performances. Mais nous avons dû ajouter un mécanisme de mise à jour qui permette la propagation des modifications dans tous les schémas hiérarchiquement inférieurs à celui qui a été modifié. De même, un mécanisme de recherche dans les ancêtres a été réalisé pour compléter le tout.

Par contre, il est impossible de réorganiser la base d'objets, c'est à dire de modifier les liens hiérarchiques, à moins de regénérer toute la hiérarchie. La base n'est donc pas dynamique comme nous l'avons décrite au chapitre III.

Ce problème nous a aussi empêché d'implémenter complétement le multi-héritage décrit précédemment. Un multi-héritage a pu être codé à partir du mono-héritage offert. Mais l'aspect statique de ce dernier outil empêche de redéfinir et de compléter la description des attributs hérités. Le multi-héritage au niveau des facettes n'a pas pu être réalisé comme il a été décrit au chapitre III. Seul un multi-héritage au niveau des attributs a été réalisé.

La combinaison des valeurs floues lors de l'héritage a par contre pu être programmée. Le système génère dans certains cas d'héritage de nouvelles valeurs floues à partir des définitions existantes, selon les techniques de combinaisons exposées au chapitre IV. Le vocabulaire dans les univers locaux s'accroît ainsi de façon autonome.

Les notions de vues différentes d'un même individu sont aussi utilisables.

Cet essai de programmation avec un langage orienté objets confirme la différence, mise en valeur au chapitre II, entre les langages orientés objets et les représentations centrées objets. LELISP seul permettra par contre la programmation complète de la représentation centrée objets décrite et de son mécanisme d'exploitation.

5/ Le filtrage réalisé

Le processus de filtrage a été réalisé en LELISP, de façon modulaire et indépendante de l'implémentation de la représentation des connaissances. Le filtrage ne manipule les schémas que par un nombre limité de primitives qui sont à modifier selon l'implémentation de la représentation.

Toutes les fonctionnalités décrites au chapitre IV ont été codées. Des termes flous peuvent être définis, manipulés et combinés pour le multi-héritage.

Les réflexes cités sont utilisables. Ils ne peuvent avoir pour valeur que des appels de procédure. Des filtres peuvent être donnés dans les facettes "$valeur" en utilisant les fonctions système "trouver" et "trouver-traitement".

Le processus comporte trois étapes. La première est le filtrage hiérarchique; la seconde, le filtrage structurel est fondée sur le type des attributs. La dernière est le filtrage atomique implémenté ainsi qu'il a été décrit au chapitre IV.

Le filtrage atomique avec des valeurs absentes (à nil en LISP) a été implémenté comme suit:

```
si deux valeurs nil alors distance = 1
sinon
   si une valeur à nil alors distance = 0.5
   sinon
      filtrage atomique classique décrit au chapitre IV
```

Dans un premier temps, lors du filtrage en profondeur, la structure du filtre ne peut pas évoluer. Le filtre ne peut être complété que par l'apport de valeurs pour les attributs présents dans sa structure. Permettre l'évolution structurelle du filtre obligerait en CEYX à regénèrer un nouveau filtre à chaque étape de filtrage à plat.

Afin de faciliter la réalisation de la maquette, la notion d'identifiabilité entre types a été simplifiée dans un premier temps et réduite à l'égalité entre noms de types. Les distances entre types sont donc binaires, 1 pour une égalité de noms et 0 sinon. Mais ceci diminue peu les nuances du raisonnement global et conserve toute son importance à la notion de typage des attributs.

CONCLUSION

Les notions d'objets sont de plus en plus répandues dans toute l'informatique grâce à leur aspect déclaratif et à leur puissance d'expression.

Les représentations centrées objets permettent de représenter les connaissances de façon explicite, modulaire et modifiable. Les avantages de cette représentation sont essentiellement relatifs aux notions d'organisation des connaissances, de contexte et de contrôle du raisonnement.

Notre travail a consisté tout d'abord en l'amélioration de la puissance d'expression de cette représentation en permettant l'utilisation de termes du langage naturel. L'essentiel de la contribution apportée concerne la spécification d'un mécanisme d'exploitation de la représentation à base de filtrage flou. Ainsi, la manipulation des objets est plus souple et fondée sur leur sémantique. Divers types de raisonnements peuvent être utilisés.

Mais tous les aspects de la représentation centrée objets n'ont pas été développés ici. Il manque en particulier l'étude des principes d'interaction avec l'utilisateur. Ces problèmes doivent être étudiés surtout au sujet des explications que le système doit pouvoir fournir et au sujet des facilités de descriptions et de modifications des schémas.

La conception et l'utilisation dans divers domaines du logiciel CRIQUET, conçu comme un outil de base pour élaborer des systèmes experts sous forme de systèmes de production, a permi de contacter divers utilisateurs et ainsi de concevoir le type de logiciel nécessaire.

L'état de l'art dépasse déjà les systèmes de production classiques. Beaucoup de systèmes permettent aussi, en plus des règles, la manipulation d'une base d'objets à la place d'une simple base de faits sans organisation. En effet, même sans expèrience, les utilisateurs réclament en premier lieu des possibilités de construction d'une base de connaissances organisée.

Nous pensons qu'une évolution logique doit amener les logiciels de ce type a' utiliser une représentation des connaissances et un mécanisme d'exploitation comme ceux décrits dans cette thèse, c'est à dire ne manipulant qu'une seule notion d'objets, sans règles ou autres artifices.

Une première utilisation de ce système dans le projet EDORA devra permettre de déterminer si la représentation retenue est naturelle pour les experts et le mécanisme d'exploitation suffisant. Ce critère, qui ne peut pas être estimé a priori, est extrémement important sachant qu'à l'heure actuelle le transfert de connaissances de l'expert vers le système informatique constitue le goulot d'étranglement dans le développement des systèmes experts.

Si les premières tentatives se révèlent fructueuses, cette approche pourra être développée et utilisée dans d'autres domaines. Il s'agit, comme le montre le projet EDORA, en particulier de domaines pour lesquels la connaissance procédurale abonde sous formes de méthodes, mais pour lesquelles le mode d'emploi n'existe pas par manque d'expertise.

Un langage d'expression et un mécanisme d'exploitation souple comme nous les avons définis peuvent aussi être utilisés pour le traitement des langues naturelles.

Nous nous sommes aussi intéressés aux rapprochements entre les études sur les représentations centrées objets et les études menées en vue de spécifier des systèmes dotés de la faculté d'apprendre. Ce problème est étudié depuis longtemp en psychologie, mais il est loin d'être résolu. Sans prétendre apporter de solutions immédiates, il nous semble que la poursuite de ces études peut être facilitée en manipulant des représentations centrées objets et des mécanismes plus souples et nuancés tels que celui qui a été présenté dans cette thèse.

Ce sujet d'étude constitue un objectif ambitieux, mais qui parait être une juste finalité dans l'étude des systèmes intelligents.

BIBLIOGRAPHIE

Bibliographie

ACSL
Advanced Continuous Simulation Language
user/guide reference manual
Mitchell and Gauthier, Associates 1975 USA

Ackley DH,Berliner HJ
"The QBKG system : Knowledge Representation for Producing and Explaining Judgements"
Rapport CMU-CS-83-116 (63 p)
Department of Computer Science
Carnegie-Mellon University 1983

a/ Aikins J
"Prototypical Knowledge for Expert Systems"
Artificial Intelligence 20 (1983) 163-210

b/ Aikins J,Kunz JC,Shortliffe E
"PUFF : An Expert System for interpretation of pulmonary function data"
Computer and Biomedical Research 16,(p 199-208), 1983.

Aikins J
"Prototypes and production rules. An approach to knowledge representation for hypotheses formation"
Stanford Heuristic Programming Project
HPP 79-10 (working paper) july 1979

Albert P
"KOOL : représentation des connaissances"
BIGRE Journées d'étude sur les langages orientés objets
Décembre 1983

Allen E
"YAPS : a production rule system meets objects"
AAAI- 83, August 22-26 1983
Washington DC (p 5-7)

Arons Swaan H
"Expert systems in the simulation domaine "
Mathematics and computers in simulation XXV, 1983 (p10-16)
North Holland Publishing Company

Ayel M,Laurent JP
"CESSOL : un système expert pour définir des campagnes de reconnaissance géotechnique du sol"
4ième congrés, Reconnaissances des formes et Intelligence Artificielle
Paris 25-27 janvier 1984

Barbuceanu M
"Objet-Centered Representation and reasoning : an application to computer-aided design "
Sigart New Letters, n 87 Janvier 1984

Barr A,Feigenbaum E
"The Handbook of Artificial Intelligence "
Volume I, II, III
Pitman 1981

Bensaid A
"SHIMER : un système relationnel en LISP"
Rapport de recherche IMAG, Laboratoire ARTEMIS
rr n 464 Aout 1984

a/ Bobrow DG,Stefik M
"The LOOPS Manual "
Knowledge-Based VLSI Design Group
Memo KB-VLSI-81-13 (working paper)
Xerox Corporation 1982

b/ Bobrow DG,Stefik M
"Rule-oriented Programming in LOOPS"
Knowledge-Based VLSI Design Group
Memo KB-VLSI-82-22 (working paper)
Xerox Corporation 1982

c/ Bobrow DG,Winograd T
"An overview of KRL,a Knowledge Representation Language "
Cognitive Science Voln=1 1977

d/ Bobrow DG, Kaplan RM, Kay M, Norman DA, Thompson H, Winograd T
"GUS : A Frame-Driven Dialog system"
Artificial Intelligence 8, 1977, (p 155-173)

a/ Bonnet A
"Quelques modes de représentation des connaissances et des mécanismes de raisonnement pour les systèmes experts "
4ième congrés, Reconnaissances des formes et
Intelligence Artificielle
Roquencourt 23-24 janvier 1984

b/ Bonnet A
"L'utilisation des langages orientés objets (LOOBS) en Intelligence Artificielle "
Les systèmes experts et leurs applications
Journées d'étude et exposition
Avignon 2,3 et 4 Mai 1984

Bonnet A,Harry J,Ganascia JG
"LITHO ,un système expert inférant la géologie du sous sol "
TSI Technique et Science Informatique ,vol 1,n=5 ,1982

a/ Brachman RJ
"What ISA is and isn't. An analysis of taxonomic links in semantics networks "
Computer October 1983
IEEE Computer Society

b/ Brachman RJ,Levesque HJ
"Krypton : A functional approach to knowledge representation"
Computer October 1983
IEEE Computer Society

Briot JP
"L'instanciation dans les langages objets "
BIGRE Journées d'étude sur les langages orientés objets
Decembre 1983 (p 173-209)

Briot JP, Serpette BP
"Le langage FORMES, analyse et implémentation"
Mémoire de stage de DEA
Paris VI Juillet 1982

CERT
"Implantation d'un système déductif sur une base de données relationnelle "
Rapport intermédiaire n= 1/3/63/DERI/
Novembre 1981

CSMP
Continuous System Modelling Program III
Program reference manual 1972
IBM Canada Limited, Program Product Centre, Ontario

Carbonell JG, Michalski RS, Mitchell TM
"An overview of machine learning"
In Machine learning : an artificial intelligence approach
Michalski RS, Carbonell JG, Mitchell TM (editors)
Tiogo Press, Palo Alto, CA, 1983

Cayrol M,Farreny H,Prade H
"Fuzzy pattern matching"
Kybernetes vol 11,1982,(p 103-116)

Chailloux J
"LE-LISP de l'INRIA "
Le manuel de reference
Rapport INRIA Novembre 1983

Charniack, Riesbeck, Mc Dermott
"Artificial Intelligence Programming"
Lawrence Erlbaum Associates, Hillsdale, New Jersey 1983

Cholvy L, Foisseau J
"ROSALIE : un système de manipulation d'objets techniques
basé sur des règles "
IFIP 1983

Chouraki E
"Sur un réseau sémantique actif"
Proc Colloque Saint Maximin IRIA-LISH
Saint Maximin 1979 (p 215 221) ed INRIA

Cohen BL
"Powerful and efficient structural pattern recognition system"
Artificial Intelligence vol 9, n= 3, Décembre 1977 (p 223-255)

Cointe P
"Une réalisation de SMALLTALK en VLISP"
TSI Technique et Science Informatiques
vol 1,n=4,1982

Colmerauer A,Kanoui H,Van Caneghem M
"Prolog, bases théoriques et développements actuels"
TSI Technique et science informatiques
vol 2,n=4,1983

a/ Cordier MO,Rousset MC
"TANGO : moteur d'inférence pour un système expert avec variables"
Proc 4ième congrés AFCET-INRIA
Reconnaissances des formes et Intelligence Artificielle
25-27 janvier 1984

b/ Cordier MO
"Les systèmes experts "
La Recherche n=151 Janvier 1984

Dahl V
"Logic programming as a Representation of knowledge "
Computer October 1983
IEEE Computer Society

Dalenoort GJ
"Some general considerations on representation of informational systems and processes "
Représentation des connaissances et raisonnement dans les sciences de l'homme
Colloque de St Maximin pub IRIA, 17-19 Septembre 1979

Davis R
"Meta rules : reasoning about control "
Artificial intelligence
Décembre 1981-5- (p 179-222)

Davis R,Buchanan BG
"Meta-level knowledge : overview and applications"
Proc IJCAI-77 1977 (p 920-927)

Delobel C,Adiba M
"Bases de données et systèmes relationnels "
DUNOD 1982

Demonchaux E,Quinqueton J,Ralambondrainy H
"Une application de l'intelligence artificielle à l'analyse des données "
Les systèmes experts et leurs applications
Journées d'étude et exposition
Avignon 2,3 et 4 Mai 1984

Descottes Y
"Représentation et exploitation des connaissances expertes en génération de plan d'actions"
These de 3ième cycle,Grenoble décembre 1981

a/ Dincbas M
"A knowledge-based expert system for automatic analysis and synthesis in CAD "
Information Processing 80
SH Lavington
North Holland Publishing Company 1980

b/ Dincbas M
"Le système de résolution de problèmes METALOG"
CERT, Rapport final n 3146/DERI
Decembre 1980

Dormoy JL
"Représentation des connaissances en physique et systèmes experts "
Les systèmes experts et leurs applications
Journées d'étude et exposition
Avignon 2,3 et 4 Mai 1984

Mc Dermott,Doyle J
"Non monotonic logic I"
Artificial Intelligence 1980 vol 13,n=1,2 pp 41-72

Doyle J
"Admissible State Semantics for representation Systems "
Computer October 1983
IEEE Computer Society

Doyle J
"A truth maintenance system"
Artificial Intelligence 12 (1979) 231-272

a/ Dubois D, Prade H
"The managing of uncertainty in fuzzy expert systems and some applications "
in The analysis of fuzzy information
CRC Press 1985 (a paraitre)

b/ Dubois D, Farreny H, Prade H
"Suivi d'exécution d'itinéraires spécifiés en termes imprécis"
Colloque international d'Intelligence Artificielle
Marseille 24 27 Octobre 1984
Tome 1 (p 101-115)

c/ Dubois D, Prade H
"On distance between fuzzy points and their use for plausible reasoning"
Proc IEEE Int, Symp Cybernetics et Society
New Dehli Bombay Decembre 30,1983,n=7

d/ Dubois D,Prade H
"Fuzzy sets and systems : theory and applications"
vol 144 in Mathematics in science and engineering
Academic Press 1980

Dufresne P
"Contribution algorithmique à l'inférence par règles de production"
Thèse Docteur Ingénieur
Université Paul Sabatier de Toulouse (Sciences)
29 Juin 1984

Engelmore R,Terry A
"Structure and function of the Crysalis system "
IJCAI 1979 (p 250-256)

Falzon P
"Understanding a technical language ; A schema-based approach"
Rapport de Recherche INRIA n 237, Octobre 1983

Farley A
"A probabilistic model for uncertain problem solving "
IEEE vol 13,4,july,/august 1983

Farreny H
"Un système de maintenance automatique d'inter-relations dans un système de production"
Proc Reconnaissance des formes et Intelligence Artificielle
AFCET Nancy 1981

Ferber J
"MERING : un langage d'acteurs pour la représentation des connaissances et la compréhension du langage naturel"
4ième congrés, Reconnaissances des formes et Intelligence Artificielle
Paris 25-27 janvier 1984

a/ Ferber J
"Représentation des connaissances "
Micro-Système Octobre 1983

b/ Ferber J
"Intelligence artificielle et LOGO. Gérer le déroulement du raisonnement"
Micro-Systeme Avril 1983

Fieschi M
"Intelligence Artificielle en médecine. Des systèmes experts"
Masson, Paris 1984

Forgy C.L
"The OPS83 Report"
Department of Computer Science
Carnegie-Mellon University
Report CMU-CS-84-113
May 1984 (51 pages)

Forgy C.L
"RETE : A fast algorithm for the many pattern many object pattern match problem"
Artificial Intelligence
19 (1982) (p 17-37)

Fraisse P, Piaget J
"Traité de psychologie expérimentale IX"
Presses Universitaires de France 1969

Gallaire H
"Impact of logic on data bases "
IEEE 1981 p248-259

Gallaire H,Lasserre C
"Contrôle par méta règles d'un processus de dérivation logique"
Actes du congrés AFCET,TTI Informatique
Editions Hommes et Techniques,Tome,1978

Ganascia JG
"Etude des contradictions entre les données dans les
systèmes de diagnostic"
Proc 4ième journées congrés AFCET-INRIA janvier 1984

Georgeff M
"A framework for control in production systems"
Proc VI IJCAI
Tokyo August 1979 (p 328-334)

Georgeff M
"Procedural expert systems"
Technical notes 314
December 9 1983
Artificial Intelligence
Computer Science and Technology Division
Department of Computer Science
Monash University Clayton, Victoria, 3168 Australia

Gevarter WB
"An overview of expert systems "
Rep NBSIR 82;2505 National Bureau of Standards
US Department of Commerce (May 1982)

Giles R
"A computer program for fuzzy reasoning"
Fuzzy Sets and Systems 4, 1980, (p 221-234)
North-Holland Publishing Company

Goldberg A,Robson D
"Smalltalk 80. The language and its implementation "
Addison Wesley Publishing company 1983

Goldstein I,Robert B
"Using Frame in scheduling "
in Artificial Intelligence
an MIT perspective, volume 1
Winston and Brown editors
MIT Press 1980

Gomez C,Quadrat JP,Sulem A
"Vers un système expert en contrôle stochastique :
interface en langage naturel "
Les systèmes experts et leurs applications
Journées d'étude et exposition
Avignon 2,3 et 4 Mai 1984

Gondran M
"Introduction aux systèmes experts"
Eyrolles 1983

Goodwin J
"Taxonomic programming with KL-One "
Tech Rapport LiTH-MAT-R-79-5
Informatics Laboratory
Linkoeping University,Sweden 1979

Gordon S, Novak JR
"GLISP User's manual"
Heuristic Programming Project
Report No HPP-82-1
Computer Science Department
Report No STAN-CS-82-895
Stanford University January 1982

a/ Gouze JL,Vignard P
"EDORA : un système intelligent d'aide à la modélisation
en biologie"
International 84 AMSE Conference Modelling and Simulation
Athen Greece June 27-29 1984

b/ Gouze JL, Vignard P
"Intelligence Artificielle et modélisation en biologie"
Colloque COGNITIVA
Paris 4, 5 et 6 Juin 1985

c/ Gouze JL, Rechenmann F, Vignard P
"EDORA : an artificial intelligence approach to dynamic system modelling"
The management and modelling of dynamic systems
Bruges, Belgium june 12-14 1984

a/ Granger C, Thonnat M, Vignard P
"Etude d'un système expert pour la classification de galaxies : SYGAL"
Les systèmes experts et leurs applications
Journées d'étude 2,3 et 4 Mai 1984 Avignon

b/ Granger C
"Symbolic scene matching"
Seventh International Conference on Pattern Recognition
Montreal Canada, July 30 - August 2 1984

Granger C
"Evaluation des performances du langage PLASMA en Intelligence Artificielle"
Rapport ENSEEIHT 1982

Groner G, Clark R, Berman R, Deland E
"BIOMOD : an interactive computer graphics system for modeling"
AFIPS Conference Proceeding, 39, (p 369-377) 1971

Hamrouni MK
"Etude et développement d'un système informatique d'aide à l'élaboration de modéles en biologie "
Thèse de 3ième cycle
Université Pierre et Marie Curie 1979

Hayes P
"The logic of frames"
in Reading in Artificial Intelligence (p 451-458)
Ed Webber BL, Nilson NJ
Tioga Publishing Company Palo Alto California 1981

Hayes-Roth F
"The knowledge based expert system : a tutorial"
Computer IEEE
Septembre 1984 (p 11-28)

Hendrix CG
"Partitionned network for the mathematical modeling of natural language semantics"
TR-NL-28
Dept of Computer Sciences
Univ of Texas Austin 1975

Himbaut S,Guillou G,Quinton JC
"AMEDIA : système expert en diagnostic de panne pour un réacteur nucléaire a neutrons rapides"
Les systèmes experts et leurs applications
Journées d'étude et exposition
Avignon 2,3 et 4 Mai 1984

Hollander C,Iwasaki Y
"The Driller Advisor : An Expert system Application"
26 th Computer Society International Conference
COMPCON String 83, IEEE 1983, (p 116-119)

Huet G, Vincent D
"ROSACE : Un outil de représentation des connaissances sous forme d'objets et d'actions. Notice de présentation et d'utilisation"
Rapport CNET/LAA/SLS/AIA le 01.10.84

Hullot JM
"CEYX"
Version 4, Le manuel de référence
Note technique INRIA, janvier 1984, (33 pages)

Ishizuka M,Fu KS,Yao JTP
"SPERIL : an expert system for damage assessment of existing structures"
6th International Conference on Pattern Recognition
Oct 1982 Munich
IEEE 1982 (p 932-937)

Israel DJ
"The role of logic in knowledge representation "
Computer October 1983
IEEE Computer Society

Kayser D
"Examen de diverses méthodes utilisées en représentation des connaissances"
4ième congrés, Reconnaissances des formes et
Intelligence Artificielle
Paris 25-27 janvier 1984

Kayser D
"Vers une modélisation du raisonnement approximatif "
Proc Colloque "Représentation des connaissances et raisonnement dans les sciences de l'homme" Saint-Maximin, 17-19 Septembre 1979 ed M Borillo pub par INRIA (p 440-457)

Kuipers BJ
"A frame for frame : representation knowledge for recognition"
in "Representing and understanding"
Bobrow BG,Collins A (ed)
Academic Press 1975

Lagache JC
"Présentation d'un système expert pour le code de calcul TITUS "
Les systèmes experts et leurs applications
Journées d'étude et exposition
Avignon 2,3 et 4 Mai 1984

Langley P, Carbonnel JG
"Approaches to machine learning"
Rapport CMU-CS-84-108
Department of computer science
Carnegie-Mellon UIniversity
February 16 1984

Langley P
"Representational issues in learning systems "
Computer October 1983
IEEE Computer Society

a/ Laurent JP
"La structure de contrôle dans les systèmes experts"
TSI vol 9, n=3, 1984

b/ Laurent JP
"Exploitation d'une base de connaissances : choix et stratégies"
4ième congrés, Reconnaissances des formes et
Intelligence Artificielle
Roquencourt 23-24 janvier 1984

a/ Lauriere J L
"Représentation et utilisation des connaissances.
Première partie: les systèmes experts"
TSI Techniques et Sciences Informatiques Vol 1,n=1,1982

b/ Lauriere JL
"Représentation et utilisation des connaissance. Deuxième partie"
TSI Techniques et Sciences Informatiques Vol 1,n=2,1982

Le Ny JF
"La sémantique psychologique "
PUF le psychologue 1979

Lenat DB, Brown J.S
"Why AM and EURISKO appear to work"
Artificial Intelligence
Vol 23, num 3, August 1984 (p 270-294)

Lenat DB
"EURISKO : A program that learns new heuristics and domain concepts "
Artificial Intelligence 21 (1983) 61-98

Lenat DB
"The nature of Heuristics"
Artificial Intelligence 19 (1982) 189-249

Lesser V R,Erman L D
"An experiment in distributed interpretation "
(HEARSAY II)
May 1979, Department of computer science
Carnegie-Mellon University

Luis Farinas del Cerro
"A deduction method for modal logic"
Proc of the 1st ECAI,Orsay,Juillet 1982,pp60-61

Mallet JL, Wild P
"An analogue to correspondence analysis with fuzzy characteristic functions"
Sciences de la Terre 1984

Mangin J C
"Construction d un système expert vue par l'expert : rapports avec l'ingénieur cognitif "
4ième congrés, Reconnaissances des formes et
Intelligence Artificielle
Roquencourt 23-24 janvier 1984

Martin-Clouaire R, Prade H
"Managing uncertainty and imprecision in petroleum geology"
Collloque International
Computer in Earth Sciences for natural resources characterization
Nancy Avril 1984

Mc Calla G,Cercone N
"Approaches to knowledge representation "
Computer October 1983
IEEE Computer Society

Mc Dermott, Doyle J
"Non monotonic logic I"
Artificial Intelligence 1980 vol 13,n=1,2 (p 41-72)

Mc Dermott
"R1 : A rule-based configurer of computer systems"
Technical Report CMU-CS no 80-119, Avril 1980

Minsky M
"A framework for representing knowledge"
in "The psychology of computer vision"
Ph Winston Eds, Mc Graw Hill 1975

Mitchell TM
"Learning and problem solving"
Technical report LCSR-TR-45 (30 p)
June 1983, Laboratory for Computer Science Research
Hill Center for the Mathematical Sciences
Busch Campus. Rutgers University, New Brunswick, New Jersay

Mittal S, Chandrasekaran B, Sticklen J
"Patrec : a knowledge directed database for a diagnostic expert system"
Computer IEEE (p 51-58), September 1984

Mulet-Marquis D, Gondran M
"Un langage pour les systèmes experts : Alouette"
Rapport Electricite De France
Service informatique et Mathématiques appliquées
HI/4773-02 Mai 1984, (63 pages)

Mylopoulos J
"Building Knowledge Based systems : the PSN experience "
Computer October 1983
IEEE Computer Society

Mylopoulos J
"An oveview of knowledge representation"
ACM 1980

NITAS
"SATIN : système multi-expert pour le transfert et l'innovation"
Les systèmes experts et leurs applications
Journées d'étude et exposition
Avignon 2,3 et 4 Mai 1984

Nagel D
"Concept learning by building and applying transformations between object descriptions "
Report LRP-TR-15 (9 p) June 1983
Laboratory for Computer Science Research
Rutgers University,New Brunswick

Nakamura K,Sage A P
"An intelligent data-base interface using psychological similarity between data "
IEEE vol smc 13,n 4,july/august 1983

Newell A
"The knowledge level "
Artificial Intelligence 18 (1982) 87-127

Newell A
"Production systems: models of control structures "
Visual Information Processing WG Chase
ed Academic Press New York 1973

Nicolas JM,Yazdanian K
"Implantation d'un système déductif sur une base de données relationnelle"
Rapport intermédiaire n 1/3163/DERI, CERT nov 1981

Nii HP, Aiello N
"AGE (Attemp to GEneralize) : A knowlwdge based program for building knowledge based programs"
Proc of IJCAI 1979,Tokyo Japan, August 1979, (p645-655)

Nilsson NJ
"Probabilistic logic"
Technical note no 321, 1984 February 6
SRI International

Nilsson NJ
"Principle of Artificial Intelligence"
Tioga Publishing (1980)

Oudet B, Nepomiastchy P
"Le projet MODULECO"
Rapport interne INRIA 1980

Patten B (edited by)
"Systems analysis and simulation in ecology"
Volume I
Academic Press 1971

Pave A,Lebreton JD
"MILADIE : un mini langage d'application pour le traitement numérique d'équations différentielles et de récurrence "
RAIRO Juin 1976 B-2 73-9

Pinson S
"Représentation des connaissances dans les systèmes experts "
RAIRO Informatique Vol 15,n=4,1981, (p 343-367)

Prade H
"Modèle de raisonnement approché pour les systèmes experts "
4ième congrés, Reconnaissances des formes et
Intelligence Artificielle
Paris 25-27 janvier 1984

Prade H
"A synthetic view of approximate reasoning technique "
Proc of the 8th IJCAI,Karlsruhe,Aout 1983, (p 130-136)

Prade H
"Modèles mathématiques de l'imprécis et de l'incertain en vue d'applications au raisonnement naturel "
Thèse d'état ,Toulouse III,June 1982

Pratt V
"A mathematician's view of LISP"
BYTE August 1979

Queinnec C
"Langage d'un autre type : LISP"
Eyrolles 1983

Quilan JR
"Consistency and plausible reasoning "
Proceeding IJCAI 1983 vol 1
August 1983 Karlsrule, West Germany

Raphael B
"The frame problem in problem-solving systems"
Artificial intelligence and heuristic programming (p 159-169)

Rechenmann F
"Shirka : mécanismes d'infèrence sur une base de connaissances centrée objets"
proposée au 5ième Congrés AFCET
Reconnaissance des Formes et Intelligence Artificielle
Grenoble 25-29 Novembre 1985

a/ Rechenmann F,Bensaid A,Granier D
"SHIRKA : des systèmes experts centrés objets"
Les systèmes experts et leurs applications
Journées d'étude et exposition
Avignon 2,3 et 4 Mai 1984

b/ Rechenmann F
"Intelligence Artificielle et construction de modèles dynamiques"
Intelligence Artificielle et Productique
2ième symposium et exposition international 84
20-22 Novembre 1984 Paris

Reggia J A,Nau D S,Wang P
"A new inference method for frame-based expert system"
Proc AAAI 83 (p 333-337)

Reimer U
"A formal approach to the semantics of frame data model"
IJCAI 1983 (p 337-339)

Rich E
"Default reasoning as likelihood reasoning "
Proc AAAI 83 (p 348-351)

Ritchie G.D, Hanna F.K
"AM : A case study in AI methodology"
Artificial Intelligence
Vol 23, num 3, August 1984 (p 247-268)

Rosenberg S
"HPRL : a language for building expert systems "
Proc IJCAI 83, Karlsruhe, Aout 1983, (p 215-217)

Rousset MC
"Etude comparative de deux moteurs d'inférence, OPS et TANGO "
4ième congrés, Reconnaissances des formes et
Intelligence Artificielle
Paris 25-27 janvier 1984

Sandewall E
"An approach to the frame problem and its implementation"
Machine intelligente 7 (p 195-204)
Edinburgh at the university press

Shapiro E,Takeuche A
"Object oriented Programming in concurent prolog"
New Geneation Computing 1, 1983 , (p 25-48)
Ohmsha LTD and Springer-Verlag

Shortliffe E.H, Carlisle Scott A, Bischoff M.B
"ONCOCIN : an expert system for oncology protocol management"
IJCAI-7 1981, (p 876-881)

Shortliffe E, Buchanan BG
"A model of inexact reasoning in medicine "
Mathematical biosciences 23, 1975, (p 351-379)

Siklossy L, Lauriere J.L
"Removing restrictions in relational data base model : an application of problem solving techniques"
AAAI 82, Aout 1982, Pittsburgh (p 310-313)

Spriet J.A, Vansteentiste G.C
"Computer-aided modelling and simulation"
Academic Press 1982

Sridharan N, Lantz B, Bresina J, Goodson J
"AIMDS user manual"
Technical Report mai 1984
Rutgers University Laboratory for computer science reasearch

Steels L
"ORBIT : an applicative view of object oriented programming"
Integrated Interactive Computer Systems
North Holland Publishing Company / ECICS 1983

Stefik M
"An examination of a frame-structured representation system"
Stanford Heuristic Programming Project
Memo H-PP-78-13 Sep 1978

Stefik M,Aikins J,Balzer R,Benoit J,Birnbaum L,Hayes-Roth F,Sacerdoti
"The organization of expert systems,A tutorial "
Artificial Intelligence, 18, 1982. (p 135-173)

Suwa M, Scott A.C, Shortliffe E
"An approach to verifying completeness and consistency in a rule-based expert system"
The AI Magazine Fall 1982, (p 16-21)

Szolovits P,Hawkinson L,Markin WA
"An overview of OWL,A language for knowledge representation "
Rapport, MIT Laboratory for Computer science 1977

Trigoboff M, Kulikowski C.A
"IRIS : a system for the propagation of inferences in a semantic net"
IJCAI5 Bloc 6.0 Cambridge (MA) 1977 (p 274-280)

Umano M,Mizumoto M,Tanaka K
"A system for fuzzy reasoning"
Proc 6th IJCAI Aout 1979
Tokyo (p 917-919)

Vignard P
"Filtrage flou avec une représentation des connaissances centrée objets"
proposé au 5ième Congrés AFCET
Reconnaissances des formes et Intelligence Artificielle
Grenoble, 25-29 Novembre 1985

a/ Vignard P
"Un logiciel de base pour élaborer des systèmes experts : CRIQUET"
Colloque International d'Intelligence Artificielle
Marseille 24-27 Octobre 1984

b/ Vignard P
"CRIQUET : un outil de base pour construire des systèmes experts"
Rapport de Recherche n 316 INRIA Juillet 1984

c/ Vignard P
"Représentations de connaissances et mécanismes d'exploitation"
Document INRIA Sophia Antipolis, 84 p
Support de cours d'Intelligence Artificielle
3ième cycle d'informatique, Faculté de Nice

Vignard P
"Système interactif extensible d'aide à la modélisation et expertise économétrique"
Rapport de DEA INPG Juin 1983

Vivet M
"Calcul algébrique et représentation de connaissances mathématiques"
AFCET Nancy Reconnaissances des formes et
Intelligence Artificielle 1981

a/ Waterman DA,Hayes-Roth F
"Pattern-Directed inference systems"
Academic Press New york 1978

b/ Waterman DA,Hayes-Roth F
"An overview of pattern directed inference systems"
in Pattern-Directed Inference Systems
Academic Press 1978

Weinreb D,Moon D
"Flavors : Messages passing in the LISP Machine "
AI Memo n 602 (working paper), November 1980
Massachusetts Institute of Technology
Artificial Intelligence Laboratory

Weiss S,Kulikowski C
"EXPERT : a system for developping consultation models "
Proc IJCAI 79, (p 942-947)

Winograd T
"Frame representation and the declarative/procedural controversy"
in "Representation and understanding"
Bobrow DG, Collins A (ed), Academic Press 1975

Winston PH,Horn BKP
"LISP"
Addison Wesley Publishing Company 1981

Winston PH
"Learning by creating and justifying transfer frames"
Artificial Intelligence : an MIT perspective
volume 1, The MIT press 1980

Winston PH
"Artificial Intelligence"
Addison Wesley Publishing Company 1979

Winston PH
"The psychology of computer vision "
PH Winston editor
Mac Graw-Hill Book Company 1975

Woods W
"What's important about knowledge representation"
Computer October 1983
IEEE Computer Society

Yager R
"Linguistic representation of default values in frames"
IEEE Transactions on systems,man and cybernetics
vol snc 14, n=4, july/august 1984 (p 630-633)

Zadeh L
"Commonsense knowledge representation based on fuzzy logic"
Computer October 1983
IEEE Computer Society

Zadeh L
"A theory of approximative reasoning "
Machine Intelligence 9, 1979, (p 149-194)

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique

AUTORISATION de SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974

VU le rapport de présentation de Monsieur F. RECHENMANN, Ingénieur de recherche

Monsieur Philippe VIGNARD

est autorisé à présenter une thèse en soutenance pour l'obtention du titre de DOCTEUR de TROISIEME CYCLE, spécialité "Informatique".

Fait à Grenoble, le 21 mai 1985

Le Président de l'I.N.P.-G

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président,

RESUME

Une représentation des connaissances centrée objets, déclarative et uniforme, est présentée. Elle permet de construire une base d'objets dynamique.

Le mécanisme d'exploitation associé est fondé sur un processus élémentaire de filtrage flou. De façon général, il permet l'exploitaton d'une base d'objets dans laquelle les traitements sont aussi spécifiés de façon déclarative. Il permet aussi la manipulation de termes du langage naturel définis à l'aide d'outils mathématiques extraits de la théorie des ensembles flous.

Le processus manipule la sémantique des objets à l'aide d'informations typées. Il calcule des distances entre objets variant entre 0 et 1 au lieu de rendre de simples réponses binaires. Deux stratégies de filtrage permettent d'utiliser des raisonnements nuancés et de diverses natures.

Ces outils sont manipulés pour élaborer un système intelligent d'aide à la modélisation mathématique en biologie.

MOTS-CLES - représentation des connaissances centrée objets
- expression de la dynamique sous forme déclarative
- exploitation d'une base d'objets
- filtrage flou
- distribution de possibilités
- distance entre objets
- sémantique des objets
- analogie, typologie

ISBN - 2-7261-0418-5