Construction incrémentale d'une base de classes correcte du point de vue des types

Cécile Capponi
INRIA Rhône-Alpes
46 avenue Félix Viallet - 38031 Grenoble - France
tel : (33) 76 57 47 99
e-mail : ccapponi@nuptse.imag.fr

Mathias Chaillot
LETI (CEA - Technologies Avancées)
DSYS - CENG - 85 X - 38041 Grenoble - France
tel : (33) 76 88 37 30
e-mail : chaillot@dsys.ceng.cea.fr

RÉSUMÉ : La gestion de la cohérence des bases de connaissances nécessite avant tout un formalisme complet doté d'outils de compilation et de preuves des algorithmes mis en œuvre. Le modèle objet pour la représentation des connaissances est privé d'un tel formalisme, et la plupart des systèmes de représentation des connaissances par objets (comme par exemple les langages terminologiques) utilisent les formalismes logiques malgré leur faible pouvoir d'interaction avec les utilisateurs.
S'inspirant de l'approche des langages à objets, cet article introduit le formalisme des *types* dans un modèle de représentation et manipulation des connaissances par objets. Partant du problème de la description et du positionnement d'une nouvelle *classe* d'objets dans une hiérarchie ordonnée par la *relation de spécialisation* (construction d'une base de connaissances), la pertinence de la notion de type d'attribut est mise en évidence. Une relation de *sous-typage* est établie en accord avec la relation de spécialisation, et permet la construction d'un *graphe des types d'un attribut*. Cette structure de graphe offre plusieurs avantages, autant au niveau vérifications formelles de cohérence qu'au niveau interaction avec l'utilisateur.
Nous montrons comment la relation de sous-typage sur des types d'attributs permet de vérifier la relation de spécialisation ; ainsi, les solutions proposées pour résoudre le problème du positionnement d'une nouvelle classe (processus appelé *classification des classes*) sont étendues aux traitements de toute modification de la structure d'une base, chaque modification étant préalablement définie par une sémantique naturelle traduite en série de transactions élémentaires.

MOTS-CLÉS : Représentation de connaissances à objet, classification, spécialisation, cohérence, vérification, dynamique des classes, type.

1. Introduction

Cette étude se place dans le cadre des systèmes de représentation de connaissances par objets ayant une approche classes/instances. Ces systèmes distinguent deux types de connaissances : la *connaissance générique*, modélisée par des classes, et la *connaissance particulière*, représentée par les instances de classes.
Une base de connaissances étant continuellement modifiée du fait de l'évolution du domaine modélisé et de la perception que l'on en a, il est important de disposer de systèmes permettant la vérification et le maintien de la consistance des liens entre entités structurelles d'une base, comme par exemple les liens de spécialisation entre classes d'instances. En effet, ces liens sont le principal support de la cohérence et la correction d'une base, et sont remis en cause lors de chaque insertion, modification ou suppression d'une entité structurelle. Une gestion manuelle s'avère dangereuse, voire impossible, à partir du moment où les bases manipulées ont une taille importante, comme par exemple les bases en biologie moléculaire qui atteignent d'ores et déjà plus de 4000 classes d'objets [Perrière 91]. En effet, un utilisateur connaît rarement tous les

liens - explicites et implicites - à considérer, et il omet donc certaines répercussions à réaliser pour garantir la cohérence de la base ou ne tient pas compte du caractère récursif de certains effets de bord. Cette gestion est encore compliquée par le fait que plusieurs utilisateurs peuvent travailler sur une même base, et y apporter successivement leurs propres ajouts et modifications : le risque d'apparition d'incohérences sémantiques et structurelles augmente alors. Une automatisation au moins partielle des processus de gestion liés à l'évolution d'une base de connaissances est donc nécessaire.

Deux aspects de l'évolution d'une base de connaissances doivent être distingués :

- l'acquisition et l'intégration de nouvelles entités structurelles (descriptions de classes d'objets) lors de la construction de la base,
- la modification ou la suppression d'entités structurelles existantes.

Ces deux points participent à un processus incrémental : la hiérarchie se construit, évolue et s'améliore en fonction des résultats obtenus et des connaissances progressivement acquises, jusqu'à se stabiliser lorsque les résultats sont satisfaisants. Un traitement déterministe de l'évolution nécessite l'intégration d'un formalisme dans le modèle objet, autant pour des facilités d'expression des actions à effectuer, que pour la preuve de correction et complétude des algorithmes établis. L'acquisition de nouvelles classes est un exemple de modification que l'on peut effectuer sur la structure d'une base de connaissances : séparer ce problème particulier des points de vue algorithmique et interactif mène à la généralisation des solutions fournies pour le traitement de toute autre modification garantissant la cohérence et la correction de la base. Les deux types de problèmes nécessitent une analyse sémantique et doivent être mis en œuvre à partir d'un support strict. Leurs traitements nécessitent le maintien de la validité des liens de spécialisation : il s'agit de formaliser cette relation, qui peut être décomposée sur les éléments de granularité inférieure à celle des classes : la projection de la relation de spécialisation sur les attributs des classes introduit une relation plus spécifique sur les types des attributs, appelée relation de sous-typage. La recherche ou la vérification de liens de sous-typage permettent ainsi la recherche ou la vérification des liens de spécialisation.

La *section II* définit les notions nécessaires à l'application de notre solution pour la classification des classes et expose les bases de cette approche qui est l'établissement d'un système ordonné de types sur les attributs de la base, reflétant les liens de spécialisation entre classes. La *section III* expose le principe d'utilisation des types des attributs pour la recherche de liens de spécialisation, basé sur le fait qu'une classe est un enregistrement d'attributs et que la relation de spécialisation est fondée sur les spécifications et raffinements d'attributs ; le principe de classification de classes y est donc détaillé, et la *section IV* compare notre approche avec celle des langages terminologiques. La *section V* met en évidence la facilité d'adaptation de la solution (proposée pour l'insertion de nouvelles classes) au problème de gestion complète de la cohérence d'une base de connaissances pour toute modification apportée sur la structure de cette base. Enfin, la *section VI* donne un exemple de mise en œuvre des solutions fournies sur un exemple simple.

2. Graphe des types dans une représentation par objets

Notre problématique est l'aide à la description des entités génériques et leur insertion dans une structure hiérarchisée d'entités de même granularité. Il s'agit du processus de *classification des classes dans une hiérarchie de spécialisation*, les classes étant la modélisation des entités génériques et la relation de spécialisation étant le lien sémantique principal entre les classes, à respecter. La classification d'une classe, à distinguer de la classification des instances qui détermine la classe d'appartenance d'une instance [Mariño& 90] [Napoli 92], revient donc à chercher les liens de spécialisation entre les classes existantes dans la hiérarchie de la base et la nouvelle classe décrite dans le formalisme considéré.

2.1. Classes et spécialisation de classes

Une classe associe une structure à un ensemble d'instances. La structure est constituée d'un regroupement de définitions d'attributs. Un attribut est défini par son nom, son domaine de valeurs, et éventuellement par des attachements procéduraux et une valeur par défaut. Une

valeur d'attribut est un élément du domaine de cet attribut. Une instance d'une classe est définie par son nom et par un regroupement de valeurs des attributs définis dans la structure de cette classe (figure 1).
La définition de la structure d'une classe est construite incrémentalement à partir du schéma de cette classe. Un schéma de classe est une unité syntaxique qui identifie toutes ses super-classes directes, et qui définit partiellement ses attributs. Les classes étant définies à partir de leurs super-classes, on parle de spécialisation (éventuellement multiple). Une sous-classe d'une classe donnée constitue une spécialisation de cette classe. Le graphe de spécialisation est construit par transitivité de la relation de spécialisation.

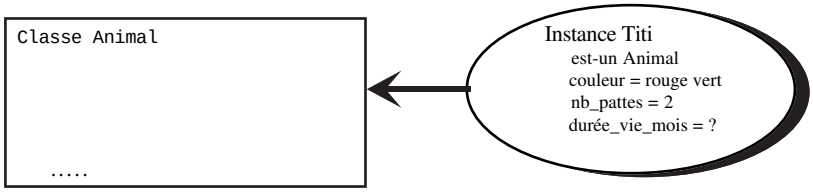

Figure 1 : Classe et instance. *Titi* est instance de la classe *Animal*. Le schéma de classe exprime les propriétés communes aux instances par des attributs. Une instance de cette classe donne des valeurs aux attributs, en accord avec les spécifications.

2.2. Attributs et spécialisation de classes

Le schéma d'une classe définit un attribut en associant à son nom une spécification complète ou partielle de son domaine. Pour construire la définition de la structure d'une classe, le mécanisme d'héritage établit qu'une classe hérite les définitions d'attributs de ses super-classes, et qu'elle complète ces définitions par celles qui sont données dans son schéma. Le schéma d'une classe peut donc définir de nouveaux attributs qui ne l'étaient pas dans les super-classes, ou associer à des attributs existants des sous-domaines des domaines associés à ces attributs dans les super-classes (figure 2).

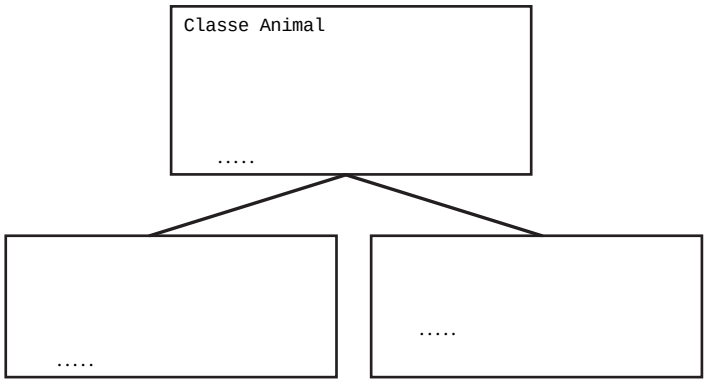

Figure 2 : Graphe de spécialisation. Les sous-classes définissent localement leurs attributs propres et peuvent éventuellement raffiner ceux des super-classes. Par exemple, la classe *Oiseau* redéfinit l'attribut *nb_pattes*, introduit *nb_ailes* et garde la spécification de *couleur* donnée dans la classe *Animal*.

La structure finale d'une classe résulte de l'application du mécanisme d'héritage pour définir complètement tous ses attributs. La définition d'un attribut varie selon la classe considérée. Les différentes définitions d'un attribut peuvent donc être identifiées dans le graphe de spécialisation. Pour être compatible avec la relation de spécialisation (notée $\leq_c$), un attribut dans une classe *C1* doit correspondre à un sous-domaine des domaines associés à cet attribut dans toutes les super-classes de *C1* qui comprennent cet attribut.

2.3. Sous-typage et graphe des types

L'intérêt de l'introduction de la notion de type vient de l'importance de disposer pour le modèle de représentation d'un formalisme qui permette de prouver la correction de cette représentation et la complétude des mécanismes de manipulations de connaissances mis en œuvre.

2.3.1. Normalisation et type d'un attribut

La définition d'un attribut dans une classe résulte de la combinaison des définitions associées à cet attribut dans les super-classes de cette classe. En particulier, le domaine de valeurs de cet attribut est défini par toutes les restrictions héritées des super-classes. Un même domaine de valeurs peut ainsi être défini de plusieurs façons, en combinant des restrictions de domaine en différents nœuds du graphe de spécialisation. Pour comparer les domaines de valeurs associés à un attribut dans des classes différentes, il importe donc de disposer d'une forme normalisée pour sa définition. Cette forme normalisée correspond au type de cet attribut.
Le type d'un attribut est défini par son domaine de valeurs entièrement spécifié. Le type d'un attribut dans une classe *C1* est donc déterminé par toutes les spécifications du domaine de cet attribut qui apparaissent dans *C1* et dans les super-classes de *C1*.
Les spécifications du domaine d'un attribut dépendent de la nature de cet attribut. Le domaine d'un attribut peut être concret ou abstrait. Un domaine concret est spécifié par application des opérateurs de construction de domaines concrets (intersection, union, énumération ...) sur un domaine primitif (réels, entiers, chaînes ...). La spécification de ce domaine peut être complétée par des contraintes qui utilisent les opérateurs d'égalité, de différence et de comparaison. Un domaine abstrait correspond à un sous-ensemble des instances d'une classe. Dans la spécification d'un domaine abstrait, un nom de classe dénote l'ensemble des instances de cette classe.

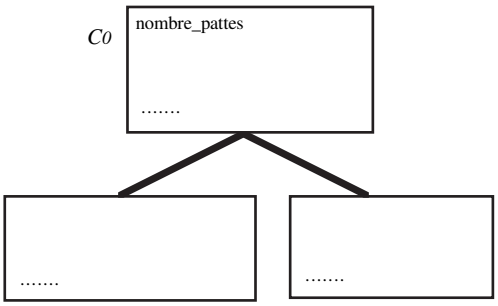

Figure 3 : Nécessité de la normalisation des définitions d'attributs. Les définitions de nombre_pattes dans C1 et C2 sont équivalentes, même si elles sont exprimées différemment. Afin d'obtenir le type de ces différentes définitions, il convient de récupérer toute l'information dans la (les) super-classe(s) ; l'équivalence entre les deux descriptions ne pourra être déduite que si l'on convient d'une forme unique d'expression.

Les domaines associés à un même attribut dans des classes différentes peuvent être comparés à l'aide des types associés respectivement à ces définitions d'attribut. La relation de sous-typage (notée $\leq_\tau$) est ainsi équivalente à la relation d'inclusion portant sur les domaines des attributs. Un type *t1* est sous-type d'un type *t2*, si et seulement si *t1* correspond à un domaine inclus dans le domaine auquel correspond *t2*.
Notons que dans le cas où le domaine de l'attribut est un ensemble d'instances d'une classe de la base, le type de cet attribut est un *type-classe*, c'est-à-dire l'ensemble des instances de la classe considérée. Dans ce cas, la relation de sous-typage correspond à la relation de spécialisation (une correspondance valeur/type - instance/classe est réalisée implicitement).

2.3.2. Graphe des types d'un attribut

Le calcul du type d'un attribut pour chaque classe du graphe de spécialisation permet de construire le graphe des types de cet attribut.

2.3.2.1. Structure du graphe des types d'un attribut

A chaque nœud du graphe des types d'un attribut sont associés le type et la liste des classes pour lesquelles ce type correspond à l'attribut (figure 4).

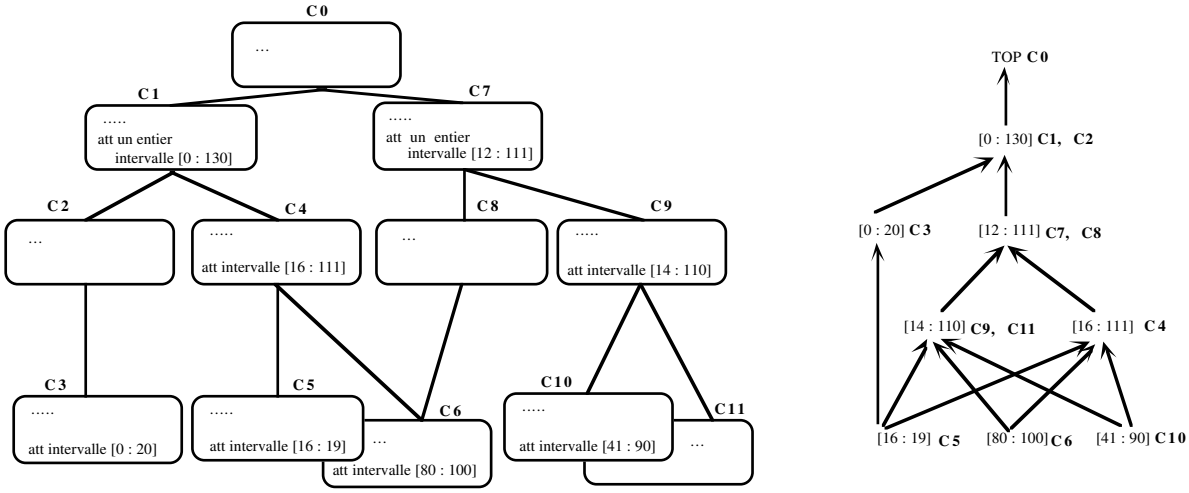

Figure 4 : Graphe de spécialisation avec les définitions de l'attribut *att*, et graphe des types de *att* correspondant.

2.3.2.2. Construction du graphe des types à partir du graphe de spécialisation

La construction du graphe des types d'un attribut se fait en parcourant le graphe de spécialisation. Soit Top, le type associé à tout attribut dans une classe pour laquelle il n'est pas défini. Soit GT_a le graphe des types associés à un attribut a. Un algorithme possible pour construire ce graphe est le suivant :

Pour toute classe C, nœud du graphe de spécialisation (parcours en profondeur d'abord) :
si a est défini ou modifié dans la classe C alors :
calcul du nouveau type t associé à a dans C
insertion du nouveau nœud (t , $\{C\}$) dans GT_a
sinon
soit t le type associé à a dans C (Top si a n'est pas défini dans C)
ajout de C à la liste l du nœud (t, l) déjà inséré dans GT_a

Cet algorithme n'est pas le plus efficace. Il garantit un ordonnancement correct de tous les types associés à a dans le graphe de spécialisation, vis-à-vis de la relation de sous-typage.
L'insertion d'un nouveau type dans le graphe notée insert(t, GT_a) doit être garante de sa structure ; en particulier, elle ne doit pas introduire de boucle, ni d'arcs redondants en respectant la propriété de transitivité de la relation de sous-typage. Pour des questions d'efficacité, elle est réalisée par un parcours ascendant du graphe des types.

2.3.2.3. Propriété du graphe des types

L'intérêt du graphe des types pour un attribut est qu'il est compatible avec la position de cet attribut dans le graphe de spécialisation :

$$\forall\ (t\ ,\ l) \in GT_a,\ \forall\ (t'\ ,\ l') \in GT_a$$
$$\forall\ c \in l,\ \forall\ c' \in l'\ :\ c \leq_c c'\ \Rightarrow\ t \leq_\tau t'$$

Cependant, le graphe des types d'un attribut n'est pas plongé dans le graphe de spécialisation, car il peut exister un chemin entre deux nœuds du graphe des types qui correspondent à des classes non comparables pour la relation de spécialisation. Le graphe de spécialisation n'est pas

non plus plongé dans le graphe des types car un nœud de ce dernier peut faire référence à plusieurs classes.

3. Position d'une classe dans le graphe de spécialisation

La position d'une classe dans le graphe de spécialisation peut être calculée à partir des graphes des types des attributs de la classe.

3.1. Position d'une classe par rapport à un attribut

L'information contenue dans le graphe des types d'un attribut *a* permet de donner la position *possible* d'une (nouvelle) classe *C* possédant *a*, relativement à cet attribut . En effet, le graphe des types associé à l'attribut *a* fournit d'une part la liste des super-classes possibles de *C* et d'autre part la liste des sous-classes possibles de *C*, vis-à-vis du type associé à l'attribut *a* dans *C*.
Les sous-ensembles de super-classes et de sous-classes de C par rapport à l'attribut a dans le graphe de spécialisation sont obtenus respectivement à partir des expressions suivantes :

$$\sup(a, C) = \bigcup_{\substack{(t, l) \in GT_a \\ t \geq_\tau t_0}} l \qquad \inf(a, C) = \bigcup_{\substack{(t, l) \in GT_a \\ t \leq_\tau t_0}} l$$

L'utilisation de ces deux expressions en parcourant le graphe des types permet de construire les deux ensembles recherchés.

3.2. Combinaison des positions par rapport à tous les attributs de la classe

La définition de la structure d'une classe comprend plusieurs attributs. La position d'une classe dans le graphe de spécialisation s'obtient par combinaison des positions calculées par rapport à tous ses attributs. Cependant, pour qu'une classe C_1 puisse être sous-classe d'une classe C_2, il faut que tous les attributs définis dans C_2 soient définis dans C_1 (propriété de l'*héritage complet*). Soit *A(C)* l'ensemble des attributs d'une classe *C*. Les super-classes possibles de *C* et les sous-classes possibles de *C* dans le graphe de spécialisation sont données respectivement par les expressions suivantes :

$$\sup(C) = \left[\bigcap_{a \in A(C)} \sup(a, C) \right] \bigcap \{ C' \ / \ A(C') \subseteq A(C) \}$$

$$\inf(C) = \left[\bigcap_{a \in A(C)} \texttt{inf}\ (a, C) \right] \bigcap \{ C' \ / \ A(C) \subseteq A(C') \}$$

Le résultat final donnant la position possible du nouveau schéma *C* peut contenir plusieurs solutions pour les super-classes, qui ne sont pas forcément reliées par la relation de spécialisation. Dans le cas de l'héritage simple, l'utilisateur choisit une des super-classes pour fixer la position. Cependant, toutes les candidates sont correctes, et peuvent être considérées comme des solutions conjointes ; en effet, considérons que l'héritage multiple impose la condition suivante :

> "si la classe *C* admet *n* super-classes *C1, ... , Cn*, alors *C* est une conjonction de *C1, ... , Cn* ; c'est-à-dire que le type de *a* dans *A(C)* doit être successivement sous-type du type de *a* dans *A(C1)* lorsqu'il existe, ... , sous-type du type de *a* dans *A(Cn)* lorsqu'il existe."

Cette condition est vérifiée par la position finale fournie par le processus, puisque le calcul des ensembles des super-classes et des sous-classes est aussi un calcul de conjonctions (ce sont des intersections). Notons toutefois que, de la même façon que pour l'héritage simple, si plusieurs classes sont candidates, toutes ne seront pas forcément retenues compte-tenu de la transitivité de l'héritage.

3.3. Algorithme de classification de classe

L'algorithme de classification qui a été développé peut être mis en œuvre dans deux circonstances : lors de la *classification interactive* d'une nouvelle classe et lors de la *classification automatique* d'une classe déjà existante, qui à la suite d'une modification ne respecte plus les propriétés du graphe de spécialisation. L'algorithme proposé ci-dessous correspond au cas le plus général. En particulier, il existe plusieurs variantes au niveau de la simultanéité des différents calculs de positions.

• /* Initialisations */
Soient C la classe à décrire, $A(C)$ la liste initialement vide des noms de ses attributs, B la base de connaissances, $\mathcal{C}(B)$ les classes de B, et $A(B)$ les noms des attributs de B.

• /* Lecture des descriptions d'attributs et calcul de la position intermédiaire */
tant que il reste un attribut de nom *n-att* à décrire **faire** :

si $n\text{-}att \notin A(B)$
alors soit *att* le type associé à sa description dans le modèle, préalablement normalisée.
calculer $sup(att, C) \leftarrow \mathcal{C}(B)$ et $inf(att, C) \leftarrow \emptyset$
sinon **calculer** $GT_{n\text{-}att}$ dans B => soit *att* la descriptionde *n-att* dans C
insert$(att, GT_{n\text{-}att})$ => soit $n(att, GT_{n\text{-}att})$ le nœud de *att* dans $GT_{n\text{-}att}$.
calculer les positions de C relatives à *att*

$$sup(att, C) \leftarrow \{ l , (t, l) \geq_\tau n(att, GT_{n\text{-}att}) \} \cup \{ C , n\text{-}att \notin C \}$$

$$inf(att, C) \leftarrow \{ l , (t, l) \leq_\tau n(att, GT_{n\text{-}att}) \}$$

fsi
calculer les positions intermédiaires de C (ne tenant pas compte de l'héritage complet)

$$sup\text{-}inter(C) \leftarrow sup(att, C) \cap sup\text{-}inter(C)$$

$$inf\text{-}inter(C) \leftarrow inf(att, C) \cap inf\text{-}inter(C)$$

calculer $A(C) \leftarrow A(C) \cup \{n\text{-}att\}$
ftq

• /* Calcul des super-classes et sous-classes compte-tenu de l'héritage complet */

$$sup(C) \leftarrow \{ C' \in sup\text{-}inter(C) / A(C') \subseteq A(C) \}$$

$$inf(C) \leftarrow \{ C' \in inf\text{-}inter(C) / A(C) \subseteq A(C') \}$$

• /* Insertion dans la base de connaissances */
si $sup(C) \neq \emptyset$
alors insertion
sinon $inf(C) = sup(C) = \emptyset$ /* pas de position possible */

4. Comparaisons avec les logiques de description

Les langages terminologiques, prolongement des travaux effectués sur KL-ONE [Brachman& 85], représentent les connaissances conceptuelles et taxonomiques d'un domaine au niveau abstrait de la logique. A partir de concepts et de rôles, ils définissent de nouveaux concepts en utilisant les opérateurs proposés par le langage. Pour aborder les applications réelles, les langages terminologiques sont étendus aux domaines concrets [Baader& 91]. Ils référencent donc les types concrets structurés à partir de la relation de subsomption qui porte sur des termes logiques.
A l'inverse, les langages de classes organisent des types concrets structurés [Cardelli& 91] avec la relation de spécialisation. Il existe une similitude entre le processus de classification des classes et le rôle du *classifieur* des langages terminologiques [Brachman& 91] [Schmolze& 83], qui permet de positionner un nouveau concept dans une taxonomie de concepts, un concept

étant indifféremment primitif, défini ou individuel. Le principe du classifieur est de déterminer les liens de subsomption entre un nouveau concept et les concepts existants dans la base, la subsomption étant une relation d'ordre partiel sur les concepts [Napoli& 92]. Ce processus a été étendu dans LOOM [MacGregor& 91] à la gestion de la maintenance des bases de connaissances : le système KREME permet non seulement de classer les concepts nouveaux, mais aussi les concepts modifiés ; il gère de plus la propagation des effets de bord dans la base, selon le même principe que les systèmes de maintenance de la vérité basés sur la justification [MacGregor& 92].
Les langages terminologiques, par la classification de concepts, construisent une hiérarchie cohérente à partir d'un ensemble de termes. Dans notre approche, la classification permet d'abord de construire une hiérarchie cohérente et aussi de vérifier la cohérence d'une hiérarchie déjà existante et arbitraire. En d'autres termes, l'approche langage terminologique permet de structurer un ensemble de termes, nous construisons et vérifions un ensemble structuré.

Il est enfin important de souligner que le graphe des types que nous construisons systématiquement pour un attribut est utile pour la récupération et le stockage des informations dans un formalisme, mais aussi pour faciliter les choix et les définitions de l'utilisateur. En effet, ce dernier peut apprécier, sous une seule structure, tous les raffinements portés sur l'attribut qu'il considère. Les méthodes développées par les langages terminologiques, basées sur les logiques de description, n'apportent pas cette dimension interactive, du fait du formalisme logique qui n'est pas toujours très simple à percevoir, en particulier pour un utilisateur quelconque [Brachman 92] [Finin 86], mais surtout à cause de l'unicité de la solution fournie par le classifieur (l'insertion d'un terme dans une hiérarchie n'admet qu'une seule position car la relation de subsomption est définie a priori) : il n'y a pas de place pour l'interaction, tout est automatique. En ce sens, le classifieur des langages terminologiques ressemble beaucoup plus à la classification des classes lorsque l'on dispose de multi-spécialisation, avec choix automatique optimal du système.

5. Utilisation des types pour la gestion de la dynamique des classes

La gestion de la dynamique d'une base de classes est la gestion, au niveau structurel, de la définition des classes et de la hiérarchie de spécialisation lors de toute transaction effectuée sur cette base. On appelle *transaction* une opération de modification de la base, comme l'ajout, la suppression et la modification d'une classe ou d'un attribut.

F. Maurer a dégagé les fonctionnalités essentielles d'un outil pour l'édition et la maintenance de bases de connaissances [Maurer 92] :

- Intégrer le fait que la construction incrémentale d'une base de connaissances s'effectue selon un ordre établi par l'utilisateur : il s'agit de respecter ses choix.
- Gérer les répercussions dues à l'insertion, la modification ou la suppression d'objets.
- Montrer les dépendances sémantiques et structurelles entre entités de connaissances.
- Montrer les effets des modifications sur la base de connaissances.
- Vérifier la correction syntaxique et la consistance sémantique de la base de connaissances.

Ainsi, un gestionnaire de la dynamique doit être capable de contrôler la validité des transactions (gestion de la cohérence sémantique de la base), ainsi que la validité des définitions internes de classes et d'attributs (gestion de la cohérence syntaxique et structurelle) : il doit être capable d'éviter les définitions redondantes, les cycles, et de garantir la validité des liens entre entités de même granularité ou de granularités différentes (spécialisation, instanciation, composition). Dans le même temps, il doit être doté d'une interface complète pour faciliter l'expression des définitions et des modifications et pouvant supporter un mécanisme de production d'explications.

Les types et la relation de sous-typage, élaborés à partir de la relation de spécialisation dont dépend la structure d'une base (graphe de spécialisation), permettent de garantir des transactions valides. En effet, chaque transaction possède sa propre sémantique et un ensemble (qu'il est possible de rendre fini) de répercussions sur la base. Si le gestionnaire connaît ces

répercussions, il est capable de mettre en œuvre les vérifications adéquates pour éviter conflits et erreurs. Ainsi, pourvu d'un algorithme particulier pour chaque transaction possible (celui de l'insertion d'une classe en est un), le gestionnaire de la dynamique anticipera par des vérifications de types les erreurs possibles ; dans le cas d'une transaction sans erreur (ou avec des erreurs corrigées), le traitement physique de la transaction sera immédiat et sans risque, et la cohérence de la base assurée (tant au niveau sémantique que structurel).

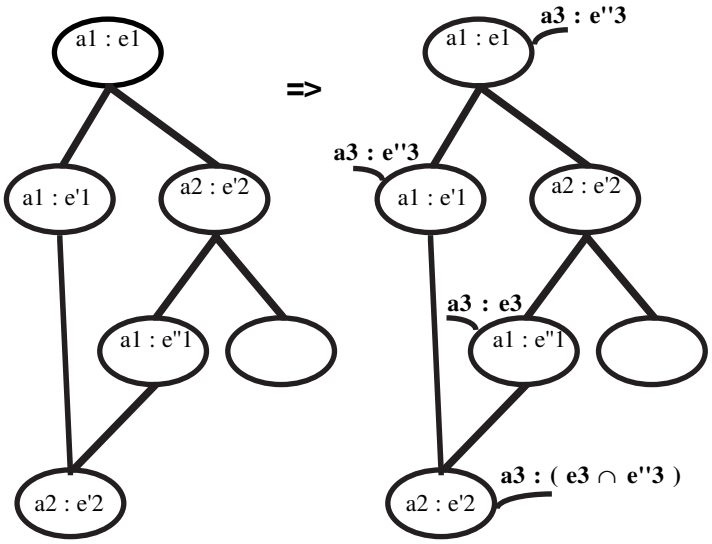

Figure 5 : Exemple de transaction : ajout d'un attribut dans une classe, donc introduction d'une nouvelle spécification pour cet attribut existant déjà dans la base.

La figure 5 montre les répercussions de l'ajout d'une nouvelle spécification d'un attribut existant dans la base. Cette transaction doit s'effectuer en quatre temps :

(1) Lecture (par le biais d'une interface adaptée) de la description de la nouvelle spécification de l'attribut (donnée pour une classe particulière)
(2) Calcul du graphe des types de cet attribut permettant la vérification de la relation de spécialisation (cette introduction ne doit pas remettre en cause des liens de spécialisation)
(3) En cas de respect de la relation de spécialisation, propagation de la modification dans les sous-classes (et éventuellement super-classes)
(4) Répercussions sur les instances des classes modifiées.

6. Mise en œuvre

Les méthodes proposées dans cet article ont été développées sur le modèle de représentation des connaissances Shirka (chap 8.5 de [Masini& 89]) et testées sur des bases de connaissances contenant environ 700 classes. Ces méthodes ont permis une nette amélioration des processus de création, insertion et modifications de classes d'objets, tant au niveau de la description des actions à effectuer (interface de développement) qu'aux niveaux répercussions cohérentes et validation.

L'exemple de la figure 6 montre une partie de l'interface développée autour des algorithmes de classification de classes. Il se déroule dans un exemple de base de connaissances destinée à aider l'identification de spécimens de grands échassiers européens. L'utilisateur cherche à définir l'attribut de nom *taille-cm* : le système lui fournit le graphe des types de cet attribut, et l'utilisateur choisit une des descriptions déjà présentes. Une fois la classe entièrement décrite, le système fournit à l'utilisateur la (les) positions(s) possible(s), et fournit en option la position optimale, c'est-à-dire les super-classes les plus spécifiques et les sous-classes les plus générales.

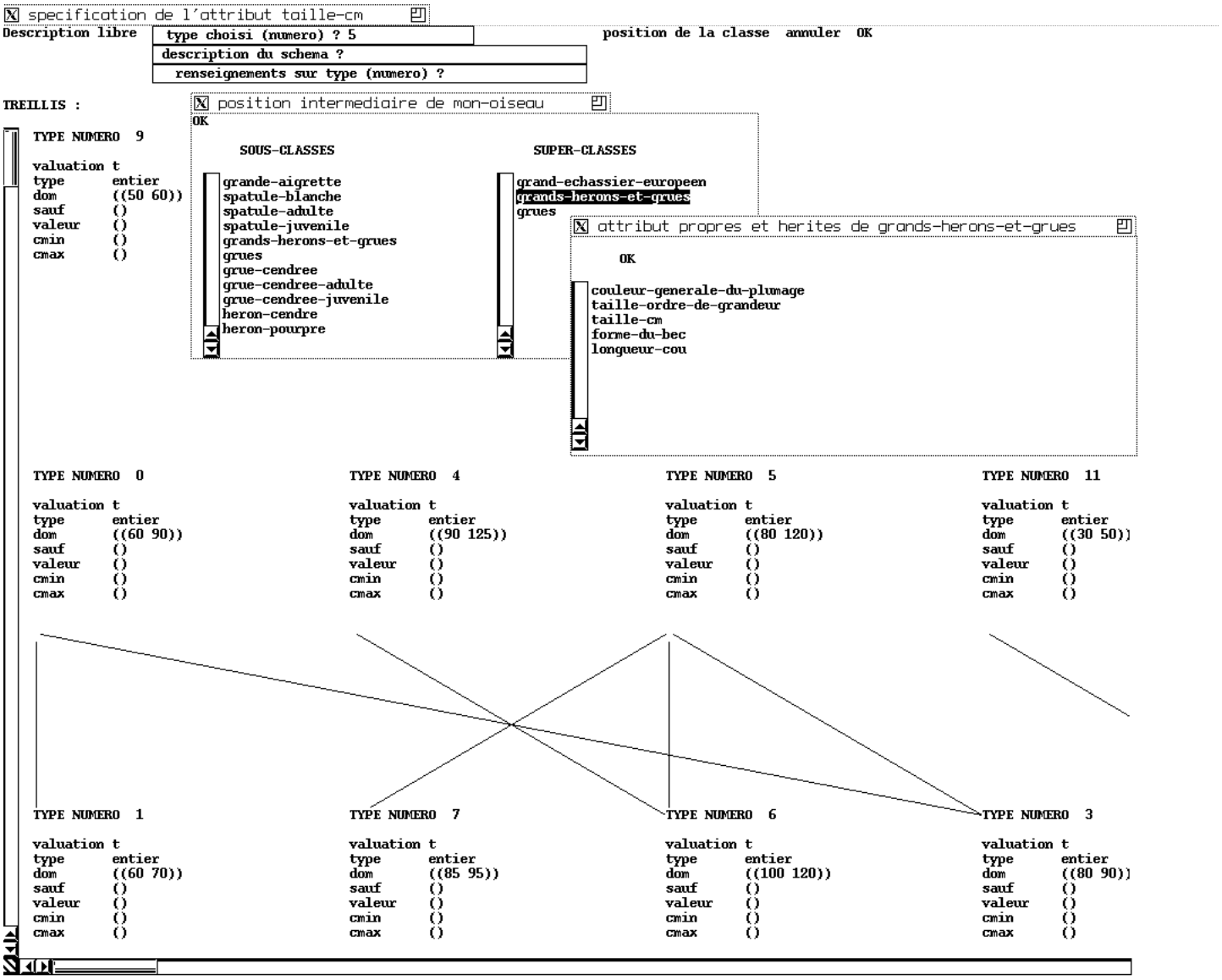

Figure 6 : Graphe des types de l'attribut *taille-cm*. Une fois sa description donnée, l'utilisateur visualise la position intermédiaire de la nouvelle classe, et peut obtenir des renseignements sur les attributs des super-classes ou sous-classes possibles.

Derrière l'interface de manipulation de la structure de la base existent plusieurs modules de gestion de la dynamique des classes dont l'architecture logicielle est détaillée sur la figure 7.
Il est important de constater que la notion de type apporte, dans un modèle objet, un formalisme important qui n'existe pas dans le domaine des représentations des connaissances : les possibilités de vérifications sont alors plus développées, plus sûres et plus efficaces, compte-tenu de tous les outils formels associés à ce formalisme. Les types permettent à la fois la définition sémantique des structures du modèle, mais aussi la définition sémantique et la mise en œuvre des mécanismes de manipulation de ces structures. En particulier, les preuves de correction et de complétude des algorithmes de vérification par les types sont réalisables [Borgida 92], alors qu'elles ne sont pas possibles si l'on reste au niveau sémantiquement incomplet des objets.

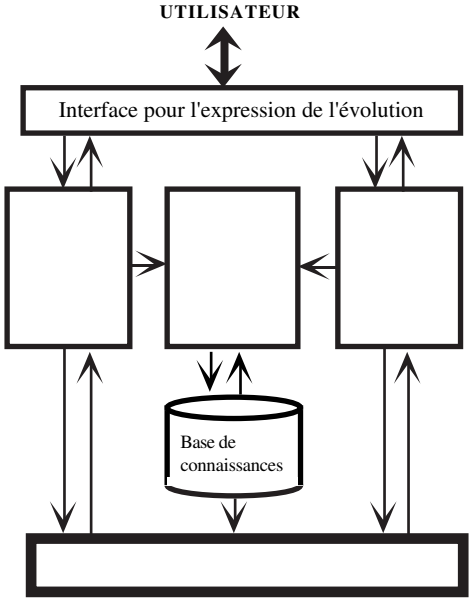

Figure 7 : Architecture logicielle pour l'édition et la maintenance de bases de connaissances.

7. Conclusion

Le formalisme apporté par les types, pour la définition d'attributs de classes dans un modèle de représentation des connaissances par objets, est relativement naturel puisque les attributs spécifient des ensembles, et apporte de plus un niveau d'abstraction supplémentaire non négligeable pour la mise en œuvre de processus de validation.
Le graphe des types d'un attribut, introduit initialement pour la classification des classes, offre une autre perception du graphe de spécialisation - du point de vue d'un seul attribut - qui permet de mettre en évidence des liens entre classes qui n'apparaissent pas toujours dans le graphe de spécialisation. D'une part, l'utilisateur peut s'appuyer sur les types pour exprimer les spécifications de chaque attribut de la nouvelle classe. D'autre part, le graphe des types contient une information correcte et complète sur l'attribut, du fait de sa construction liée à la relation de sous-typage. Ainsi, les preuves de la correction et de la complétude de l'algorithme de classification des classes sont évidentes. Cette approche de la classification des classes, par *types* ou *profils d'attributs*, offre d'autres avantages importants. Elle est en fait fondée sur une décomposition du problème de classification en n sous-problèmes successifs (où n est le nombre d'attributs de la nouvelle classe) : on réalise n classifications d'attributs successives puis on considère l'intersection des solutions obtenues. La décomposition permet la simultanéité des tâches de définition des attributs et de classification de la classe les regroupant. Cette parallélisation est importante pour la compréhension des explications produites (et des messages d'erreurs) car elles sont fournies dans le *contexte* immédiat de la définition : le processus de classification est ainsi plus facilement assimilé par l'utilisateur.
Le processus de classification de classes est un cas particulier de modification d'une base de connaissances. En fait, le traitement de toute modification d'une base peut être projeté sur les attributs : la relation de sous-typage permet en effet la formalisation, et par là même des vérifications correctes [Laurent 92], de la relation de spécialisation, donc de la cohérence de la base.
Outre la formalisation de la relation de spécialisation, les types permettent la formalisation d'autres relations telles que la composition ou l'instanciation, jusqu'à la définition formelle de tous les mécanismes d'inférence et de déduction appliqués sur les structures du modèle de représentation. Pour aller plus loin, le formalisme associé aux types permet une formalisation partielle des modèles à objets pour la représentation des connaissances.
Toutefois, certains problèmes restent ouverts. En particulier faut-il distinguer nettement les notions de type et de classe comme dans certains langages de programmation à objets ? Mais surtout, quel formalisme de type faut-il adopter pour quel modèle objet ?

Références

[Baader& 91] F. Baader, P. Hanschke, *A Scheme for Integrating Concrete Domains into Concept Languages*, proceedings of 12th IJCAI, Sydney, août 1991.
[Borgida 92] A. Borgida, *From type systems to knowledge representation: natural semantics specifications for description logics*, Int. Jour. of Intelligent and Cooperative Information Systems, vol. 1 n°1, April 1992, pp 93-126.
[Brachman& 91] R.J. Brachman, D.L. McGuinness, P.F. Patel-Schneider, L. A. Resnick, A. Borgida, *Living with CLASSIC, When and How to Use a KL-ONE-Like Language*, Sowa (ed), Principles of Semantics Networks, Explorations in the Representation of Knowledge, Morgan Kaufmann Publisher, 1991.
[Brachman& 85] R.J. Brachman, J.G. Schmolze, *An overview of the KL-ONE Knowledge Representation System*, Cognitive Science n°9, 1985.
[Brachman 92] R.J. Brachman, *Reducing CLASSIC to Practice : Knowledge Representation Theory Meets Reality*, proceedings of 3rd International Conference on Principles of Knowledge Representation and Reasoning, Cambridge (MA), Octobre 1992.
[Cardelli& 91] L. Cardelli, J. C. Mitchell, *Operations on records*, Math. Struct. in Comp. Science, vol.1 n° 1, pp 3-48, Mars 1991.
[Finin 86] T.W. Finin, *Interactive Classification : a Technique for Acquiring and Maintaining Knowledge Bases*, proceedings of the IEEE, vol 74, n°16, octobre 1986.
[Laurent 92] J.P. Laurent, *Proposal for a Valid Terminology in KBS Validation*, proceedings of 10th ECAI, Vienne, août 1992.
[Maurer 92] F. Maurer, *Knowledge Base Maintenance and Consistency Checking in MOLTKE/HyDi*, proceedings of EKAW'92, Heidelberg and Kaiserslautern, Germany, Lecture Notes in Computer Science n°599, mai 1992.
[MacGregor& 91] R. MacGregor, M.H.Burstein, *Using a Description Classifier to Enhance Knowledge Representation*, IEEE Expert vol. 6 n°3, juillet 1991.
[MacGregor& 92] R. MacGregor, D.Brill, *Recognition Algorithms for the LOOM classifier*, proceedings of AAAI'92, San Jose (CA), juillet 1992.
[Mariño& 90] O. Mariño, F. Rechenmann, P. Uvietta, *Multiple Perspectives and Classification Mechanism in Object-Oriented Representation*, Proceedings of the 9th ECAI, Stockolm, août 1990.
[Masini& 89] G. Masini, A. Napoli, D. Colnet, D. Léonard, K. Tombre, *Les Langages à Objets*, InterEditions, collection Informatique Intelligence Artificielle, 1989.
[Napoli 92] A. Napoli, *Représentations à Objets et Raisonnement par Classification en Intelligence Artificielle*, thèse d'état, CRIN-INRIA Lorraine, janvier 1992.
[Napoli& 92] A. Napoli, R. Ducourneau, *Subsumption in Object-Based Representations*, proceedings of ERCIM Workshop on Theoretical and Experimental Aspects of Knowledge Representation, Pise, mai 1992.
[Perrière 91] G.Perrière, *Coligene*, Traitement de l'information des séquences biologiques, cahiers IMABIO, vol. 2, pp 61-66, CNRS, Paris, 1991.
[Schmolze& 83] J.G. Schmolze, T.A. Lipkis, *Classification in the KL-ONE Knowledge Representation System,* proceedings of the 8th IJCAI, Karlsruhe, Germany, 1983.